Computer Sciences Department

Dynamic Time Windows: Congestion Control and Avoidance in High Speed Networks

Theodore V. Faber

Technical Report #1253

November 1994

UNIVERSITY OF WISCONSIN MADISON

DYNAMIC TIME WINDOWS: CONGESTION CONTROL AND AVOIDANCE IN HIGH SPEED NETWORKS

by

THEODORE V. FABER

A thesis submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN – MADISON

1994

DYNAMIC TIME WINDOWS:

CONGESTION CONTROL AND AVOIDANCE

IN HIGH SPEED NETWORKS

Theodore V. Faber

Under the supervision of Professor Lawrence H. Landweber

at the University of Wisconsin–Madison

This thesis is a description and study of Dynamic Time Windows (DTW), a new system for congestion control and avoidance in high speed computer networks. DTW's approach to controlling congestion by modulating source burstiness through the use of a *time window* is new. A time window controls a source's burstiness without affecting its throughput, in a way analogous to a conventional sliding window controlling a source's throughput. Time windows are the basis of DTW's source control algorithm, which guarantees that network congestion times are bounded. This algorithm is combined with feedback from the network which adjusts a source's time window. Adjusting time windows in response to feedback from the network causes sources to fully utilize the network while avoiding congestion. Switches in the network implement a queueing discipline that allocates resources fairly among sources.

This combination of source control, feedback, and queueing disciplines results in a powerful system that controls congestion in high speed networks. DTW is especially well suited for use in the high speed, wide area ATM networks of the future.

DTW also provides a framework for sources to tailor network performance to their quality of service requirements. Sources tailor performance by allocating resources in the network, and by specifying parameters to the time window adjustment algorithm. Each of these methods is effective, but combining them results in a more powerful

system than either alone.

The thesis investigates DTW through analysis, simulation and implementation. The property of recovering from congestion without feedback, called DTW stability, is proven mathematically. Proving this stability property for multiple switches in series provides a general intuition into how switches change traffic patterns by queueing. The feedback system is studied through simulation, including comparisons with major existing congestion control systems. DTW compares favorably with today's systems. The quality of service mechanisms are also studied through simulation, demonstrating the power of combining the mechanisms. The implementation of DTW on AT&T's XUNET network is described, and several experiments using that system are reported. The implementation of DTW performs as predicted by analysis and simulation.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

Chapter 1

Introduction

"A beginning is the time for taking the most delicate care that the balances are correct."

— *Frank Herbert,* **Dune, ch. 1**

This thesis describes the Dynamic Time Windows congestion control and avoidance system. This system is designed to mitigate the effects of network congestion on today's networks and the networks of the future by directly controlling source burstiness.

This chapter will discuss the need for congestion control in computer networks, identify the aspects of congestion control that present particular challenges in high speed networks, and introduce the Dynamic Time Windows system (DTW). It will also provide an overview of the remainder of the thesis.

1.1. Computer Networks

A computer network may interconnect several autonomous, geographically separated computers that communicate with each other over the shared medium. Although much of networking research is applicable to interconnected computers that are not geographically separated, we include that constraint to distinguish networks from parallel processors. More important than the geographic separation is the autonomy of the two communicating entities. Although many computers may share a network, they may each be using it to perform separate tasks, unlike nodes in a parallel machine which generally cooperate on a single task.

A further distinction is made based on the physical scope of a network. A network covering a small area, at most a few kilometers, is called a local area network (LAN). One that covers a city is a metropolitan area network (MAN), and larger networks are called wide area networks (WANs). Although the distinctions are somewhat arbitrary, they are useful in that the larger a network, the larger the potential latency in communicating between two sites. The work in this thesis is concerned with WANs, because of the challenges in high latency systems.

The Internet, a worldwide network of computers that developed from the ARPANET, is the largest computer WAN currently operating in terms of the number of people using it, the geographic scope of the network, and the amount of traffic carried. In the United States, many are saying that it will evolve into the National Information Infrastructure (NII). These attributes make the Internet a logical choice to determine the requirements for a more advanced congestion control system.

1.2. Congestion Control and Avoidance

A network is said to be congested when increasing the load on it does not result in an improvement or causes a decline in a figure of merit (*e.g.,* throughput) for that network's performance. For example, a source that views throughput as a figure of merit may reach a point where increasing load on the network results in little or no increase in throughput. Increasing the load further may even result in a lowering of throughput. In this case, the source's traffic (along with the traffic of other sources) has probably loaded some network element beyond its capacity. That element has become a bottleneck in the system, and the system throughput is limited by this bottleneck's capacity. The additional load results in the bottleneck being pushed so far beyond its capacity that its performance suffers. For example, it may begin dropping some of the data that the source is sending. Other figures of merit include per–packet delay and packet loss rates.

Controlling source behavior so that the system bottleneck is utilized at its capacity, and not beyond it, is *congestion avoidance;* restoring a congested network to a stable state is *congestion control[1].* Most networks that address congestion provide both congestion control and avoidance. The avoidance systems operate at all times, and if the avoidance system fails the congestion control system is invoked. The systems are usually closely related, and it may be difficult to discern what aspects of a given network address control and what aspects address avoidance at first glance.

In contrast, DTW decouples the avoidance and control mechanisms. DTW uses allocation of resources and source control to control congestion, and feedback to tune that source control to avoid congestion while adjusting source behavior to utilize the network efficiently. This partition of function keeps the network stable in the face of high latencies, while allowing feedback to adapt the sources' behavior to the current state of the network.

A goal of congestion avoidance and control is to maximize some performance measure of the network. Examples of a performance measure include throughput and packet delivery to loss ratio. There are many characterizations of this maximization; finding the knee of the throughput versus load curve[2, 3], saturating the bottleneck switches in a network[4], and maximizing user incentives[5] have all been proposed. Maximizing user incentives involves a pricing scheme including incentives in the network arranged so that users are guided to avoid congestion economically. In each of these systems, the network provider is trying to maximize some aspect of network performance delivered to the sources, individually or in the aggregate, while avoiding congestion. DTW maximizes the amount of burstiness it will tolerate from a source while guaranteeing it an average rate, and avoiding congestion.

1.3. Trends in Today's Internet Applications

The size and bandwidth of computer networks are increasing. New users are being added to existing networks, such as the Internet, at the same time that the capacity of those networks is being increased. Researchers are investigating new high bandwidth networking technologies[6, 7] which promise to continue, if not accelerate, this trend.

The increase in available network resources is leading to both an increase in the number of users, and the introduction of new applications. Both of these trends increase the likelihood of congestion in networks. Increasing the number of users increases the load offered to the network, which directly causes congestion.

In addition to their appeal in drawing new users to the network, new applications also place new demands on the network. Many new applications have traffic patterns that differ from those of existing applications. This can undermine fundamental assumptions made by a congestion control system. As an example, consider real–time video in the Internet. This application requires a continuous multi–megabit per second stream of data to be delivered over an extended period. The data needs to be delivered quickly and with minimal jitter. These requirements are different from the assumptions made about source requirements by TCP[8], the most common protocol providing congestion control and avoidance in the Internet. TCP expects sources that are maximizing their throughput, and that can adjust their sending patterns. TCP is less effective in the face of this new application, which both measures network performance by a different metric, and is more sensitive to perturbation of its traffic. Such new applications, a few of which we describe below, are becoming common on the Internet. Although TCP's congestion control and avoidance mechanisms are not ideally suited to these applications, they are flourishing due to the amount of free information available in the Internet.

World Wide Web viewers, like NCSA Mosaic, [9] are an example of applications that are network intensive. Mosaic is a hypermedia browser that is used to view a distributed multimedia database residing on various sites in the Internet. Items in the database range from small ASCII files to multi–megabyte compressed full–motion video clips. The interface is such that users are not always aware that their actions can result in the transmission of large amounts of data. Even expert users are often unable to tell the capacity or state of network links between them and the data, and therefore are unable to accurately estimate the effect of their request on the network. The result is an application that can place great demands on the network without passing any information that this is occurring to the user.

Scientific research applications can also require large amounts of network bandwidth. Distributed simulations of Grand Challenge style problems can require gigabit per second network bandwidth to solve in a reasonable time[10]. These applications must be written with a knowledge of both the network and the problem being attacked, and are highly customized. Because these applications require such large amounts of bandwidth, even for short periods, they can cause a great deal of congestion. They often exhibit unusual traffic patterns as well.

These two examples give a feel for some trends in network use. Simple to use interfaces like Mosaic will bring many bandwidth hungry users into networks, which may be populated with scientific and commercial applications that also require high bandwidth. As all these sources' traffic collide in the networks, congestion is the likely result.

Congestion avoidance and control has been a fertile area for research. Chapter 2 will discuss related congestion control work in detail, but we will sketch some major approaches here to give a flavor for them and how they have influenced Dynamic Time Windows.

1.3.1. Allocation Systems

One approach to congestion avoidance and control is to require sources to tell the network what resources they require and for the network to enforce those resource allocations. We call these types of systems *allocation systems.* An allocation system consists of a protocol used by sources to communicate their resource requirements and traffic profile to the network, a method to enforce those profiles at the source or network entry point, and a method to mete out those allocations in the network.

Resource allocation protocols include RSVP[11], and RCAP[12]. These protocols allow sources to request resources in the network by including a source profile in a connection request message, which carries the profile to the network elements that have the resources. The profile is a description of the resources that the source wishes to acquire. If the resources are available, the request is granted and the source may begin sending data, otherwise, the source must make another request before sending data. A profile contains information such as average and peak sending rates of the source.

The most common form of enforcement mechanism is Leaky Bucket[13], or some variation of it[14-16]. Leaky Bucket allows a source to negotiate both a maximum burst size and a maximum sustained average rate. Packets that violate the negotiated parameters may be discarded, marked as violators, or queued to be sent later. Leaky Bucket can also be modified to police traffic entering the network, rather than traffic leaving the source[17]. Other enforcement mechanisms include Zhang's User Behavior Envelope[18], and a variety of windowing systems[19]. Although the specific mechanisms vary, all of these enforcement policies constrain source behavior to meet the parameters negotiated with the congestion avoidance and control system.

The final element in an allocation system is a mechanism in the network to provide the requested resources to traffic. The most common mechanism is a system of queueing packets at network nodes, like Weighted Fair Queueing[20], Virtual Clock[21, 22], or Stop–and–Go[23]. These queueing disciplines provide each source with a guaranteed rate of service, or a guaranteed amount of another resource. For example, Stop–and–go provides both rate and jitter bounds. All disciplines of this type constrain the way traffic is served in the network.

The combination of these three mechanisms constitutes an allocation–based congestion avoidance and control system, which we refer to as an allocation system. Such a system is often provably congestion–free at the expense of full utilization of the network. These systems generally operate without any attempt to modify source behavior based on network state, so if sources have characterized themselves conservatively, there is no way to make use of the excess capacity of the network.

Bursty traffic can be particularly disruptive to these schemes. If sources are bursty and uncorrelated, a network can support more sources by overbooking the network on the assumption that when some sources are bursting others are idle, and the average use of the network is at a congestion free level. This sharing of network resources due to source burstiness is known as stochastic multiplexing. Allocation systems either assume worst case traffic, or make static assumptions about the amount of stochastic multiplexing. In the first case, the network is underused if there is any room for stochastic multiplexing. In the second, no steps are taken to determine the validity of the assumption under real traffic. The network may be either underused or overbooked. Furthermore, traffic patterns differ with sources, so a network that is overbooked for some set of sources may be underbooked for another set of the same number of sources meeting the same source profiles, but sending different traffic.

Although most allocation systems do not attempt to adjust their behavior to current network state, it is possible to adjust parameters of allocation systems in response to network state. Most allocation systems are designed to be free of feedback as a matter of design philosophy. DTW has many characteristics of an allocation system that is tuned by feedback. As we will show, this results in many of the benefits of both feedback and allocation systems being present in it.

1.3.2. Feedback Systems

Another approach is taken by the TCP congestion avoidance mechanisms[8] and the DECBit algorithms[2, 3]. These systems use feedback to determine the current state of the network, and then utilize that information to modulate some parameter of their source control algorithms. We call systems that control source behavior based on information about the current network state *feedback systems.*

Commonly, feedback is used to adjust the maximum size of a sliding window, which controls the amount of data that a source may have outstanding. Notice that this modulates a source's throughput directly. Since systems of this type depend on feedback from the network to determine the state of the network, a high *bandwidth–delay product* limits their effectiveness. The bandwidth–delay product is the product of the bandwidth of a link and its round trip delay. It represents the maximum amount of data that the source could have sent before receiving feedback from the destination. The bandwidth–delay product is a good metric for how much the state of a network can change before a source learns about it. For a coast–to–coast OC-12 SONET link (622 Mbps) the bandwidth–delay product is 31.1 megabits, assuming a 50 ms round trip time.

For a coast–to–coast 56 kbps link (the bandwidth of the backbone of the Internet until the late 1980's) this is 2800 bits. Since feedback is used for both control and avoidance in this type of system, the time to clear congested queues depends directly on the bandwidth delay product[24]. The larger the bandwidth–delay product, the less accurate feedback from the network is about the current state of that network. Noting that from the 1980's to the present the bandwidth delay product has increased by a factor of 1000, feedback systems face a potentially insurmountable challenge.

1.4. Dynamic Time Windows

As traffic on computer networks becomes more diverse, and the bandwidth of those networks becomes higher, the allocation and feedback congestion control and avoidance mechanisms described above will prove inadequate. Wide area networks of the future will have higher bandwidth–delay products than current wide area networks, exercising the weaknesses of feedback–based control. Since the number of both network users and new applications is rising steadily, new traffic patterns are likely to appear. If Mosaic traffic is any indication, much of the new traffic will also be bursty. This influx of different traffic profiles, many of them bursty, makes allocation systems unappealing due to the potential loss of network capacity.

Dynamic Time Windows is a hybrid of some of the best aspects of allocation and feedback systems with the addition of a mechanism that directly controls source burstiness. This section describes the basics of Dynamic Time Windows, and extensions to that system to tailor network service to source preferences.

1.4.1. DTW Congestion Control and Avoidance

Dynamic Time Windows (DTW)[25, 26], is a system designed to meet the challenges of bursty traffic in a high bandwidth–delay product network. It is an integrated system of queueing disciplines, feedback, and source controls that directly addresses sources burstiness. Initial designs of the DTW system were put forth in 1992 by Landweber (who coined the term "time window"), Mukherjee and Faber. That early work laid the foundations of the system, and provided some analysis and simulation studies of the system working on a single switch. This work generalizes the system to function in networks of arbitrarily many switches, provides more simulation studies, and reports on a prototype implementation. The basic system of the earlier work is extended to allow sources to request different service from the network.

DTW is designed for use in a connection–oriented network that sends small packets. An Asynchronous Transfer Mode (ATM) network is an example of such a network. Such networks rely on switches to forward traffic from a source to a sink along a fixed route called a virtual circuit. Switches allocate resources, such as buffer space, on a per–virtual circuit basis. A more detailed description of the network model may be found in Chapter 3.

Sources are characterized by a peak rate, an average rate, and a *time window*. The time window is a direct measure of source burstiness, and is the interval over which the average rate is enforced at the source. The name is chosen for the analogy that it draws with packet windows. A packet window controls a source's throughput by its size; a time window controls a source's burstiness by its size. The larger a time window, the longer a period of time a source has to adjust its behavior to meet its average, and the burstier it can be. A smaller time window means that a source must conform more closely to its average rate, and send smoother traffic. Notice that as a time window approaches the time to send one packet and be idle long enough to maintain the average, a source's traffic becomes more like time division multiplexing at the negotiated average rate. The larger the time window, the more that source's traffic is unconstrained. The time window is the parameter that DTW modifies in response to network congestion.

The peak rate of a source is defined as the fastest rate that it is permitted to send two of its minimum sized transmission units. For a source sending ATM cells, this is the rate at which it can send two back–to–back cells. A source is always permitted to send at its peak rate. Often the peak rate will be the maximum rate of the transmission medium attached to the source. If sources negotiate a lower peak rate than the attached transmission medium, an additional mechanism must be implemented to insure that all pairs of cells sent conform to the peak rate.

The specific constraint imposed on a source, other than enforcement of peak rate, is called the *time window criterion.* A source is obeying the time window criterion if over any time window the source sends at or below its average rate. (The time window criterion is defined precisely in Chapter 3.) This leaves room for bursty behavior since periods less than the time window size are unconstrained. For example, a source with a time window of 1 second, a peak rate of 100 Mb/sec, and an average rate of 50 Mb/sec could send at 100 Mb/sec for the first half second, and then be idle until the rest of the time window had passed. For the first half second, the source exceeds its average rate, but it still meets the criterion.

Enforcing the time window criterion constrains the duration of network congestion, a fact proven in Chapter 4. Constraining the duration of network congestion is a powerful feature of DTW, absent in feedback systems and most allocation systems. The criterion is enforced by a source control algorithm, which is described in Chapter 3. This algorithm is a mechanism for congestion control. The algorithm receives feedback, but the fact that it controls congestion is independent of feedback, and therefore is effective in the face of a high bandwidth–delay product. Although the duration of network congestion depends on the sources' time windows, the more important effect of time window size is in limiting source burst sizes. Combining the source control algorithm with feedback to adjust time windows provides a system that controls congestion while making efficient use of network resources.

The bound on network congestion times also requires that switching nodes in the network use Weighted Fair Queueing (WFQ). This requirement is due to the fact that switching distorts traffic, and DTW's bounds on congestion time are based on traffic meeting the time window criterion. Using WFQ bounds the distortion of traffic in the network, and allows DTW to control congestion across many switches.

The switches provide feedback to sources. Switches control the time windows of sources by providing the sources with the maximum time window (MTW) that any sources sending traffic through them can use. A source calculates the time window it will use from the minimum of the set of MTWs that it collects from switches on the path to its destination. DTW assumes an underlying fixed route connection, so that a single path from source to sink exists while the entities communicate. A virtual circuit in an Asynchronous Transfer Mode network is an example of such a connection.

Switches determine the MTW by using algorithms similar to those used to adjust packet windows in feedback congestion control systems. In particular, the MTW is increased linearly in the absence of congestion, and is decreased multiplicatively when congestion is detected. Switches determine congestion by examining their internal state, usually their queue lengths. A specific algorithm similar to one that works for packet windows is effective for time windows, further emphasizing the analogy between packet windows and time windows. The algorithm appears to be a good balance between finding a fixed value for source time windows, and detecting changes in network state. It is described in more detail in Chapter 3, and simulation studies of its performance are presented in Chapter 5.

Feedback rescues DTW from erroneously estimating the amount of stochastic multiplexing in the network. That one source's traffic may use buffers allocated to another source while that source is idle is an example of stochastic multiplexing of buffer space. The MTW of a given switch can be seen as an estimate of how much stochastic multiplexing is currently being observed there. *I.e,* the MTW is inversely proportional to the observed multiplexing. The MTW is constantly sent to sources who base their time windows on this value, and therefore on the observed stochastic multiplexing at the bottleneck switch. If there is a lot of stochastic multiplexing, queue lengths at switches will be short, since switches will be serving only a few bursts from few sources at any given time. The time window may be increased so sources can send longer bursts if they have them. If queue lengths are building, only a little stochastic multiplexing is happening, and bursts are colliding. In this case, reducing the time window reduces the size of the colliding bursts, and therefore the level of congestion.

1.4.2. DTW Service Tailoring

Although Dynamic Time Windows is primarily a system for congestion avoidance and control, it is also effective for tailoring network performance to meet source needs. As new applications appear, they will prefer different types of service from the network. For example, the video sources discussed earlier have different expectations than real–time file transfer sources like Mosaic. The first prefers loss to delay, the second prefers a high throughput to a low per–packet delay. Unlike most congestion avoidance and control systems, DTW allows users to specify how their traffic is to be handled to suit their preferences while maintaining an uncongested network.

Service tailoring is implemented by two integrated techniques: resource allocation and selective feedback. Resource allocation involves the distribution of network resources in such a way as to change how traffic is forwarded inside the network. Selective feedback involves sending the same signals to sources that expect the same service from the network, and tailoring those signals to the sources' needs. In DTW these signals take the form of the switch MTW sizes.

The resources that DTW allows sources to request from the network are buffering and processing power. The allocations of these resources are referred to as the source's *buffer share* and *service share,* respectively. Notice that these resources are allocated in the network, *e.g.,* at switches, not at the source. Allocating a high service share to a source means that its traffic will be served at a higher rate by switching nodes in the network, and that its traffic is likely to have a shorter per–packet delay. Increasing a source's buffer share allocates more buffering to the source in the network, making that source's traffic less vulnerable to losses. In both cases these allocations are opportunistic, meaning that if some source has reserved capacity but is not using it, a source that is in need of that capacity can use it until the source that reserved the capacity claims it. For example, if source A has used all its reserved buffering at a network node while source B has not, and a packet arrives for source A, that packet may be placed in source B's buffer until source B has need of that space.

Selective feedback refers to conceptually grouping sources with similar preferences together, and tailoring their feedback to those preferences. Sources grouped together in this way are said to be in the same *feedback group.* For example, all video sources may be in the same feedback group, while all file transfer sources are in another. Feedback to the video feedback group would be tuned to keep per–packet delays low. Ideally, maintaining this low queue occupancy would not require excessive traffic smoothing by the source. File transfer sources would receive feedback designed to keep their buffer occupancy high, thus avoiding loss and keeping throughput high. The mechanisms to tailor this feedback include having switches monitor specific buffer pools for congestion, changing the increase and decrease factors in the MTW adjustment algorithms, and even routing signals generated for a feedback group that cannot adjust their time windows to another feedback group that can.

A more precise definition of the mechanisms used for service tailoring appears in Chapter 7, and simulation results showing their effectiveness are presented in Chapter 8.

1.5. Structure

The thesis is structured in the following manner. Chapter 2 is a discussion of related work in the field of congestion avoidance and control. Chapter 3 describes DTW in detail. It discusses the network model, the system components in detail, and the specific algorithms used. Chapter 4 contains analytic results, including a proof that congestion will clear periodically from all switches in the network. It also includes an analysis of the DTW source control, including a comparison to Leaky Bucket. Simulation results concerning DTW are presented in Chapter 5, including a comparison to TCP and Leaky Bucket with WFQ. Chapter 6 discusses a prototype implementation in the XUNET network. Chapter 7 describes the service tailoring algorithms in detail.

Simulation results using those algorithms are presented in Chapter 8. Chapter 9 draws conclusions and discusses possible directions for future work.

Chapter 2

Related Work

''What's past is prologue.''

— *William Shakespeare,* **The Tempest, II, i, 261**

The topic of congestion control and avoidance has been studied for many years. When networks were smaller and had lower bandwidth links, the problem was essentially one of flow control. The problem of flow control involved limiting a source's sending rate so that it did not fill the buffers at the destination faster than they could be emptied. As networks became larger, faster and more heavily used, issues of cross traffic and the effects of queueing disciplines at switching nodes became more important. Some people have gone so far as to say that the technology of the day, cheap buffer memories, has solved the congestion control problem, since we can buffer as much traffic as we could ever need to deal with[27]. Sadly, this is not so, since delaying traffic by putting it in excessively large queues is in many cases as bad as losing it. Jain investigates this and other myths about congestion control[28].

We describe the previous work in three subareas: feedback systems, allocation systems, and others. Feedback systems rely on information from the network, directly or indirectly, to adjust source behavior to meet changing network conditions. Allocation systems are static systems that depend primarily on mechanisms to reserve resources in the network and maintain those safe reservations. Of course there is some overlap between the two types, but in general, a system can be characterized as one or the other.

However, there are some approaches that fall outside either category, and we will also describe these.

2.1. Feedback Systems

Feedback systems can further be classified into systems that control source throughput directly, called rate based control, or indirectly by controlling the size of a sliding window, called window–based control. When using the latter, a sliding window is used to set a limit on the number of packets a source can have outstanding over the period of a round trip time. When one of the packets sent is acknowledged by the receiver, the sender has a free space in its window and can send a new packet. This control works best when the source can always send; that is the acknowledgement for the first packet sent arrives just as the last packet allowed by the window is being sent[29].

2.1.1. Rate–Based Systems

Bharath-Kumar and Jaffe provide an extensive analysis of the two virtual circuit case of a rate control system[30]. The system is designed to maximize power, which is source throughput divided by the packet delay raised to a given power. They develop three algorithms. The first is a simple greedy maximization of power by each sender in turn. The second is a modification of that algorithm to take into account the effect that one sender has on the other. The third uses effective capacities of the lines assuming that all sources share them equally. In all cases, the algorithms assume global knowledge of the network. Jaffe later shows that this performance metric is nondecentralizable, meaning that this metric can be maximized for the entire network only by the use of a centralized server[31]. In other words, a group of sources avoiding congestion cannot optimize the usage of the enitre network without global knowledge. He also describes a system that uses a centralized system that maximizes power[4]. The decentralizability issue is taken up by Selgar, who proposes another definition of power that is decentralizable[32], and puts forth a class of algorithms to maximize it[33]. The above do not probe the network state to determine the state of the network. The only change in state they detect is the addition or deletion of a source. They set the rate parameters based on calculations of the number of sources and their requests for resources, all of which are re–evaluated whenever a new source enters or leaves the network.

Other systems probe the network for information. Matsumoto describes a system where virtual circuits have their throughputs reduced and then cut off altogether based on queue lengths at the switches[34]. Haas uses back to back sampling packets to sense network load, and adjust sending rate by adjusting the minimum interpacket gap[35, 36]. Bolot and Shankar have also investigated rate based flow control in detail[24, 37]. They have treated network traffic as a fluid and investigated the properties of feedback–based rate controls mathematically. Their work provides many clear intuitions into the dynamics of such systems.

2.1.2. Window–based Systems

As mentioned above, one can set a rate indirectly by setting a window size. There have been many analyses of how window–based flow control affects networks. Harrison describes and analyzes a two mode queueing model designed to force the queue length at a switch to zero after congestion occurs. The system seems effective in controlling congestion, when latencies are low. His work is presented only for one switch[38]. Morgan gives a concise and intuitive description of the problem of determining optimal window size on a trunked byte stream[39]. He is able to derive a window size that is optimal for throughput, but the system is centralized and does not probe the network state. Kleinrock and Kermani use markovian analysis to model a single source sending to a single destination via window based flow control, in both the case that the window is static, and that it is changed with network state[40, 41]. The results provide a thorough understanding of the simple case that they model. Mitra and Seery perform product–form queueing analysis of networks of queues, where the sources are controlled by windows controlled by feedback[42-44]. They provide both analyses and simulations of their work under various configurations. Multiple users over multiple paths are simulated, and their findings corroborate their analyses. They can find optimal window sizes to maximize throughput, but their method is less effective in a high–latency environment. The simplifying assumption they make is that they only model the bottleneck switch, modelling the others as delays. Fernow and El-Sayed analyze a feedback system based on opening and closing windows depending on how much actual input rates to a switch vary from a predicted value. They claim that the resulting packet flow rate will have a deterministic component and a stochastic component that depends on the error in measurement, the rate of adjustment, and the magnitude of adjustment[45, 46]. Further analysis of windowing with feedback has been a common topic, with the common conclusion that it is possible to maximize throughput for small values of the bandwidth–delay product[47-52]. In general, these have been analytic studies where sources are modelled as sending at constant rates. The fact that computer traffic is inherently bursty is not explored in any of the above.

Jain and Ramakrishnan have done extensive work on feedback based sliding window congestion avoidance. They published several papers describing congestion control in a packet switching environment with feedback, both with and without assistance from switches[1-3]. The work presents many useful concepts, such as the importance of filtering sampled congestion data and the advantages gained by directly involving switches in congestion determination. They also confirm Jaffe's earlier observation that without sharing information, it is impossible for sources to converge to an optimal usage of resources for the network. Jain and Chiu also use simulation to investigate several methods of adjusting window size[53]. They are able to control congestion effectively using existing protocols and routers. However, their feedback method does not scale particularly well to high speed networks, due to the familiar problem with a large bandwidth delay product.

Another interesting packet based feedback system is the TCP/IP system in the Internet[54]. Of primary interest are the congestion control mechanisms added to the BSD UNIX kernel by Van Jacobson[8]. The mechanisms detect congestion in the network by detecting packet loss either via an out of order acknowledgement or a timeout mechanism. The timeout mechanism depends on being able to estimate the round trip time for a given connection, and Jacobson presents a moving average method to do so. When congestion is detected, the window is closed by multiplying its size by a fraction, and when no congestion has been detected for a full window worth of sending, the window is increased by one packet. The system also incorporates a *slow start* mechanism, which reduces the window to one packet when congestion is present and then allows it to open quickly to a given threshold. This allows the network to recover from the detected congestion before the source begins sending at its full rate again. Slow start should be considered a congestion control algorithm, while the adjustment of the window size is a congestion avoidance method. Since the system is deducing the state of the network, it is required to activate congestion control whenever it believes the network state has changed for the worse. Studies have revealed this algorithm to have both desirable and undesirable properties[55]. Packets tend to clump with packets from the same source remaining together in the switch queues, and when the network becomes congested, all sources lose packets. Furthermore, due to the FIFO service disciplines of the switching nodes in the Internet, and TCP's use of acknowledgement packets as an implicit clock on senders' rates, the system can exhibit instabilities and packet loss due to acknowledgement compression[56, 57].

Wang and Crowcroft propose a variant of the Jacobson congestion controls which seeks to more explicitly find the knee of the throughput/load curve by following the normalized throughput gradient[58]. The normalized throughput gradient is essentially the rate of change of throughput with respect to load. They also seek to explicitly synchronize changes in source window sizes rather than relying on the implicit synchronization losses induce.

Keshav has taken a control theoretic approach to congestion control[59]. He derives a control law for source sending rates based on the rate of the bottleneck server on a given path. He sends packets back to back to probe the service rate at the bottleneck server by measuring the difference in arrival times of their acknowledgements. He requires a fair queueing discipline at all switches in the network to ensure that this is a reasonable estimate. The system works well, but is susceptible to the limitations of feedback systems in high–bandwidth delay product networks[59, 60].

2.2. Allocation Systems

Allocation systems are systems that control congestion by reserving resources for sources based on descriptions of their behavior. If sources maintain the negotiated behavior, these systems guarantee that congestion does not occur. The distinguishing feature of allocation systems is that there is no attempt to sense the state of the network and adjust system parameters accordingly.

A common aspect of these systems is a specification of the queueing disciplines used at intermediate switches. Keshav and Hui Zhang provide an overview of many of these disciplines[61].

Among the most interesting queueing disciplines are Virtual Clock[21, 22] and Fair Queueing (including Weighted Fair Queueing)[20]. These both approximate a weighted bitwise round robin service to packets, and Keshav and Hui Zhang claim that the two are equivalent[61]. Both provide a way of fairly allocating switch processor time to multiple sources, in a way that becomes more fair as the packets get smaller. These disciplines approximate the processor sharing discipline from queueing theory at switches in the network. Each packet can be seen as a job, and the packet size as the scheduling quantum. Ideally these queueing disciplines allocate the switch service rate perfectly, like processor sharing, but in reality they allocate the switch service rate more like a round robin queue. Virtual Clock allocates this fairness based on the average sending rate of sources using the switches, while Fair Queueing does not require any particular weighting scheme to assign priority. Both also insulate well behaved users from misbehaving ones by guaranteeing each source the agreed upon allocation of switch processor time. Morgan independently verifies the firewalling capabilities of these disciplines[62]. Parekh and Gallagher analyze them extensively, proving many useful properties[63-65]. Virtual Clock is incorporated in Lixia Zhang's flow network, which uses a *user behavior envelope* based on sources' negotiated parameters to smooth traffic[18]. The user behavior envelope allows the source to send only its average rate times its averaging interval bits in any time period of the length of its averaging interval. This is similar to DTW's packet admission, except that DTW's packet admission system allows the average interval to change as well. Fair Queueing has been simulated and employed in several systems, including Keshav's Packet Pair system[59], and a study of Fair Queueing and TCP[66].

Golestani describes a queueing policy designed to maintain guaranteed jitter, delay, and loss bounds. The policy and framing discipline is referred to as Stop–and–Go queueing[23, 67-70]. Under Stop–and–Go, time is divided into frames on both incoming and outgoing links. Packets arriving in an incoming frame are sent in the next non-overlapping outgoing frame. Incoming and outgoing frames need not be synchronized, and in general are not. Based on the fixed frame sizes, this provides the delay and jitter bounds. Controls on the source sending rates ensure that the switches can always fit all the traffic from one incoming frame in one outgoing frame. Providing multiple frame sizes allows sources with different requirements to share links. There are questions regarding the tightness of the jitter bounds[61] and one must note that the discipline is not work conserving, which makes certain analyses difficult.

Traffic admission control and resource reservation, with or without sophisticated queueing, constitute another component of allocation systems. The two are closely related, since resource reservation algorithms often use the traffic control parameters to decide what resources to reserve, and traffic controls often are used to label packets for which there are no resources.

One of the earliest congestion control strategies proposed was an isarithmic control scheme, where the number of packets in the network as a whole was limited. In order to send a packet, a source had to acquire a credit from the network, which was reintroduced when the packet left the network. This system still allowed local points of congestion to occur[71]. Giessler, Jagemann, Maser and Hanle proposed controlling congestion by setting aside a number of buffers at each switch labeled by the number of hops a packet had taken. If a packet arrived at a switch and there were no buffers available labeled with the number of hops it had taken to arrive at this switch, it was discarded[72]. This scheme was originally put forward to prevent deadlock in networks, but is also proposed as a congestion control method. Kamoun proposes a similar scheme, but only partitions the buffers into a set for new packets and a set for packets that have been forwarded once. By controlling the number of buffers set aside for new packets, the number of new packets entering the network as a whole can be constrained[73].

A popular method of source traffic control is the *leaky bucket,* proposed by Turner[13]. Each source is permitted to send one packet (or number of bits) for each credit it has in a pool. There is a maximum number of credits, and when credits are used, they are replaced at a fixed rate. Packets to be sent when there are not enough credits are either buffered until there are enough, or dropped at the source. This allows a source to send at a fixed rate, while allowing a measure of burstiness. Sohraby, *et. al.* have investigated the algorithm thoroughly, including using it to mark packets that are sent faster than the source has negotiated[14-16]. Once these packets are marked, the switches can use this information in times of congestion, for example, discarding marked packets. The papers discuss and simulate this and other alternatives.

Eckberg, Luan, and Lucatoni use leaky buckets in their Bandwidth Management (BWM) congestion control strategy[74-76]. This choice is the result of the investigation of several policing policies[19]. The theme of BWM is that congestion control should be be simple and robust, since many sources in their network environment will be very simple, or unable to devote significant resources to congestion control processing. Sources are allowed to send packets into the network based on their negotiated parameters, and data sent in excess of those parameters is marked. If the network becomes congested, marked packets are discarded first.

Ramamurthy and Dighe propose a network access control called Distributed Source Control (DSC)[77, 78]. It is based on the idea that each source should smooth its traffic over fixed units of time called smoothing intervals. A source is allowed to send a window of data every non–overlapping smoothing interval. The windows are derived from the end–to–end throughput that the source desires, its projected round trip time, and the reservations currently in the network. They study the effect of the size of the smoothing interval on the number of users the network can support, and find that smaller windows allow more users to be supported. This is unsurprising since smaller smoothing intervals mean smoother traffic, and smoother traffic is more predictable and easier to serve. They also mention an end–to–end flow control system may exist as a back up that would adjust sources window size in case of congestion. The feedback does not directly affect the DSC system.

2.3. Other Work

There have been several other approaches taken to congestion control and avoidance and we mention these here both for completeness, and to demonstrate the diversity of the field.

There has been a significant amount of work devoted to fighting congestion by dynamically rerouting virtual circuits to remove congestion[79-81]. Although this is an interesting approach to the congestion control problem, it is not clear how one can reroute a call while data is being sent. It is also unclear how effective call rerouting would be in a system without calls. Once those problems are solved there are all the issues raised in the cited work to deal with, as well as what to do when the traffic simply cannot be rerouted.

Others have approached the congestion control as a problem in Game Theory. Jain suggests this may be fruitful due to the apparent conflict between a selfish optimum sending rate and a network optimal sending rate[2]. Sanders investigates this area, and describes a pricing system whereby rational users will adjust their sending rates based on price incentives[5, 82]. The incentives are calculated by studying the partial derivatives of sending rate and throughput, and the maximum obtained by gradient hill climbing Since the system is based on pricing incentives, it is unclear how it would work in today's networks where many users either do not pay for service or are ignorant about how their sending habits influence their costs. Also, the fact that gradient hill climbing is used means that the system is subject to finding false maxima.

Williamson and Cheriton propose having the network provide sources with a description of how likely the network is to drop packets based on a source's sending rate[83, 84]. What results is essentially a pricing system where the costs are given in terms of the loss performance of the network. Although this is an interesting idea, it also suffers from the same problems as Sanders' work. Not every user will be able to decide how to best take advantage of the network's pricing scheme.

2.4. DTW Work

This thesis is not the first published work on Dynamic Time Windows. Early work was done on the system by Mukherjee, Landweber and Faber. They have presented work on the Pulse queueing discipline, which initially used in DTW, and outline the basic concepts of time windows[25]. The DTW stability theorem, stated and proven in Chapter 3, was also previously published, along with simulation studies of the DTW feedback system used through one switch[26]. The thesis builds on the lessons learned in that work by extending the queueing discipline to be more general, and adapting the feedback system to more complex networks. It also provides new analytic results, more complex and detailed simulation results, and experimental evidence from a prototype implementation.

Chapter 3

Dynamic Time Windows

''Time travels in divers paces with divers persons. I'll tell you who Time ambles withal, who Time trots withal, who Time gallops withal, and who he stands still withal.''

— *William Shakespeare,* **As You Like It, III, ii, 286**

''These windows, sir, are the most perverse creatures in the world.''

— *Joseph Addison,* **The Spectator, no. 335, March 25, 1712**

This chapter describes the DTW network model and algorithms. We begin with a description of the network model, and follow with a high level overview of DTW. We conclude with descriptions of the algorithms used by DTW at each network element.

3.1. The Network Model

The target environment for DTW is a high speed, wide area, Asynchronous Transfer Mode (ATM) network. We take high speed to mean a network with transmission speeds on the order of 1 Gigabit/sec, with hosts separated by at least several hundred miles. Such a network has a bandwidth–delay product on the order of 30 megabits. The unit of transmission on the network is an ATM cell, which is a 53 byte packet containing 48 bytes of user data. The network model is connection–oriented, requiring a virtual circuit to be established from source to sink before data can be transmitted.

There are three elements in the network: sources, sinks and switches. A source is the transmitting endpoint of a virtual circuit, and a sink is the receiving endpoint of a virtual circuit. These definitions specifically leave the physical realizations of sources and sinks vague. They may be computers attached to the network, applications running on computers, gateways from local area networks to wide are networks running DTW, or even other objects like cameras or microphones. A source is required to have sufficient intelligence to implement a connection establishment protocol, the ability to send data, the ability to receive feedback, and the ability to regulate its transmission.

Switches are entities that multiplex and demultiplex cells from one of several incoming transmission lines to one of several outgoing lines. The rate that a switch can process cells is called the switch's *service rate* and is denoted by μ. Switches have significant buffering, and sufficient processing power to implement complex queueing disciplines. Switches must have control over their buffering policies and be able to monitor their internal state for signs of congestion. They must also be able to send cells into the network. An example of an intelligent switch is the XUNET switch[6], which is described briefly in Chapter 6.

The network is connection–oriented and uses virtual circuits. A virtual circuit is a route through the network from source to sink through one or more switches. All traffic associated with a given virtual circuit follows the same route. Resources to be used by traffic associated with the virtual circuit are allocated at each switch along the route.

When a source and sink wish to communicate, they first establish a virtual circuit by negotiation with the switches along the path connecting them. Virtual circuit establishment reserves resources at the relevant switches, and assigns the communication a identifier, called a virtual circuit identifier (VCI), which is used to route cells. The connection may have a different VCI at each switch, but the source, the sink and each switch on the path can uniquely identify the connection. DTW associates traffic with a source by VCI. All virtual circuits are one–way, so a two–way communication requires two virtual circuits. Two communicating processes may be modelled as two sources and two sinks each source–sink pair communicating over a separate virtual circuit. See Figure 3.1.

3.2. DTW Overview

The goal of DTW is to provide congestion control and avoidance in this network. Since DTW is designed to serve bursty traffic, we must define burstiness. Intuitively, sources that alternate transmission of data at their peak speed with idle periods, rather than sending data continuously at their average rate, are bursty. We quantify this by

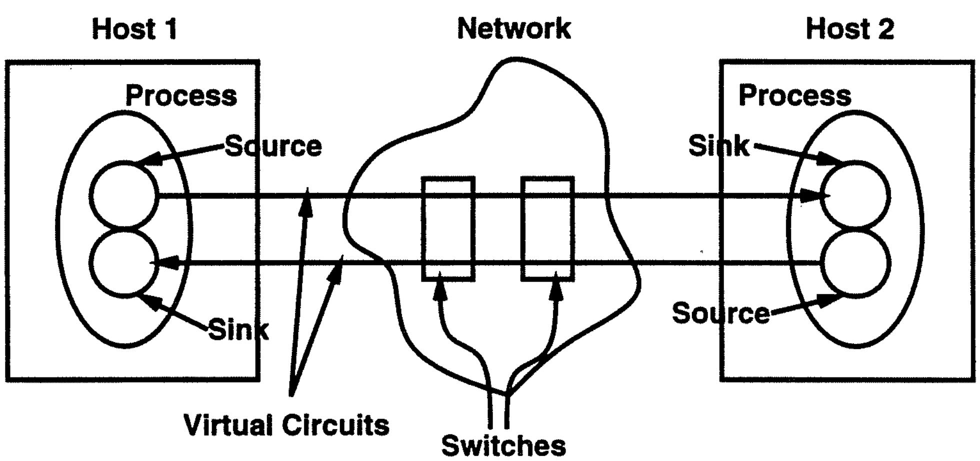

Figure 3.1 : Sources and Sinks in a Process

saying that a source with a significant difference between its peak and average rates over the life of its connection is bursty. A reasonable measure of this burstiness is the ratio of peak to average rate. When two sources share the same peak to average ratio, their relative burstiness is determined by the average size of their bursts. This distinction makes the intuition explicit that a source sending long bursts followed by long idle times is burstier than one that sends closely spaced short bursts. When comparing two source control algorithms, the one that allows a source to send larger bursts is more tolerant of bursty traffic.

DTW is a combination of algorithms that run at the sources and switches. The algorithms are based on the idea that source burstiness can be controlled directly to avoid and control congestion. The abstraction of source burstiness that DTW manipulates is the *time window,* described in Chapter 1, and defined below.

3.2.1. Congestion Control Using Source Burstiness

A source is defined by three parameters: its peak rate ($\hat{\lambda}$), its average rate ($\bar{\lambda}$), and its time window (I). The peak rate is the fastest the source will send, measured over the transmission of two cells. The average rate is the rate the source intends to sustain over several seconds. The time window is the current period over which the average rate is being enforced by the source. The time window is the measure of source burstiness modulated by DTW.

Throughout this work, we describe the source as controlling its traffic based on the time window. Other placements of the source control may be effective as well. Traffic may be policed entering the network at the first switch, or the algorithms may be implemented in smart network interface hardware. There is nothing in the specification of the source control algorithms that prevents them from being implemented in these other places. However, there are benefits to having the traffic control close to the generating process so that the process can adjust itself to the control. An example of this is a file transfer source that suspends itself if the source control algorithm has as much traffic queued as possible. The generating process stops because there are no buffers available at the source, as opposed to in the network. Trying to achieve timely communication with a regulator in the network could be as hard as controlling congestion.

Pedagogically, describing the algorithms as implemented at the source partitions the source control and feedback subsystems well. This enables a clearer understanding of the two systems and how they interact. This convention will be followed throughout this work.

The average rate is enforced by forcing traffic to obey the time window criterion, which states that *over any time period of length I, the source will send at most $\bar{\lambda}I$ bits.* The time window criterion can be taken to be the definition of a time window. If $\lambda(t)$ is the instantaneous sending rate of the source, the time window criterion can be written as

$$\forall t, \int_{t}^{t+I} \lambda(\tau) d\tau \leq \bar{\lambda} I \tag{3.1}$$

Time windows are not anchored in any sense. A source's time window does not start at a particular time. The criterion must be met for any continuous period of I time units.

Since $\hat{\lambda}$ is the source's peak rate, $0 \leq \lambda(t) \leq \hat{\lambda}$. In most cases $\lambda(t)$ will be two–valued, being either 0 or $\hat{\lambda}$, but the formulation in Equation (3.1) allows more general sending patterns. In reality, the sending rate function will be square waves whose amplitude is $\hat{\lambda}$ and $\lambda(t) = \hat{\lambda}$ for an integral number of cell transmission times.

When a source is obeying the time window criterion, the largest continuous burst it can send at a rate greater than $\bar{\lambda}$ is $\bar{\lambda}I$ bits, which depends directly on I. By definition, the time window determines the maximum burst size that a source is allowed. The time window directly controls a source's burstiness in the same way a packet window directly controls a source's throughput. A source's time window is the parameter that sources adjust, in response to feedback generated by switches, to avoid congestion while fully utilizing the network.

Under DTW, we can bound the the time that the switch will have traffic queued. The time a switch has traffic queued is referred to as a *busy period.* It will be shown in Chapter 4 that if multiple sources send traffic that meets the time window criterion through a switch, there exists an upper bound on the duration of the busy period of the switch, given two requirements.

First, the sum of the average rates of sources using a switch must be less than the switch service rate. For a switch that serves traffic from other switches, the average rates of the sources using the switch must be modified in a manner described in Chapter 4. This modified sum must be less than the service rate of the switch.

The switches must use Weighted Fair Queueing for the bound to hold at switches serving traffic that has passed through another switch. The use of WFQ, in conjunction with the enforcement of the time window criterion, allows us to bound the changes in traffic patterns inside the network. Without the use of WFQ or similar discipline, the distortion of traffic caused by queueing cannot be bounded, and the test above cannot be used.

The maximum length of the switch's busy period is the least common integer multiple of the time windows of the sources feeding into the switch. The value for this bound is derived by construction. Recall that each switch maintains an MTW, which is the maximum time window that sources passing traffic through that switch can use. Each switch MTW is the result of an integer division of a system constant, the *absolute maximum time window* (AMTW). Source time windows are determined by the integer division of the minimum of the MTWs from switches along the connection's path. Each source picks the integer (called a source's time window ratio and denoted by $r \geq 1$) it will use to calculate its time window at the time a connection is established. A source can alter its time window ratio at any time without upsetting the system, but in practice r is fixed. The time window ratio can be thought of as a source's estimate of its burstiness relative to other sources. In the absence of a pricing scheme, most sources will choose $r = 1$. Since the commodity that DTW is modulating and maximizing is burstiness, sources will be charged for using a lower r value. This will force users to make an accurate estimate of their burstiness requirements.

By construction, every source's time window is an integral divisor of the AMTW, so congestion times in the system are bounded by the AMTW. The property that switch congestion times are bounded is called *DTW stability*. DTW stability implies that if all sources obey the time window criterion, congestion will be controlled in the network. In this context, ''controlled'' has its more precise meaning of restoring a congested network to a stable state as defined in Chapter 1. The periodic emptying of queues controls congestion.

DTW stability is the key property of DTW that allows it to operate effectively in a high bandwidth–delay product network. It partitions congestion control and avoidance. As long as time windows share a least common integer multiple, congestion will be controlled because of the constraints of the time window criterion. This does not mean that the network will always be loss free; it means that the network will always return to a stable state. The time that congestion is permitted to remain is the AMTW, which can be set by network administrators. The sizes of the time windows, which are adjusted by feedback from switches, control the utilization of the network and the chance of losses. Feedback is used by DTW to avoid congestion, rather than controlling it. The high bandwidth–delay product can affect the efficiency of network usage, but not the fact that congestion will periodically disappear.

3.2.2. Congestion Avoidance Using Switch Feedback

To avoid congestion, switches communicate the amount of burstiness they can tolerate to the sources via feedback cells. The information in these feedback cells is the switch's maximum time window (MTW).

Switches choose a new MTW by observing their state for a period of time, and then increasing or decreasing their current MTW. The period over which a switch observes its state is called a *monitoring interval,* and is an integral number of MTWs. The decision to increase or decrease the MTW is based on whether the switch considers itself to be congested over the monitoring interval. A congested switch decreases its MTW; an uncongested switch increases its MTW.

This algorithm is implemented for its simplicity. Many changes could be made to the algorithm including damping oscillations in when the network is perceived to be in a stable state[26]. However, implementing this algorithm allowed us to experiment with an algorithm that is a direct analog to a packet window adjustment algorithm. Since the use of burstiness control to avoid and control congestion is new, avoiding unnecessary complexity in the manipulation of time windows makes the use of burstiness control easier to evaluate.

Under DTW, a switch is congested if its queue length is greater than a fixed threshold. Switches use Weighted Fair Queueing, which maintains a separate queue for each virtual circuit, and the sum of the individual queues determines congestion. Chapter 7 describes other criteria for switch congestion that are used for service tailoring.

When a switch increases its time window, it does so almost linearly. A linear increase would be to add a constant value to its current MTW value. Because all switch MTWs must be integer divisors of the AMTW, the switch finds the number that is an integer divisor of the AMTW that is closest to the MTW that would be obtained by a strict linear increase. The new MTW is the AMTW divided by $\left\lfloor \frac{\text{AMTW}}{\text{old MTW+increase}} \right\rfloor$. When a switch decreases its time window, it does so multiplicatively. It divides the current MTW by a constant factor. When the factor is an integer, the multiplicative decrease is exact, otherwise an approximation similar to the one used for the linear increase is used. Time windows are decreased as soon as congestion is detected, and increased at the end of the congestion–free monitoring interval. Once a switch adjusts its MTW, it refrains from doing so again until the end of its first busy period after one round trip time to the boundary of the network. This ensures that any congestion observed is caused by sources operating at the new MTW value.

Using these algorithms results in switch MTWs that increase until congestion is detected and then are reduced, and repeat the cycle. This reflects DTW's algorithms seeking the highest amount of stochastic multiplexing that the network will support. In an unchanging network, source time windows will oscillate regularly about an optimal value. In a changing network, the probing is constantly pushing upward on the current burstiness limit, looking for additional capacity. The oscillations of a source's time window are analogous to the oscillations of a source's packet window under TCP or DECBit, which are caused by those systems seeking the maximum throughput the network will support.

There are some important differences between DTW and packet window systems. The smart switches enable DTW to get negative feedback before congestion losses occur. (Some packet window systems share this attribute, *e.g.*, DECBit[3]). Simulations have shown that even if a switch considers itself uncongested until 90% of its buffers are in use, losses are rare. More importantly, since the feedback in DTW is not responsible for clearing congestion in the switches once it occurs, the size of the oscillations does not affect the duration of congestion. Under windowing feedback systems, congestion duration is directly related to the size of window oscillations, which depends on the bandwidth–delay product of the network[24].

In summary, DTW is the combination of source control algorithms enforcing the time window criterion and switch algorithms using feedback to adjust the sources' time windows to fully utilize the network. DTW is unique in the extent to which it decouples congestion control and avoidance. The mechanisms used to avoid each are more distinct than in most prior systems. Allocation style congestion control meshes well with feedback style congestion avoidance to form a unified system. The remainder of this chapter will describe the algorithms used by sources and switches.

3.3. Algorithms at Network Elements

This section describes some specific algorithms used in DTW. We discuss the mechanisms used by switches to allocate service rate and buffering, the information that must be exchanged between sources and switches to establish a virtual circuit. Finally, we describe the implementation of the algorithm for enforcing the time window criterion at sources. The use of WFQ is integral to DTW stability, as shown in Chapter 4.

3.3.1. Resource Allocation at Switches

Switches allocate resources to ensure fairness among the sources using them, and as the basis for providing service tailoring, which will be discussed in Chapter 7. They use Weighted Fair Queueing[20] to allocate their service rates among sources, and a custom buffer allocation system to allocate buffering to sources. We describe WFQ first, and then explain the buffer allocation.

3.3.1.1. Allocation of Switch Service Rate

We model a switch in the DTW system as a Weighted Fair Queueing server with deterministic service times. The service times are deterministic because the switch spends a fixed time processing each cell.

Fair Queueing[20] is a queueing discipline that allocates a switch's service rate equally among the virtual circuits using the switch. Let the switch have rate μ, and let it have n circuits established through it. In the worst case, each virtual circuit is served at the same rate as if it were the only circuit being served by a switch with a service rate of μ/n. In other words, each circuit is guaranteed a service rate of at least μ/n.

Fair Queueing is implemented in switches by using a timestamping algorithm to prioritize packets. When a packet arrives at such a switch, the switch calculates the departure time of the last bit of the packet if it were served at the circuit's guaranteed minimum rate. The calculated departure time also takes into account the delay of any packets from the same virtual circuit that are queued awaiting service at the switch. This departure time is stamped on the packet. At any time, the packet with the lowest timestamp is the next packet sent.

and DTW can be modified to use the more exact bounds.

Because WFQ bounds the rate at which cells from a given virtual circuit are served at a switch in terms of s_i, we can use the discipline to allocate a switch's service rate.

Although the assignment of a service share determines the worst case service rate that traffic on a queue receives, it does not determine the instantaneous rate. The rate a circuit sees is time dependent. The lower bound in Equation (3.2) is realized when all circuits passing through the switch have cells queued. Since the sum of the service shares is always at most one, the lowest instantaneous rate at which source i can be served is $s_i\mu$. The upper bound is reached when only queue$_i$ has cells queued, and that bound is the switch rate, μ. WFQ is opportunistic in that it allows one to bound the minimum service rate of a circuit, given the number of circuits passing through a switch and their service shares, while allowing circuits to take advantage of excess switching capacity. It predicts a worst case without enforcing it.

The behavior of WFQ only preserves the bounds given in Equation (3.2) if the virtual clocks are synchronized periodically. Consider the case where virtual clocks are never synchronized. A source can have its traffic served at an arbitrary priority by remaining idle and letting its clock lag behind those of the the other sources using the switch, and then releasing a burst. Since the idling source's clock is far behind the others, all the cells in its burst will have timestamps much lower than those of other sources, and these cells will receive absolute priority. Considering the speeds at which busy sources will have their virtual clocks incremented, this idle period need not be very long to achieve the desired effect.

Zhang addresses this problem by resetting the virtual clocks with every cell that arrives[21]. This maintains the properties of Virtual Clock, but is restrictive of bursty traffic. Since DTW is designed to serve bursty sources, it cannot include a queueing discipline that punishes traffic for exhibiting burstiness. The approach we take is simple and is facilitated by an integration between the queueing discipline at the switch and the source control algorithm. When the first cell arrives at an empty switch, the virtual clock for that cell's queue is set to real time. Whenever a cell arrives on an empty queue while the switch is serving cells, that queue's virtual clock is set to the value of the smallest virtual clock. Thus any newly filled circuit begins synchronized with the others, and the service rates of the circuits are determined by Equation (3.2). The source control algorithm ensures that switches in the network will regularly be empty, and thus their queues' clocks will be regularly resynchronized.

3.3.1.2. Allocation of Switch Buffering

Switches allocate buffers to each circuit as well. In order to accommodate burstiness as seamlessly as possible, buffer allocations at the switch are opportunistic in the same sense as service shares.

The goals of the buffer allocation strategy are to allow maximum flexibility in buffering bursts, and to guarantee each source a minimum amount of buffering when the switch is busy. Let b_i be a buffer share for source i, analogous to the service share (s_i) described above. Like service shares, buffer shares are negotiated between a source and switches when a connection is established. Like service shares, $0 \le b_i \le 1$ and $\sum_j b_j \le 1$ at any switch. If a switch has B buffer space available, then source i will have $\frac{b_i}{\sum_j b_j} B$ buffer space assigned to it, where b_j is the buffer share of the j^{th} source. Since $\sum_j b_j \le 1$, the amount of buffering that source i will receive in the worst case is $b_i B$.

One implementation of the buffering strategy is described below. The per-virtual circuit cell queues are linked lists of cell buffers, as in the XUNET switch[6]. Each virtual circuit has a header containing the current allocation of buffers for that circuit, the current number of buffers in use, and a pointer to the first cell in the virtual circuit's queue. If the circuit has more cells queued than it has buffers reserved, the header also contains a pointer to the first cell queued in ''borrowed'' buffers. The header table is stored in a table indexed by virtual circuit identifier. A list of pointers to headers whose queues contain more cells than are allocated for them, called the overbooked list, is also maintained. A pointer to each circuit's entry on the overbooked list (if any) is also kept in that circuit's header. See Figure 3.2.

When a cell arrives and there are free cell buffers at the switch, the cell is enqueued on the proper virtual circuit's queue, and the queue length is incremented. If the new queue length is less than that circuit's buffer allocation, or the old queue length was greater than that allocation, nothing else happens. If the queue length has just exceeded the allocation for that circuit, a pointer to its header is placed on the list of overbooked queues, that entry's overbooked list address is saved in the circuit's header, and the pointer to the cells using unreserved buffers is set to the newly arrived cell's buffer.

When cells are served, the reverse process occurs. The first cell is removed from the queue and served and the queue length is decremented. If the queue was overbooked and remains overbooked, the pointer to the first overbooked cell is set to the successor of the cell it points to now. If the removal of this cell brings the circuit's queue length to within its allocation, its header is removed from the overbooked list and the pointer to the first overbooked cell and the pointer to the entry on the overbooked list are nulled.

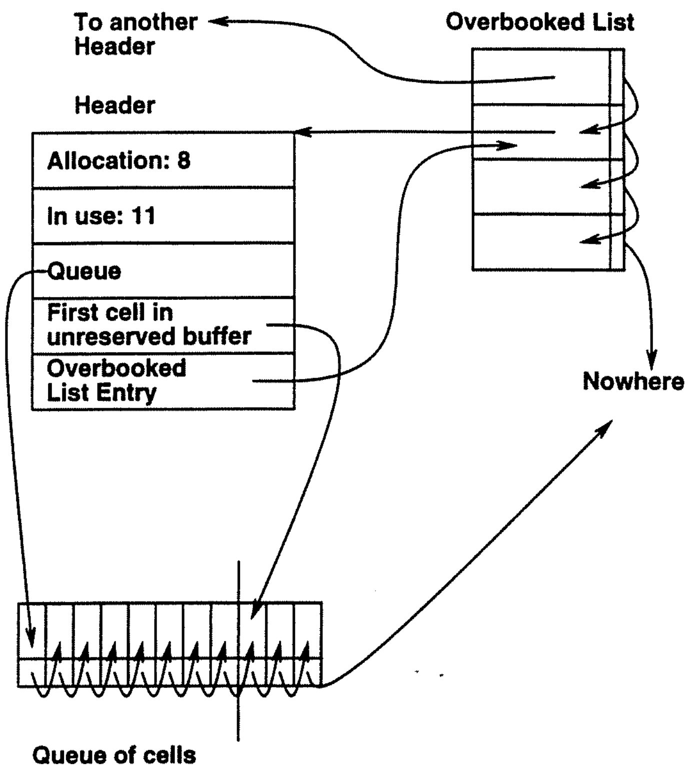

Figure 3.2 : Custom Buffering Data Structures

When a cell arrives at the switch, and its circuit still has buffers allocated to it (*i.e.,* adding the new cell to its queue will not overbook the circuit), but all buffers are in use, the first queue is taken from the overbooked list, its first overbooked cell buffer is discarded, and its first overbooked cell pointer adjusted, and that buffer is used to store the incoming cell on the appropriate virtual circuit's queue. If the overbooked queue is still overbooked, it is put at the end on the overbooked list.

Under this system, buffers are used opportunistically when possible, but under conditions of heavy load each source receives buffering proportional to its buffer share. This allows for a graceful degradation under high load.

3.3.2. Switches and Virtual Circuit Establishment

Switches also play a role in virtual circuit establishment. Virtual circuit establishment is the establishment of a virtual circuit from source to sink. The criterion for admitting circuits to a DTW network is that the new circuit does not violate the constraints at each switch that guarantee that congestion times remain bounded. The circuit establishment method given here guarantees that the network always remains in a stable configuration.

Protocols have been specified for communicating information to switches for call establishment[11, 12], and we assume that one of these protocols is used to transfer the information from a source to switches along the path and to a sink. We will describe the information communicated from source to switch, and how switches use that information to decide whether or not to establish the call. A request travels from the source to each switch on the path between source and sink in turn, then to the sink and back again.

In the forward direction each switch decides if it can accept the new call, and if so, sends the message on to the next. If not, the message is immediately started back to the source with a rejection. A switch will make a tentative allocation of resources when the message passes in the forward direction and either make the reservation real or abort it when either the confirmation or the rejection is received. If a call is admitted, the virtual circuit exists and can be used, otherwise, the source is unable to communicate with the sink. Routing the establishment request is a function of the call establishment protocol and not discussed in this work.

Other than routing information, a call establishment request contains the source's requested average rate, its effective average rate, and its requested service and buffer shares. The effective average rate of the source is the highest average rate that an observer at this switch could record. It is determined by an algorithm presented in Chapter 4. Traffic passing through a switch is distorted by the process of queueing, and traffic arriving at subsequent switches may exhibit an apparent average rate higher than that requested by the source.

When a switch receives a request, it compares the effective average rate against its remaining capacity, and if there is sufficient capacity, the switch prepares to accept the call. If not, it returns a rejection message toward the source. A switch may accept a call when the sum of the effective average rates of sources that have routes established through this switch and the source making the request is less than the service rate of this switch, and this switch can provide the requested service and buffer shares. Assuming the switch has capacity to accept the call, it then calculates the distortion its queueing will cause this source's traffic using the requested average rate, local knowledge and the algorithm in Chapter 4. If this distortion is more severe than the request has encountered, *i.e.* results in a higher effective average rate than the one in the request, the switch forwards the calculated value to the next switch as the effective average rate. Otherwise it forwards the request as it arrived. The switch then waits for either a confirmation or rejection, and abandons the tentative allocation if it does not receive a confirmation after a sufficient time. The details of communication of acceptance or rejection is left to the actual call establishment protocol. The details of calculation of a source's effective average rate will be discussed in Chapter 4.

3.3.3. The Source Control Algorithm

We now discuss the implementation of the DTW source control, which ensures that cells are sent in accordance with the time window criterion. The implementation uses a credit method, similar to a Leaky Bucket[13]. Credits are in terms of cells of size C. If a source has a credit, it can send a cell, otherwise it must wait until it has a credit. Initially there are $\left\lfloor \frac{\bar{\lambda} I}{C} \right\rfloor$ credits in the credit pool, but when a credit is used, it is restored I time units later, unlike a Leaky Bucket where the credits are restored at the average sending rate.

This algorithm can be implemented with one counter to hold the number of cells that have been sent in the last I time units, and a credit queue. The credit queue is queue of timestamped credits kept in sorted order, with the smallest–valued entry the head of the queue. When the current time is greater than the timestamp on a credit, it means that the cell that caused that credit to be enqueued was sent more than I time units ago. The control algorithm can forget that it sent that cell by decrementing the counter and removing the credit from the queue.

Whenever the algorithm is called to send a cell, it checks the credit queue and removes any entries with timestamps earlier than the current time. The counter is decremented once for each credit removed from the queue. Then the algorithm consults the counter, and if its current value is less than $\left\lfloor \frac{\bar{\lambda} I}{C} \right\rfloor$, then the cell may be sent. To send a cell, a credit is put on the credit queue with a timestamp I time units in the future, the counter is incremented, and the cell is physically transmitted. If the cell cannot be sent, it is discarded.

The algorithm can be modified to queue cells that cannot be sent, but the details of managing the queues depend heavily on where and how the algorithm is implemented. An implementation in a user–level library will use different mechanisms to suspend itself until credits become available than an implementation in hardware. We omit these details.

Changing I requires rewriting entries in the credit queue as well as adjusting the value in the counter. If I increases in size, all the credits must be rescheduled as though they had been sent under the larger time window, and additional worst case credits added to the queue. For example, if the time window increases from I_1 to I_2, all the timestamps in the queue must be reset to be (timestamp$-I_1+I_2$). The worst case credits are added to the credit queue as a though the sources had sent a burst of $\left\lfloor \frac{\bar{\lambda}}{C}(I_2-I_1) \right\rfloor$ cells I_1 time units in the past, with the same number of cells added to the counter. Since the credit queue only records when cells were sent in the past I_1 time units, we must assume that as many cells as possible were sent at the most recent time to insure that the time window criterion is met. If I gets smaller, again from I_1 to I_2, credits for cells sent more than I_2 time units in the past are discarded from the queue and subtracted from the counter, and all remaining credit timestamps are updated to (timestamp$-I_1+I_2$).

This algorithm performs the same function as Zhang's User Behavior Envelope[18], but we have modified the terminology and implementation to allow the source's time window to vary, thereby making DTW admission control part of a dynamic control system.

Although the implementation requires a large queue in the general case, for a more restrictive and realistic case, the system can be implemented by a simple array–based queue. The simple implementation assumes a bounded value of I, and a minimum granularity of the sending times. Then the queue can be implemented as an array–based queue, since the maximum length is bounded, and, instead of requiring a separate entry for each possible time value, they are rounded to the nearest time quantum. Each entry in the queue tells how much to increase the credit pool when that quantum of time has elapsed. Operations on that queue are fixed time operations, so the system can be implemented cheaply.

This concludes the functional description of DTW. We have described a elements of a simple system that mesh together to avoid and to control network congestion. Switches use WFQ to grant sources a guaranteed minimum share of their service rate while monitoring their aggregate queue length for signs of congestion. The congestion state of switches results in feedback to sources telling them either that the network can tolerate more burstiness or that they must cut down to avoid additional congestion. In either case the source control guarantees that any congestion that occurs despite the avoidance mechanisms is bounded in duration. Source throughput is guaranteed. The following chapters discuss the performance of DTW.

Chapter 4

Analytical Results

"If in other sciences we should arrive at certainty without doubt and truth without error, it behooves us to place the foundations of knowledge in mathematics."

— *Roger Bacon,* **Opus Majus, bk. I, ch. 4**

This chapter relates analytical results concerning DTW. We begin with an exploration of DTW source control and how it differs from Leaky Bucket[13] source control. Then we prove the property that a queue served only by DTW sources empties periodically for the single switch case. This property is called DTW stability. We also prove a bound on a source's effective average rate, mentioned in Chapter 3. This effective average rate is shown be at most twice the negotiated average rate. We show that under WFQ, traffic passing through multiple switches is perturbed no worse than traffic that passes directly from the source to the bottleneck switch. These results are used by the algorithm used to establish circuits described in Chapter 3.

4.1. DTW and Leaky Bucket

This section describes the behavior of sources under two source controls, Leaky Bucket and DTW source control. We show that, with equivalent parameters, both policies approach the same throughput limit, and that this limit is the source's negotiated average rate. This does not imply that the two disciplines are equivalent. We show that DTW source control is more strict because a source's throughput is constrained to reach the limiting throughput under DTW source control, while a source policed by Leaky Bucket is not. DTW also allows a source's throughput to exhibit as much burstiness as a Leaky Bucket.

To analyze these controls we define throughput functions. The throughput function defined by a source control algorithm is a function of time, t, that returns the value of the maximum allowed throughput of a source sending under that algorithm over the period $[0, t]$. The value of a throughput function is the throughput measured at the entrance to the network; throughput functions do not reflect network performance. The throughput function reflects the throughput of the input process to the network.

Throughput functions show how much data a source is capable inserting into the network without violating a given source control algorithm. They reflect the best possible throughput to the network for a source that is maximizing throughput. Studying throughput functions illuminates both the worst case behavior of the control algorithm, and the effect a source control algorithm will have on sources that are maximizing throughput.

Throughput functions reflect the worst case behavior for source controls that enforce an average throughput while allowing sources to exhibit burstiness. The source represented by a throughput function will be using the burstiness tolerance of the source control to increase its throughput. The burstiness allowance in both DTW and Leaky Bucket assumes that sources will send a burst of traffic and then be idle for some time independent of the control. Throughput functions reflect the behavior of sources that are not becoming idle more than the control requires. This study investigates how Leaky Bucket and DTW source control force such sources to conform to their negotiated average rate.

4.1.1. Leaky Bucket

Leaky Bucket is a token based source control. A source can send a bit whenever it has a token, which is consumed in sending. Tokens are restored at a rate, r, which is the negotiated average rate of the source, expressed as tokens/sec. Tokens need not be used immediately, but a source may only possess B tokens at any time. The source initially has B tokens. The name ''Leaky Bucket'' arises because this system is analogous to a bucket with capacity B sitting under an open ''token faucet'' running at rate r; B is often referred to as a bucket size.

We define $l_{B,r}(t)$ as the number of bits it is possible for a source policed by Leaky Bucket with bucket size B and rate r to send in $[0,t]$. The value of $l_{B,r}(t)$ is given by:

$$l_{B,r}(t) = B+rt \tag{4.1}$$

The throughput function, $l_{B,r}(t)/t$, is therefore:

$$\frac{l_{B,r}(t)}{t} = \frac{B+rt}{t} = \frac{B}{t}+r \tag{4.2}$$

And in the limit, this is:

$$\lim_{t \to \infty} \frac{l_{B,r}(t)}{t} = \left[\lim_{t \to \infty} \frac{B}{t} \right] + r = r \tag{4.3}$$

Equation (4.3) shows that, in the limit, Leaky Bucket enforces the negotiated average rate, r.

4.1.2. DTW Admission Control

DTW admission control is described in Chapter 3. The system enforces the time window constraint, *i.e.,* over any period of time of length I a source can only send $\bar{\lambda} I$ bits. However, over shorter periods the source can send at its peak rate, $\hat{\lambda}$, as long as it does not violate the criterion. The quantity $d_{\bar{\lambda},I,\hat{\lambda}}(t)$ is analogous to $l_{B,r}(t)$ but represents the number of bits it is possible to send in the interval $[0,t]$ for a source under DTW. The value of $d_{\bar{\lambda},I,\hat{\lambda}}(t)$ is given by:

$$d_{\bar{\lambda},I,\hat{\lambda}}(t) = \bar{\lambda}\left\lfloor \frac{t}{I} \right\rfloor I + \hat{\lambda}\min\left(t - \left\lfloor \frac{t}{I} \right\rfloor I, \frac{\bar{\lambda}I}{\hat{\lambda}}\right) \tag{4.4}$$

The first term of Equation (4.4) represents the data sent during the portion of the sending interval that can be evenly divided into periods of length I. During each of these periods, at most $\bar{\lambda}I$ bits have been sent by the source. The second term represents the amount of data sent in the remaining time, during which the source could send at its peak rate. The minimum is introduced because during the period reflected in the second term, the source can send at most $\bar{\lambda}I$ bits. Equation (4.5) gives the DTW throughput function.

$$\frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t} = \frac{\bar{\lambda}\left\lfloor \frac{t}{I} \right\rfloor I + \hat{\lambda}\min\left(t - \left\lfloor \frac{t}{I} \right\rfloor I, \frac{\bar{\lambda}I}{\hat{\lambda}}\right)}{t} \tag{4.5}$$

We find the limit of the throughput function as $t \to \infty$ to show that DTW enforces the average rate, $\bar{\lambda}$, in the limit.

To find the limit of $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$, note that:

$$\frac{\bar{\lambda}\left\lfloor \frac{t}{I} \right\rfloor I}{t} \leq \frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t} \leq \frac{\bar{\lambda}\left\lfloor \frac{t}{I} \right\rfloor I + \bar{\lambda}I}{t} \tag{4.6}$$

We will prove that the limit of $\frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t}$ goes to $\bar{\lambda}$ by squeezing it between the two expressions in Equation (4.6).

Note that :

$$\frac{\bar{\lambda}(t-I)}{t} \le \frac{\bar{\lambda}\left\lfloor \frac{t}{I} \right\rfloor I}{t} \le \frac{\bar{\lambda}t}{t} \tag{4.7}$$

The limit of the right side of Equation (4.7) is $\bar{\lambda}$ and since $\lim_{t \to \infty} \frac{I}{t} = 0$, the limit of the left side is also $\bar{\lambda}$, hence

$$\lim_{t \to \infty} \frac{\bar{\lambda}\left\lfloor \frac{t}{I} \right\rfloor I}{t} = \bar{\lambda} \tag{4.8}$$

The expression in Equation (4.8) is lower bound in Equation (4.6). Since $\lim_{t \to \infty} \frac{\bar{\lambda}I}{t} = 0$, the upper bound in Equation (4.6) is also $\bar{\lambda}$ in the limit. Therefore $\lim_{t \to \infty} d_{\bar{\lambda},I,\hat{\lambda}}(t)/t = \bar{\lambda}$. This indicates that DTW also enforces the average rate in the limit. However, stronger statements can be made about the DTW behavior.

4.1.3. Comparison and Contrast of Limits

In the previous sections, we showed that the two source controls enforce a negotiated average rate in the limit. We now investigate how the two approach that limit. A Leaky Bucket can be shown to be equivalent to DTW if the parameters for the Leaky Bucket are chosen as:

$$B = \bar{\lambda}I\left(1 - \frac{\bar{\lambda}}{\hat{\lambda}}\right) \tag{4.9}$$

$$r = \bar{\lambda}$$

A Leaky Bucket with those parameters allows sources to send a burst of $\bar{\lambda}I$ cells over $\frac{\bar{\lambda}I}{\hat{\lambda}}$

seconds at rate $\hat{\lambda}$, leaving the bucket empty, for $\bar{\lambda} < \hat{\lambda}$. In other words, $l_{B,r}(\frac{\bar{\lambda}I}{\hat{\lambda}}) = \bar{\lambda}I$ for $B = \bar{\lambda}I(1-\frac{\bar{\lambda}}{\hat{\lambda}})$ and $r = \bar{\lambda}$. A DTW source and a Leaky Bucket source with parameters given by Equation (4.9) are equivalent in the sense that sources policed by either can send equal size bursts and have the same average rate enforced.

We will show that these two admission strategies approach the same limit at the same rate, in the sense that as $t \to \infty$, $\frac{l_{B,r}(t)}{t} \geq \frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t}$ and the two are equal infinitely often.

To show that $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ and $l_{B,r}(t)/t$ are equal infinitely often, we will need the following lemma.

Lemma 1

$\frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t}$ takes on its maximum value over any interval $[kI, (k+1)I]$ at $t = kI+\frac{\bar{\lambda}I}{\hat{\lambda}}$, for k a positive integer.

Proof

Before proving the lemma, we provide an intuition to guide the reader. The function $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ represents the throughput of a source repeatedly sending bursts of $\bar{\lambda}I$ bits at its peak rate and remaining idle for the remainder of the time window. One expects that source's throughput to reach a maximum immediately after sending one of the bursts. Because the source is sending these bursts at its peak rate, it will have just completed sending one at $t = kI+\frac{\bar{\lambda}I}{\hat{\lambda}}$.

We will prove the lemma by showing that the DTW throughput function increases for $t = kI + \Delta$ when $0 < \Delta < \frac{\bar{\lambda} I}{\hat{\lambda}}$ and that the DTW throughput function decreases for $t = kI + \Delta$, when $\frac{\bar{\lambda} I}{\hat{\lambda}} < \Delta < I$. We then show that $kI+\frac{\bar{\lambda} I}{\hat{\lambda}}$ is the largest of the excluded points on the interval.

First we show that for $t = kI$ and $0 < \Delta < \frac{\bar{\lambda} I}{\hat{\lambda}}$, $\frac{d_{\bar{\lambda},I,\hat{\lambda}}(t+\Delta)}{(t+\Delta)} = \frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t} + \frac{\Delta(\hat{\lambda}-\bar{\lambda})}{t+\Delta}$.
Since $\frac{\Delta(\hat{\lambda}-\bar{\lambda})}{t+\Delta} > 0$, this shows that $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ is increasing for $kI < t < kI+\frac{\bar{\lambda} I}{\hat{\lambda}}$. This fact is revealed by direct computation.

$$\frac{d_{\bar{\lambda},I,\hat{\lambda}}(t+\Delta)}{t+\Delta} = \frac{\bar{\lambda}\left\lfloor\frac{t+\Delta}{I}\right\rfloor I+\hat{\lambda}\min\left(t+\Delta-\left\lfloor\frac{t+\Delta}{I}\right\rfloor I, \frac{\bar{\lambda} I}{\hat{\lambda}}\right)}{t+\Delta} \tag{4.10}$$

Since $\Delta < \frac{\bar{\lambda} I}{\hat{\lambda}}$ and $t = kI$, $\left\lfloor\frac{t+\Delta}{I}\right\rfloor I = t$. This implies that the first term of the numerator reduces to $\bar{\lambda} t$. The second term reduces to $\hat{\lambda}\Delta$ since $\Delta<\frac{\bar{\lambda} I}{\hat{\lambda}}$. Factoring, reducing, and noting that $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t = \bar{\lambda}$ at $t = kI$ gives:

$$\frac{d_{\bar{\lambda},I,\hat{\lambda}}(t+\Delta)}{t+\Delta} = \frac{\bar{\lambda} t+\hat{\lambda}\Delta}{t+\Delta} = \frac{\bar{\lambda}(t+\Delta)+(\hat{\lambda}-\bar{\lambda})\Delta}{t+\Delta}$$

$$= \frac{\bar{\lambda}(t+\Delta)}{t+\Delta}+\frac{(\hat{\lambda}-\bar{\lambda})\Delta}{t+\Delta} = \bar{\lambda}+\frac{(\hat{\lambda}-\bar{\lambda})\Delta}{t+\Delta}$$

$$= \frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t} + \frac{(\hat{\lambda}-\bar{\lambda})\Delta}{t+\Delta} \tag{4.11}$$

Equation (4.11) shows that the DTW throughput function is increasing on $(kI,\ kI+\frac{\bar{\lambda}I}{\hat{\lambda}})$, and completes the first part of our proof.

We now show that $\frac{d_{\bar{\lambda},I,\hat{\lambda}}(t+\Delta)}{(t+\Delta)}$ is decreasing for $\frac{\bar{\lambda}I}{\hat{\lambda}} < \Delta < I$. For all values $\frac{\bar{\lambda}I}{\hat{\lambda}} < \Delta < I$, the second term in the numerator of Equation (4.5) is a constant, $\bar{\lambda}I$. The first term in the numerator of Equation (4.5) remains constant, with the value $kI\bar{\lambda}$, across the entire interval of interest for this lemma. Since the numerator of $d_{\bar{\lambda},I,\hat{\lambda}}(t+\Delta)/(t+\Delta)$ is constant, while the denominator continues to increase with increasing Δ, $d_{\bar{\lambda},I,\hat{\lambda}}(t+\Delta)/(t+\Delta)$ is strictly decreasing on the interval $(kI+\frac{\bar{\lambda}I}{\hat{\lambda}},\ (k+1)I)$. Since the DTW throughput function increases over $(kI,\ kI+\frac{\bar{\lambda}I}{\hat{\lambda}})$ and decreases over $(kI+\frac{\bar{\lambda}I}{\hat{\lambda}},\ (k+1)I)$ the maximum on the interval $(kI,(k+1)I)$ must be realized at $t = kI+\frac{\bar{\lambda}I}{\hat{\lambda}}$. Since $\frac{d_{\bar{\lambda},I,\hat{\lambda}}(kI)}{kI} = \bar{\lambda}$ for all k, and $\frac{d_{\bar{\lambda},I,\hat{\lambda}}(kI+\bar{\lambda}I/\hat{\lambda})}{kI+\bar{\lambda}I/\hat{\lambda}} > \bar{\lambda}$, $kI+\frac{\bar{\lambda}I}{\hat{\lambda}}$ is also the maximum over $[kI,(k+1)I]$.

Since the proof does not constrain k, it holds for any positive integer. This proves the lemma. □

With this lemma proven, we now show that the two throughput functions are equal infinitely often, and that the points of equality are at these maxima. We then prove that the DTW throughput function is less than or equal to the Leaky Bucket throughput function at all points.

Theorem 1

$\frac{l_{B,r}(t)}{t} = \frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t}$ at $t = kI + \frac{\bar{\lambda}I}{\hat{\lambda}}$ for k a positive integer, assuming the two throughput functions are defined for equivalent controls.

Proof

The proof consists of evaluating $\frac{l_{B,r}(t)}{t}$ and $\frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t}$ at the given value of t. Equation (4.12) simplifies $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ for $t = kI + \frac{\bar{\lambda}I}{\hat{\lambda}}$.

$$\frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t} = \frac{k\bar{\lambda}I + \hat{\lambda}\frac{\bar{\lambda}I}{\hat{\lambda}}}{kI + \frac{\bar{\lambda}I}{\hat{\lambda}}} = \frac{k\bar{\lambda}I + \bar{\lambda}I}{kI + \frac{\bar{\lambda}I}{\hat{\lambda}}} = \frac{(k+1)\bar{\lambda}I}{kI + \frac{\bar{\lambda}I}{\hat{\lambda}}} \tag{4.12}$$

Equation (4.13) shows a similar computation of $\frac{l_{B,r}(t)}{t}$ for an equivalent Leaky Bucket at the same value of t:

$$\frac{l_{B,r}(t)}{t} = \frac{B+rt}{t} = \frac{\bar{\lambda}I(1-\frac{\bar{\lambda}}{\hat{\lambda}}) + \bar{\lambda}t}{t} = \frac{\bar{\lambda}I(1-\frac{\bar{\lambda}}{\hat{\lambda}}) + \bar{\lambda}(kI + \frac{\bar{\lambda}I}{\hat{\lambda}})}{kI + \frac{\bar{\lambda}I}{\hat{\lambda}}} \tag{4.13}$$

$$= \frac{\bar{\lambda}I - \frac{\bar{\lambda}^2 I}{\hat{\lambda}} + k\bar{\lambda}I + \frac{\bar{\lambda}^2 I}{\hat{\lambda}}}{kI + \frac{\bar{\lambda}I}{\hat{\lambda}}} = \frac{(k+1)\bar{\lambda}I}{kI + \frac{\bar{\lambda}I}{\hat{\lambda}}}$$

Equation (4.12) and Equation (4.13) show that the Leaky Bucket throughput curve and the DTW throughput curve meet infinitely often, at $t = kI + \bar{\lambda}I/\hat{\lambda}$ for k a positive integer. □

Theorem 2

$\frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t} \leq \frac{l_{B,r}(t)}{t}$ for all t, assuming the two throughput functions are for equivalent controls.

Proof

We have shown that the two functions are equal at the maxima of the DTW throughput function, which are at $t = kI + \frac{\bar{\lambda}I}{\hat{\lambda}}$. Furthermore, the DTW throughput function has the value $\bar{\lambda}$ at $t = kI$. The equivalent Leaky Bucket throughput function is always greater than that value.

Since $l_{B,r}(t)/t$ is monotonically decreasing and the two functions meet at the maxima of $\frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t}$, we know that all values of $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ where $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ is increasing are less than $l_{B,r}(t)/t$. Therefore, $\frac{l_{B,r}(t)}{t} > \frac{d_{\bar{\lambda},I,\hat{\lambda}}(t)}{t}$ on the intervals $(kI\ ,kI+\bar{\lambda}I/\hat{\lambda})$ for k a positive integer.

Over the intervals where $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ is decreasing, namely $(kI+\bar{\lambda}I/\hat{\lambda},\ (k+1)I)$, both terms in the numerator of Equation (4.5) are constant. The first derivative of $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ at these points is $-\frac{(k+1)\bar{\lambda}I}{t^2}$ and the first derivative of $l_{B,r}(t)/t$ is $-\frac{\bar{\lambda}I(1-\bar{\lambda}/\hat{\lambda})}{t^2}$ at all points. The two functions are equal at $t = kI + \frac{\bar{\lambda}I}{\hat{\lambda}}$, and the first derivative of the DTW throughput function is less than that of the Leaky Bucket throughput function throughout the interval $(kI+\bar{\lambda}I/\hat{\lambda},(k+1)I)$. Therefore, the $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ is less than $l_{B,r}(t)/t$ on that interval.

The sets of intervals and the points above cover all positive real numbers, so the theorem is proven. □

Theorems 1 and 2 imply that both $l_{B,r}(t)/t$ and $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ approach their limits at the same speed. Practically, this means that the DTW performs as well or better than Leaky Bucket in terms of enforcing an average rate.

The DTW throughput function $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ is equal to the limit value $(\bar{\lambda})$ for an infinite number of points, namely $t = kI$, for k a positive integer. This property is responsible for DTW stability, as shown later in this chapter. In contrast $l_{B,r}(t)/t$ never reaches

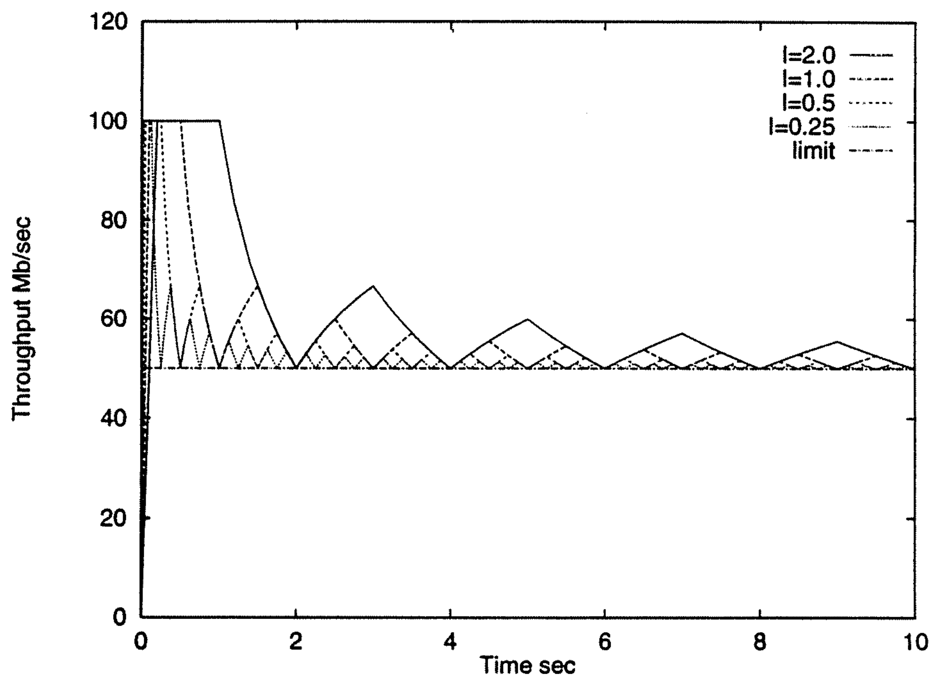

Figure 4.1 : DTW Throughput Function $(d_{\bar{\lambda},I,\hat{\lambda}}(t)/t)$

the limit value, which is why Leaky Bucket is not DTW stable. The contrast is important. A properly aggressive or malicious source can always get a higher throughput than it negotiates from Leaky Bucket, but any source operating under DTW will be forced to its negotiated throughput repeatedly. Furthermore, the frequency with which the source will be forced to its negotiated throughput is directly related to its time window. This can be seen in Figure 4.1.

We provide plots of both $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ and $l_{B,r}(t)/t$, to aid in the comparison of the controls. Figure 4.1 shows $d_{\bar{\lambda},I,\hat{\lambda}}(t)/t$ for $\bar{\lambda} = 50$ Mb/sec, $\hat{\lambda} = 100$ Mb/sec, and a range of time window values I. Each point where a throughput line intersects the flat line at 50

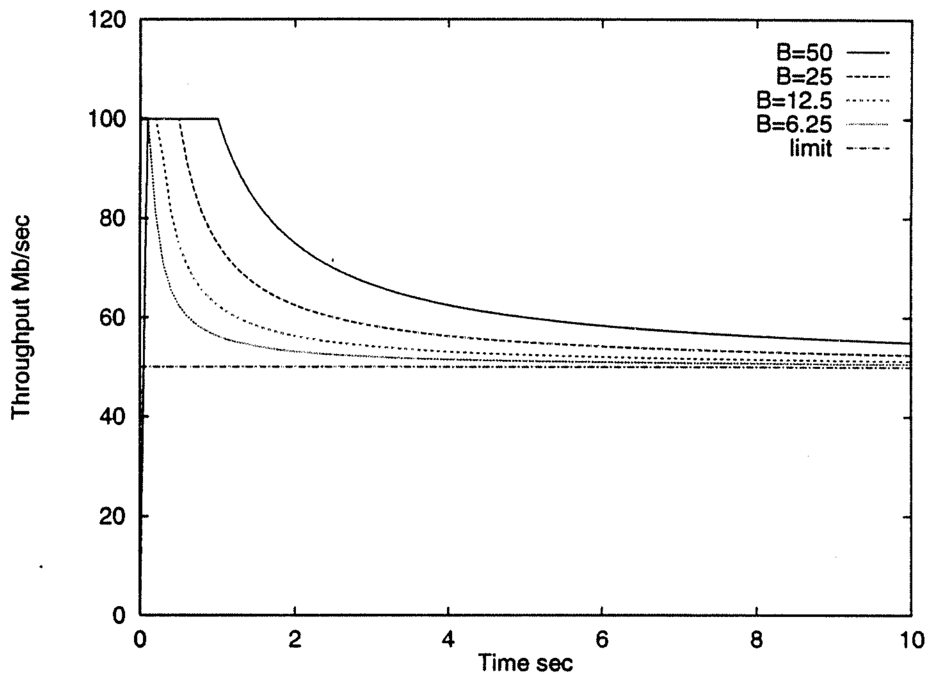

Figure 4.2 : Leaky Bucket Throughput Function ($l_{B,r}(t)/t$)

Mb/sec throughput is a place where the throughput of that source is constrained to be its average rate. Figure 4.2 shows a a set of plots of $l_{B,r}(t)/t$ for the equivalent leaky buckets. The plots in Figure 4.2 represent a modified $l_{B,r}(t)/t$ that takes into account a finite maximum sending rate of the source. These asymptotically approach the 50 Mb/sec throughput limit, but never reach it. Figure 4.3 shows the relationship between DTW and Leaky Bucket directly. It is clear where the DTW throughput and Leaky Bucket throughput are equal, as well as where the DTW throughput is constrained to meet the enforced average.

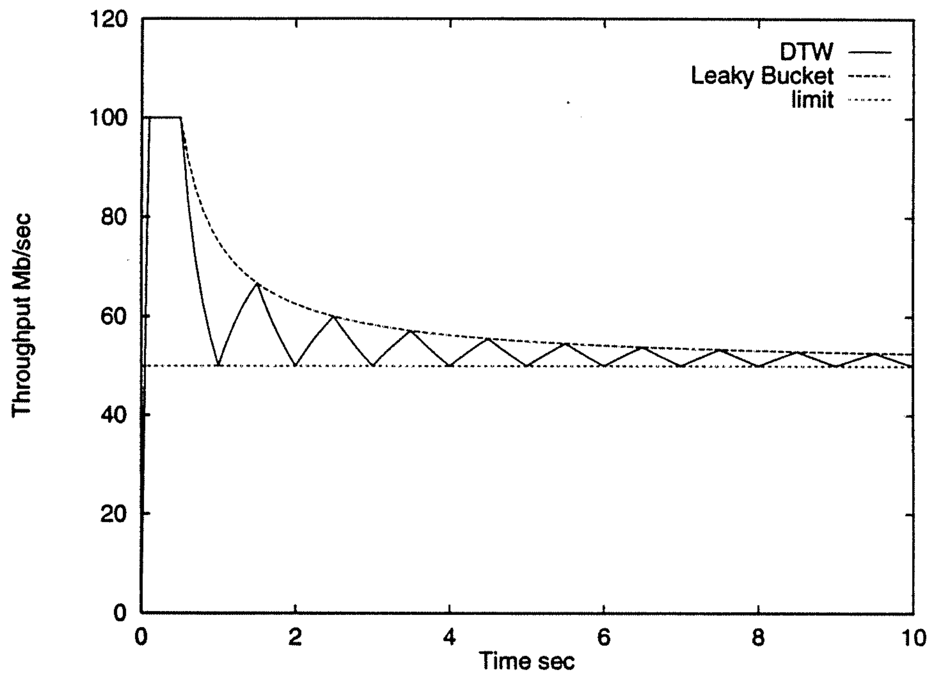

Figure 4.3 : Throughput Comparison

To summarize, The throughput functions for DTW and Leaky Bucket have been defined and analyzed. The DTW throughput function is shown to approach the same limit as the Leaky Bucket throughput function having equivalent parameters. The Leaky Bucket throughput function is a tight upper bound on the DTW function, since they share infinitely many points. The DTW function also reaches the throughput limit at infinitely many points. This shape of the throughput function reflects the DTW stability of the DTW source control system. The fact that the the DTW throughput function returns to the limiting throughput is the direct cause of switches serving DTW sources periodically emptying their queues.

This analysis should highlight the similarities between Leaky Bucket and DTW source control as well as their differences. DTW source control can be thought of as a version of Leaky Bucket where the restoration of tokens is based on the sending pattern of a source. When a source is sending as fast as it can, this results in the tokens being restored every I seconds at a rate of $\hat{\lambda}$ for $\frac{\bar{\lambda} I}{\hat{\lambda}}$ seconds. The idea of DTW as a Leaky Bucket with a token rate dependent on a source's transmission history over the last I seconds may provide a useful intuition.

4.2. DTW Stability

This section proves theorems regarding DTW stability. DTW stability is the property that all queues in a DTW system will periodically empty, controlling congestion without feedback. Feedback is used to avoid congestion, and tune network utilization. We explore DTW stability by proving a stability theorem for a single switch and showing the conditions required to extend this theorem to all switches in the network. We use those conditions to derive circuit establishment criteria that ensure DTW stability.

4.2.1. Traffic Model

In this analysis, we assume that traffic is a continuous quantity, and that ideal WFQ switches are used. The data sent by a source or switch during a period when it begins sending data, sends continuously for a period, and stops sending, is a burst. The rate at which the source or switch sends a burst need not remain constant. An ideal WFQ switch is a variable rate server that serves all bursts awaiting service simultaneously. The rate at which each burst is served is based on the service share of its source, and the number of other bursts being served. The instantaneous rate at which an ideal WFQ switch will serve a queued burst from source i is

$$\frac{s_i \mu}{\sum_j s_j} \tag{4.12}$$

where the summation is taken over the circuits with bursts queued at the switch. These are the same assumptions made to make the bound in Equation (3.2) hold.

Figure 4.4 shows the effect of an idealized switch on continuous traffic. The upper graph is a plot of rate versus time for two bursts arriving at a switch; the lower plot is of the same quantities for the same bursts leaving the switch. The leftmost burst burst is served by the switch at a constant rate equal to half its arrival rate. This burst is output at half the rate for twice the time. The rightmost burst is served at a variable rate by the switch. The service rate of this burst changes during its transmission from the switch. The bursts being served at any time divide the service rate of the switch in a manner given by Equation (4.12). Since a new burst may arrive or a switch may finish sending a burst at any time, the instantaneous rate at which bursts are served varies with time. Although this instantaneous rate may change, it is always at least the guaranteed minimum given in Equation (3.2).

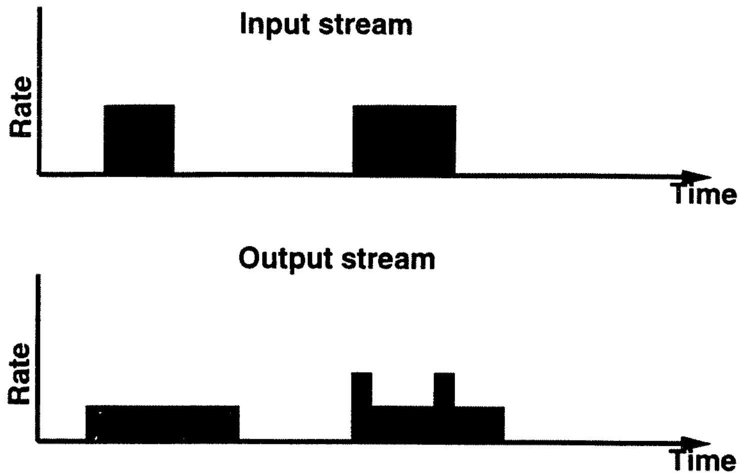

Figure 4.4 : Continuous Traffic Being Served at an Ideal Switch

An ideal switch cannot serve traffic faster than it arrives. If a burst arrives at a lower rate than the rate at which the switch can send it, the switch must serve it at the arrival rate. The difference between the amount of data that has arrived at a switch and the amount of data the switch has sent is buffered at the switch. The amount of a burst that the switch has sent can be at most the amount that has arrived.

The assumption of continuous traffic and ideal Weighted Fair Queueing is analogous to the use of perfect processor sharing as a scheduling discipline in a queueing analysis of a computer system. The same way that processor sharing models round robin scheduling with an infinitesimal scheduling quantum, the idealized WFQ switch serving continuous traffic models a network with infinitesimal cells. Both are simplifying assumptions used to analyze complex systems.

In reality, traffic is not continuous, and switches do not implement ideal Weighted Fair Queueing. However we believe the approximations are justified in a high speed ATM network. Parekh has shown that the deviation from the ideal behavior of WFQ is directly related to the ratio of the cell size to the rate of the switch[63]. ATM cells are small compared to the bandwidth of switches in high speed networks. As an example, the ratio of an ATM cell size to the backplane bandwidth of the XUNET switch is 738 ns. The maximum deviation of the queueing delay of a cell from that of a cell–sized burst is this ratio multiplied by the number of established circuits.

Furthermore, the small size of ATM cells compared to burst sizes approximates continuous traffic. Few sources will use ATM cells as their unit of data transfer, since the overhead of sending single cells can be high. Sources are likely to send larger packets that will be transmitted across the network as bursts of ATM cells. These bursts will be gathered at the sink and reassembled into packets. A switch can begin forwarding such a burst when it receives the first cell of that burst. This approximates the immediate forwarding of continuous traffic. For example, if a source is sending 4800 byte packets as bursts of ATM cells, a switch can begin forwarding a packet when only 1% of that packet (one cell) has arrived. We believe most sources will send larger packets than 4800 bytes.

Because of these two effects of switching ATM cells, we believe that the continuous traffic approximations are valid. The theorems proven here have all held in simulations where the size of the smallest transmission unit was larger than an ATM cell. These simulations are described in Chapter 5.

4.2.2. Single Switch DTW Stability

Consider a single switch with a maximum time window, MTW, serving k DTW sources. If the switch is empty, *i.e.,* all of its per–circuit queues are empty, at time t, it will be empty again at or before t+MTW. A switch which has this property is DTW stable by definition. We state the property as a theorem.

Theorem 3

Assume a single switch serving traffic from k sources each observing the time window criterion (Equation (3.1)). Further assume:

(1) The switch service rate is μ whenever any of its queues are non-empty. (*I.e.,* its queueing discipline is work conserving)

(2) MTW is maintained by the switch, and remains constant over the period in question.

(3) Source i's time window parameters are $\hat{\lambda}_i$, $\bar{\lambda}_i$, I_i, and r_i an integer. $I_i = \frac{\mathrm{MTW}}{r_i}$.

Source i's instantaneous sending rate is $\lambda_i(t)$.

(4) $\mu \geq \sum_{j=1}^{k} \bar{\lambda}_j$

Then: If a switch queue transitions from an empty state to having a non-zero queue length at time t, it will again be empty at or before time t+MTW.

Proof

We proceed by contradiction, assuming that the queue will not have a zero queue length in the time period $[t, t+\mathrm{MTW}]$. This means the switch will be emptying queues at rate μ for MTW time units. Since the queue length was zero at t and by assumption the queue length is non-zero until time t+MTW, the queue length at time t+MTW will be:

$$\left[\sum_{i=1}^{k} \int_{t}^{t+MTW} \lambda_i(\tau)d\tau\right] - MTW\mu \tag{4.14}$$

$$=\left[\sum_{i=1}^{k}\left[\sum_{j=1}^{r_i}\int_{t+(j-1)\frac{MTW}{r_i}}^{t+j\frac{MTW}{r_i}}\lambda_i(\tau)d\tau\right]\right]-MTW\mu \tag{4.15}$$

Since each of the integrals in the inner sum of Equation (4.14) is over a period of length I_i, Equation (3.1) yields the following:

$$\leq\left[\sum_{i=1}^{k}\left[\sum_{j=1}^{r_i}\bar{\lambda}_i I_i\right]\right]-MTW\mu \tag{4.16}$$

$$=\left[\sum_{i=1}^{k}r_i\bar{\lambda}_i I_i\right]-MTW\mu \tag{4.17}$$

and since $I_i = \dfrac{MTW}{r_i}$

$$=MTW\left[\sum_{i=1}^{k}\bar{\lambda}_i-\mu\right] \tag{4.18}$$

Since $\mu \geq \sum_{i=1}^{k}\bar{\lambda}_i$ by Assumption 3, the queue length at $t+MTW$ must be zero or less, which contradicts the assumption that it was non-zero at that time.

□

The proof shows that even if all sources are sending as much traffic as they are allowed to send, the source control algorithm will force the sum of their throughputs over a given period to be at most the switch service rate. This can be seen graphically in Figure 4.1. Points where the throughput function is equal to the negotiated average rate, which is 50 Mb/sec in Figure 4.1, indicate times when that source must have sent no more data than if it sent continuously at its average rate. Since time windows are unanchored, (Equation (3.1) holds for all t), these points will always coincide for sources sharing time windows. This occurs because we are free to begin measuring source throughput at any point. When all sources using a switch share a coincident point, a switch serving those sources must have been able to serve all they could have sent. The switch is therefore in the same state as it was at time zero. If the queue was empty at time zero, and we can always pick such a time as time zero, the switch will be empty again. The fact that all time windows are integer divisors of MTW guarantees that they will all meet at least once per MTW.

If we relax the assumption that all source time windows are the result of integer divisions of MTW, the theorem remains true; except that the bounding time becomes the least common integer multiple of the individual sources' time windows. Since arbitrary values of r can imply an arbitrarily long time bound, we constrain r to be an integer so that the upper bound on congestion times is MTW. Any constraint that bounds the least common integer multiple of the time windows is sufficient to bound the renewal interval.

The theorem is very general. No constraints on the switch queueing discipline are made other than it being work–conserving. It is a powerful feature that DTW source control enforces this stability under very general conditions.

DTW stability reduces the effects of the high latency of control messages on congestion in the network. Consider a switch in a normal packet window-based congestion control method, like the DECbit algorithm[2]. In the worst case, as the switch becomes congested, the switch's queue length grows and will continue to grow until the control messages that the switch is sending reach the sources that are flooding it. The severity of the congestion, specifically the queue length at the bottleneck switch, is directly related to the time required for the control messages to be received and the to bandwidth of the network[24]. Once the sources have been throttled back, the network must still remove the excess switch queue buildup in the face of repeated retransmissions. This reduces the effectiveness of the congestion messages once they arrive.

A switch congested in a DTW-based network will always have a zero queue length one MTW after the congestion started. There is no unbounded buildup of the queue length to make the congestion worse; the time that the switch is congested is always bounded. The congestion will clear itself even if no control message appears, although it may recur if no corrective action is taken. Control messages now play the role of preventing congestion and packet loss from recurring.

4.3. Multiple Switches and DTW Stability

The requirements for ensuring that a single switch fed by DTW sources will be DTW stable is not sufficient to ensure the DTW stability of all switches along a multiple switch path. This is because switches can change the temporal relationships between bursts, so that the streams of bursts coming into a switch (input streams) and streams of bursts coming out of a switch (output streams) can be quite different. This effect is illustrated in Figure 4.4. This section describes how to characterize the worst case changes in the burst streams. We define and calculate the effective average rates that traffic exhibits at switches beyond the first.

The goal of this analysis is to find conditions under which Theorem 3 holds at interior switches. Another choice would be to try to bound congestion times at switches without trying to apply Theorem 3. This approach has merit, and the work of Cruz[85, 86] suggests that success along these lines is possible. Proceeding along those lines does not seem to provide the bound on the distortion presented in Theorem 5, however, so we chose this approach.

4.3.1. Calculation of Effective Average Rate

Because traffic at the switches is changed by the differing rates with which each burst is served, a source that sends traffic meeting the time window criterion into the network can have its traffic emitted from a switch in such a way that the traffic violates the time window criterion for the source's negotiated parameters. That is, a second switch monitoring traffic may observe an interval of length I in which more than $\bar{\lambda}I$ bits arrive. For worst case traffic, the number of bits in such an interval divided by I is the effective average rate of the source. The effective average rate of sources can be bounded.

Because DTW switches use WFQ, they can serve bursts at different rates, but not at arbitrary rates. From Equation (3.2), we know that source i is guaranteed a minimum service rate of $\frac{s_i}{\sum_j s_j}\mu$, for a switch of rate μ. This is sufficient to provide bounds on a source's effective average rate. We state these bounds as a theorem.

Theorem 4

Given:

(1) All constraints of Theorem 3 are met, except where expressly contradicted here.

(2) The switch uses ideal Weighted Fair Queueing and has service rate μ.

(3) There are k sources using a switch sharing a time window size I, and having individual $\bar{\lambda}_i$, $\hat{\lambda}_i$, and s_i, where s_i is the service share of source i at the switch. $\forall i$, $s_i\mu \geq \bar{\lambda}_i$.

Then the worst case effective average rate for an arbitrary source i, which is computed by dividing the number of bits seen in a period of length I by I, is $\bar{\lambda}_i(2-\min(1, \frac{s_i\mu}{\hat{\lambda}_i \sum_{j=1}^{k} s_j}))$

Proof

We calculate the largest amount of data that can appear in any interval of length I using the bound on a switch's minimum rate under WFQ. Consider source i sending two maximum sized bursts that arrive at a switch as closely in time as the time window criterion allows. The bursts are of size $\bar{\lambda}_i I$. The worst case effective average rate of the traffic seen on the output stream occurs when the first burst is sent by the switch at the switch's guaranteed minimum rate for source i, and the second is passed through unchanged. The number of bits from the first burst that are delayed to within I time units of the second determine the effective average rate.

The input stream of Figure 4.5 has two reference intervals marked to show the traffic pattern meets the time window criterion. The output stream has the interval used

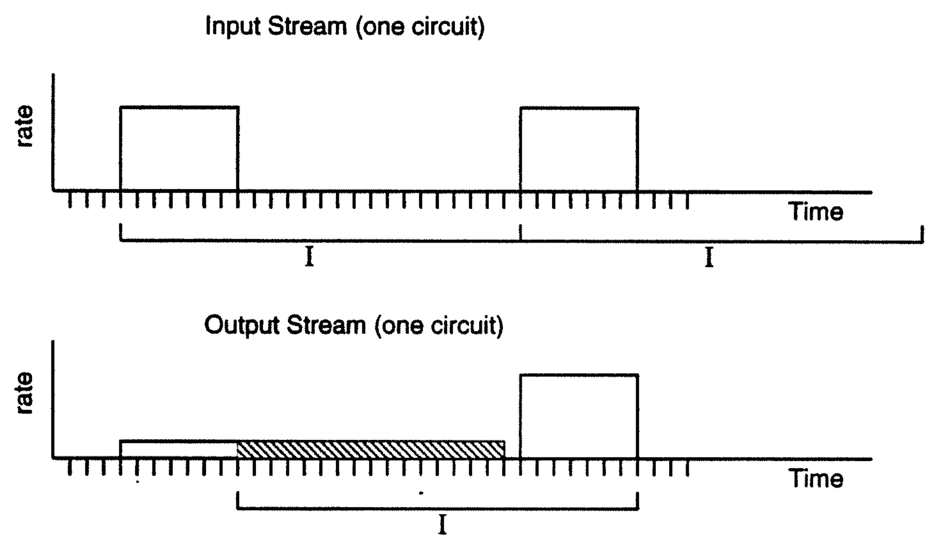

Figure 4.5 : Maximum Distortion of Traffic

to measure the average rate marked. The number of bits in this interval, divided by I, gives the effective average rate of traffic leaving this switch. Note that since Theorem 3 holds at the switch, the first burst will have been completely sent before the second arrives at the switch. Similarly, no bits from any previous time window can delay the first burst.

To compute the number of bits in the measurement interval marked on the output stream of Figure 4.5, we calculate the number of bits in the shaded region, and add the $\bar{\lambda}_i I$ bits in the second burst. The number of bits in the shaded region of the first burst is found by subtracting the number of bits in the unshaded region from the total burst size of $\bar{\lambda}_i I$. Bits from the first burst arrive at the switch for $\frac{\bar{\lambda}_i I}{\hat{\lambda}_i}$ time units, since source i's largest burst is being sent at $\hat{\lambda}_i$. The arrival of the last bit ends the unshaded region on the output stream. During the time the first burst is arriving, the switch is sending the burst at source i's guaranteed minimum rate, $\frac{s_i}{\sum_{j=1}^{k} s_j}\mu$, and therefore removes the following number of bits from the queue:

$$\left(\frac{s_i}{\sum_{j=1}^{k} s_j}\mu\right)\left(\frac{\bar{\lambda}_i I}{\hat{\lambda}_i}\right) = \frac{\bar{\lambda}_i I s_i \mu}{\hat{\lambda}_i \sum_{j=1}^{k} s_j} \tag{4.19}$$

the remaining number of bits in source i's queue are removed during the measurement interval (the shaded region). The total in the shaded region is therefore:

$$\bar{\lambda}_i I - \frac{\bar{\lambda}_i I s_i \mu}{\hat{\lambda}_i \sum_{j=1}^{k} s_j} \tag{4.20}$$

and the total sent during the measurement interval, which includes all of the second burst is :

$$\bar{\lambda}_i I + \bar{\lambda}_i I - \frac{\bar{\lambda}_i I s_i \mu}{\hat{\lambda}_i \sum_{j=1}^{k} s_j} = 2\bar{\lambda}_i I - \frac{\bar{\lambda}_i I s_i \mu}{\hat{\lambda}_i \sum_{j=1}^{k} s_j} = \bar{\lambda}_i I \left(2 - \frac{s_i}{\hat{\lambda}_i \sum_{j=1}^{k} s_j} \mu \right) \tag{4.21}$$

Dividing the number of bits by I gives an effective average rate of:

$$\bar{\lambda}_i \left(2 - \frac{s_i}{\hat{\lambda}_i \sum_{j=1}^{k} s_j} \mu \right) \tag{4.22}$$

For cases in which there is no queueing at the switch, the effective average rate is $\bar{\lambda}_i$. This possibility is accounted for by the minimum function in the expression for the effective average rate in the theorem statement. □

Theorem 4 is applied to determine whether a connection can be established through an internal switch. Theorem 4 derives the effective average rate that a switch can receive from a source, whose traffic has been previously queued by a switch. Theorem 4 is applied at the intermediate switch to calculate the source's effective average rate. If the sum of these effective average rates for sources using the switch is less than the service rate of the switch, Theorem 3 holds at the switch, and it is DTW stable. Circuits that do not violate the preconditions of Theorem 3 when their effective average rate is calculated by Theorem 4 are permitted by the network. This is enforced using the algorithm described in Chapter 3. The next section shows that the effective average rate need only be calculated at the switch with the lowest minimum guaranteed rate. Before proving that theorem, we comment on Theorem 4.

Theorem 4 trivially holds for sources that have different time windows if those time windows are integer multiples of each other. To be precise, if for all time windows of sources using the switch, I_j, $\exists\ k \in \mathbf{Z}$ such that $kI_j = AMTW$, the theorem holds. Any source meeting the time window criterion for I, $\bar{\lambda}$, and $\hat{\lambda}$ meets it for kI, $\bar{\lambda}$, and $\hat{\lambda}$ for $k \in \mathbf{Z}$. This follows directly from the definition of the time window criterion. So all sources meet assumption 3 of Theorem 4, since they all meet the time window criterion for $I = AMTW$. The expression for a source's effective average rate given by Theorem 4 can be used at a switch where all source time windows are integer divisors of the network AMTW.

If traffic is not continuous, inserting Parekh's guaranteed minimum rate for real traffic into Equation (4.19) will give an exact bound on the effective average rate. For that matter, any queueing discipline that can guarantee a source a minimum rate of service can be modelled by this equation.

The derivation of the effective average rate must consider worst case traffic, but such traffic may be rare in a real network. The conditions necessary for traffic to exhibit the worst case effective average rate predicted by Theorem 4 require significant collusion between sources. All sources must time their traffic so that it all arrives at the switch simultaneously, and send large enough bursts that the switch is loaded for the entire time that source i's first burst is queued. Then the sources other than source i have to stop sending so that only source i's second burst will be served at the switch's full service rate. This is unlikely to occur if the sources using the switch are not in collusion. Analytically determining the probability of a set of random sources sending worst case traffic is possible, but is only applicable to traffic that meets the statistical model used to calculate that probability.

Our simulation experiments have shown that the chance of randomly exhibiting worst case traffic is low for many traffic models. Some network administrators may choose to allow sources to establish connections that would violate the constraints of Theorem 3 at an internal switch by calculating their effective average rate by a method other than the formula in Theorem 4. The effective average rate in Theorem 4 is realizable, so using some other value makes the network theoretically unstable. However, experience with simulation has shown that if alternative effective average rate calculations are be used, a network without collusion among sources is likely to behave as if it were DTW stable, in the sense that no switch will exhibit a busy period in excess of the AMTW.

4.3.2. The Effect of Many Switches

A naive application of Theorem 4 would be to have the first switch calculate the effective average rate of a source as seen by the second switch and forward it to that switch. The second switch would use the effective average rate from the first switch to be sure that the sum of the average rates is less than the its service rate, then repeat the process. The effective average rate of the source would become larger from switch to switch without bound. Fortunately, it is only necessary to consider the switch that will distort the traffic the most.

This is somewhat counterintuitive, since we think of switches as independently rearranging traffic. Under ideal WFQ each virtual circuit can be observed in isolation as being served by a variable rate switch. Using the continuous traffic assumption, traffic is is served immediately at a rate determined by the number of other circuits being served.

We show that, under these assumptions, only the traffic distortion induced by the bottleneck switch is important. The bottleneck switch is the switch with the minimum guaranteed service rate for the virtual circuit. The worst case distortion, for the purposes of determining effective average rate, occurs when a source burst is served by a switch at the source's minimum guaranteed rate. Slowing this burst potentially pushes some of the data closer to an undelayed burst as seen in the proof of Theorem 4. (See Figure 4.5). We show that the effective average rate of a stream leaving the bottleneck switch is the effective average rate of that stream leaving the last switch of the virtual circuit.

Theorem 5

In the worst case, a burst of continuous traffic passing through multiple ideal WFQ switches has its sending rate reduced and sending time increased as though it had only passed through the bottleneck switch.

Proof

We proceed by proving the theorem holds for any two adjacent switches, and then iterate.

Consider three cases of a burst being served by two switches in series. Throughout this proof, we will refer to the guaranteed minimum service rate of a burst's source at the switch as the guaranteed minimum rate of the switch. We are only concerned with one burst. The three cases are distinguished by the relative guaranteed minimum rates of the switches. In all cases, the guaranteed minimum rates of the switches are less than the arrival rate of the burst to the first switch, since switches with a higher guaranteed average rate do not distort traffic at all. Since we are considering the worst case, we assume no burst or part thereof is lost, and that each switch serves each burst at its guaranteed minimum rate.

First, consider sending the same burst through two switches with equal guaranteed minimum rates. The burst leaves the first switch at the guaranteed minimum rate of the first switch, and arrives at the second. The second switch sends the burst at the exact speed that it receives it. Other than the propagation delay, the output streams from both switches are the same. The second switch introduces no new distortion, so whichever switch is chosen as the bottleneck determines the distortion of the burst.

If the second switch is faster than the first, in terms of the guaranteed minimum rate, the output stream of the second switch remains the same as the output stream of the first switch. The rate at which an ideal Weighted Fair Queueing switch can serve a continuous burst is at most the rate at which that burst arrives, since we assume the instantaneous rate is constant. Since the sending rate remains the same, so does the sending time. So in this case, the first switch, which is the bottleneck, determines the worst case traffic.

In the reverse case, the second switch has the lower guaranteed average rate. In this case, the stream leaving the first switch and the stream leaving the second switch are different. The second switch sends the burst at a lower rate than the first, and therefore spreads it over a longer time. However, the time that the burst is sent is a function only of the size of the burst and the rate of the second switch. As long as a burst of a given size is arriving at the switch faster than the switch will send it, the switch will send the burst for the same amount of time. The product of the sending time and the sending rate of a burst on the output stream must equal the product of those quantities on the input stream, again assuming a constant output rate. (If the rates are not constant a similar statement can be made for the integrals of the sending functions.) That product is the size of the burst. In this case, the first switch is irrelevant; the second switch would transmit the same output stream whether the first switch slows the burst or not. The bottleneck switch completely determines the distortion of the burst.

These three cases of switch behavior are exhaustive. Furthermore, any pair of switches behaves exactly like the bottleneck switch of that pair. We can replace such a two switch combination by the bottleneck switch, and repeat the analysis for the next switch. Repeating this process for an arbitrary number of switches always results in the replacement of all switches by the bottleneck switch in the path. This proves the theorem. □

In an ideal network, Theorem 5 allows us to calculate the effective average rate of the source at the bottleneck switch and use that rate to be sure that the preconditions of Theorem 3 are met at all switches. This ensures DTW stability at all nodes.

4.3.3. Application to Circuit Establishment

Theorem 5 allows each switch to locally calculate the effective average rate seen by switches following it on the circuit's path. This allows the algorithm in Chapter 3 to only pass the negotiated rate and the highest effective rate encountered so far.

Theorem 4 provides the formula used by switches to calculate the effective average rate used in the circuit establishment algorithm described in Chapter 3. As stated in Chapter 4, a sources effective average rate depends on the other sources using the switch. Although it is technically possible for switches to recalculate the effective average rate of all sources whenever they establish a new connection, the complexity of sending the new effective rates of all sources to all adjacent switches is computationally prohibitive. Switches use the fact that the sum of the service shares is at most 1 to compute the worst case effective rate. This removes the dependence on other sources. This worst case effective average rate is:

$$\bar{\lambda}_i(2-\min(1,\frac{s_i\mu}{\hat{\lambda}_i})) \tag{4.23}$$

This is a somewhat pessimistic estimate, but avoids the trap of having to notify every switch in the network when a source wishes to establish a new connection.

If the network providers are willing to take advantage of the fact that random traffic rarely causes the worst case to appear, they can simplify circuit establishment further. Rather than calculating an inflation factor at each switch based on Equation (4.23), each switch can simply inflate the negotiated average rate by a fixed amount between 1 and 2. This removes the guarantee of DTW stability, but allows simple circuit establishment. In most cases random traffic will not cause the worst case to appear, and the network will behave as though it were DTW stable. Using this method is also likely to allow more circuits to be established, thereby utilizing more network resources.

4.3.4. DTW Stability Summary

The three theorems in this section provide the basis for DTW stability in the network. Theorem 3 describes the conditions that must be met for a switch to be DTW stable. Most notable of these is that the sum of the average rates of the sources must be less than the switch service rate. Theorem 4 quantifies the effect of queueing on source traffic, and provides a worst case bound on a source's effective average rate as seen by switches further along a burst's route. This effective average rate allows us to apply Theorem 3 at internal switches. Theorem 5 shows that since WFQ provides a guaranteed average rate to each burst, the bottleneck switch completely determines the effective average rate of the source at any point in the virtual circuit. This allows us to apply Theorem 4 once at the bottleneck switch, rather than repeatedly at each switch. Theorem 5 has the effect of making Theorem 4 an end–to–end bound rather than a per–switch bound.

These theorems are combined in the circuit establishment algorithm in Chapter 3 to assure that any DTW network is DTW stable by construction. A DTW stable network controls congestion without the use of feedback, since congestion times at all switches are bounded.

Chapter 5

Simulation Results

''All that Adam had, all that Caesar could, you have and can do ...

Build, therefore, your own world.''

— *Ralph Waldo Emerson,* **Nature, sec. 4**

Much of the study of Dynamic Time Windows has been based on simulation. This chapter describes several of those simulation studies. First, we verify the analysis of DTW stability presented in Chapter 4. Then we analyze the tradeoffs of source buffering versus network buffering. The DTW feedback system is the subject of the remaining simulations. Those simulations show the effectiveness of the system in a static network, and then demonstrate DTW's ability to adapt to changing network state. Finally, we compare DTW directly to a TCP–style packet feedback control control and to an allocation system based on Leaky Bucket and WFQ.

5.1. Simulation Environment

The simulations that follow use the same source model, which is described here. A source is modelled as a packet generating process that passes bursts of packets to a regulator that enforces the time window criterion. The regulator is a set of buffers and control algorithms that take the raw traffic from the source and smooth it to meet the time window criterion. A regulator may be required to buffer cells until they can be sent. In reality, it may be implemented in an operating system kernel or in a smart network interface.

In the simulations, the entire burst arrives instantly at the regulator for the purposes of determining if there is sufficient space to queue it. When packet delays are computed, the amount of time that the packet must spend queued due to finite line speed is not included, although the additional queueing delay dud to smoothing is. So if a five packet burst arrives to a regulator with room for four packets, one is immediately discarded. However, when computing cell delays, the packets are assumed to have arrived at the peak rate of the source. A burst of packets that arrives at a regulator that has an empty queue and that can send the burst without violating the time window criterion will have no additional delay reported.

This models typical source behavior, namely that the source has performed a computation or fetched a file for transmission and presents a block of data to the network to send. In reality, the rate at which that block can be delivered to the network is limited by the memory or disk bandwidth, but to concentrate on network effects, disk and memory bandwidth is assumed to be infinite in these simulations.

The same model is used to enforce all simulated source controls . The TCP–style packet feedback system and the Leaky Bucket based allocation system are both enforced with regulators. This provides a basis for comparing the restrictions placed on incoming traffic by the various source control systems.

Throughout the simulations in this chapter, we assume that a regulator has finite buffering, and discards packets that it cannot queue. We will see that losses at the regulator are the majority of losses for the systems studied. For this reason, actually implementing a regulator in the manner modelled would be detrimental to any source that is concerned with losses. Such systems would implement a regulator that stopped the sending process before the packets were dropped by the regulator. This is called a loss free regulator. An example of a loss free regulator is one implemented in a source's operating system that implements a blocking write. When all the kernel buffers used for regulation are full and a process tries to send, it is blocked until kernel buffers become free.

Losses at the regulator reduce the number of packets that are sent end–to–end, which reduces the source throughput. A loss free regulator reduces throughput by almost exactly the same amount. The throughput loss of a loss free regulator is caused by the additional time a process spends idle.

We have compared the performance of the lossy and lossless regulators through simulation, and the throughputs and delays are effectively the same. For this reason, we have chosen to report simulations for lossy regulators. It allows us to report the additional metric of losses at the regulator, while preserving the information of how a lossless regulator would perform, *i.e.*, it would have the same delay and throughput, but no losses at the regulator.

Two types of packet generating processes are used in the reported experiments. The first is a throughput maximizing process which sends a packet at the source's peak rate whenever the source control allows This is the same source behavior modelled by the throughput functions defined in Chapter 4.

The other packet generating process is a packet train process. This process sends a cluster of packets at the source's peak rate, and then remains idle for a period of time. See Figure 5.1.

When using this sending process, the number of packets in the packet train is uniformly distributed, and the idle time, or intertrain time, is exponentially distributed. The means of these distributions will be presented in each simulation where packet train sources are used. Since the minimum train length is always one packet, the mean

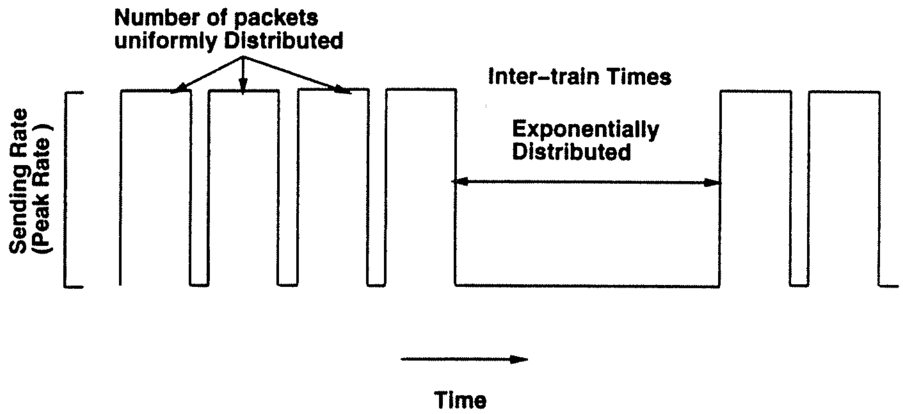

Figure 5.1 : A Packet Train

completely defines the uniform packet train length distributions. The mean burst length is given in the simulations, dividing this value by the 0.008 Mb packet size used in all simulations gives the train length in packets.

Technically, both the throughput maximizing sending process and the packet train sending process send packet trains. The former is deterministic (for a fixed *I*), and the latter is probabilistic. We refer to the deterministic process as the worst case process and the probabilistic process as the packet train process. We refer to a source using a given sending process by the name of the process, *i.e.*, a packet train source is a source using the packet train sending process.

5.2. Global Simulation Parameters

Certain parameters of the simulator remain constant across the simulations. Those that vary will be addressed in the description of each simulation study, while those that are constant are discussed here. These values are also used in all simulations in Chapter 8.

Limits on the memory size of the computers used for simulation required us to use 0.008 Mb (1048 byte) packets, rather than 48 byte cells. The effect of this on the simulations is minimal.

Switches in the simulation have variable amounts of buffering, and a service rate of 1 Gb/sec. Lines between switches and connecting switches to sinks have 1 Gb/sec capacity. Lines from source to switch have a 100 Mb/sec capacity. Delays on all lines are 10 ms. For a circuit passing through two switches to a sink, these values give a bandwidth–delay product of 30 Mb, which is approximately the bandwidth–delay product of a cross–country SONET link.

All switches implement WFQ, even those used by the TCP–like feedback system described later. It has been shown that WFQ improves TCP performance, so using WFQ in this context does not penalize the TCP–like system[66]. In all simulations, sources use equal buffer and service shares with values of 0.05.

The AMTW is 50 secs, primarily so that switches can have relatively fine control over source time windows. Although that is a large theoretical upper bound on network congestion times, in practice, no switch in a simulation of DTW experiences a busy period longer than 2 seconds. Switch MTWs can vary between 10 ms and 10 seconds. All switches use an additive increase factor of 50 ms, and a multiplicative decrease factor of 2. In other words, when there is no congestion over a monitoring interval, switches increase their MTW by 50 ms, and when there is congestion switches divide their MTW by 2. Monitoring intervals are always 1 MTW.

Unless explicitly noted, the time window ratios of the sources are picked randomly from the set {1,2,4,5}. Comparison suites of simulations always use the same r values at sources. This provides a variety of different source time window sizes.

5.3. Circuit Establishment Validation

This section describes simulations that verify the theorems relating to DTW stability presented in Chapter 4. We show that under worst case traffic, the network is stable when Theorem 3 holds at all switches subject to the constraints of Theorems 4 and 5.

In the experiment, we simulate various networks in which DTW stability is guaranteed by the theorems. The networks simulated are increasingly more loaded to the point a network with any more traffic would not be DTW stable. All sources send worst case traffic. We also simulate networks beyond the point at which DTW stability is guaranteed to show that worst case traffic makes switches exhibit DTW instability.

5.3.1. Experimental Method

The network has the configuration shown in Figure 5.2 and the parameters shown in Table 5.1.

The sources entering on the left side of Figure 5.2 are sources that send traffic through both switches. Sources sending from top to bottom are cross traffic sources that only use the switch to which they are adjacent. In this experiment, no feedback is sent, all sources have a ratio of 1, and both switches have an MTW of 0.05 second. Since both switches

Parameter	Value
Switch Buffering	25 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	2 Mb
Congestion Threshold	90%
Time Window Ratios	1

Table 5.1 : Simulation Parameters for Circuit Establishment Validation

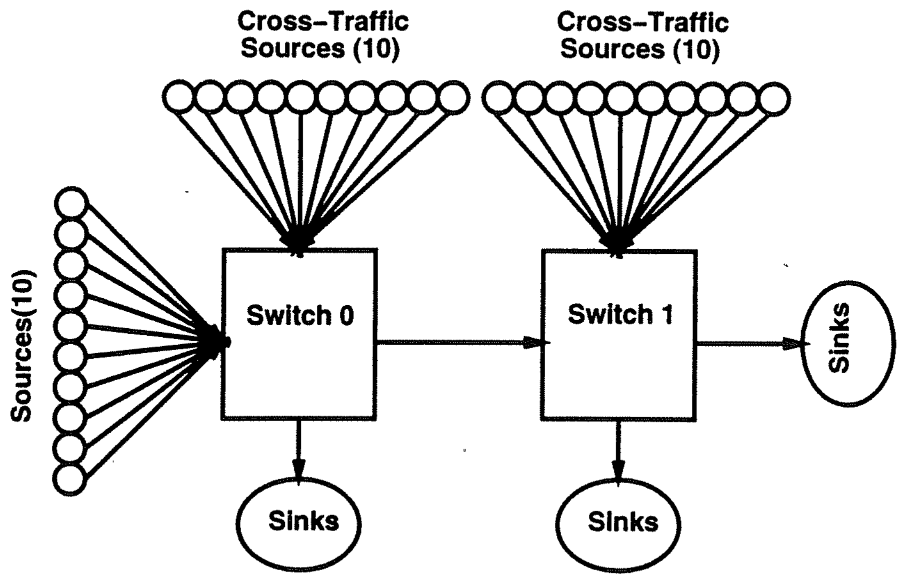

Figure 5.2 : Simulation Configuration for Circuit Establishment Validation

share a static MTW, the effective AMTW of this experiment is the value of that MTW, 0.05 seconds.

The MTW size was chosen to allow the ten sources sending to the first switch to just fill the queue at that switch. While sources are sending at 100 Mb/sec, the switch is eliminating packets at 1000 Mb/sec. Twenty sources sending 2.5 Mb bursts will send at an aggregate 2000 Mb/sec for 25 ms. The switch queue will grow at 1000 Mb/sec for that 25 ms, resulting in a queue length of 25 Mb.

By Theorem 4, each source using both switches has an effective average rate at the second switch of 75 Mb/sec. Applying Theorem 3 at the second switch, it is guaranteed to be stable for up to 5 cross traffic sources. We vary the number of cross traffic sources at switch 1 from one to ten, although ten are shown in Figure 5.2. All sources are sending the worst case traffic described in Chapter 4 in the commentary on Theorem 4. We expect the network to be DTW stable for five or fewer cross traffic sources. In this configuration, a switch which displays a busy period less than 0.05 second is DTW stable.

We simulate the network for each number of cross traffic sources at switch 1, and report the busy periods. These simulations last 1 second.

5.3.2. Results

These results of the simulations are reported in Table 5.2.

The fact that the first switch's maximum busy time is always less than 0.05 seconds indicates that the Theorem 3 holds at the first switch. Notice that the first switch is booked to capacity, *i.e.*, $\sum_{i=1}^{20} \bar{\lambda}_i = \mu$. Any additional traffic at the first switch will violate Theorem 3, and it may exhibit DTW instability.

As predicted by Theorems 3 and 4, the second switch is stable in worst case traffic for five or fewer cross traffic sources. If more than five cross traffic sources are present, the switch quickly violates DTW stability. Although the stability theorems do not predict such a rapid degradation of stability, it is an important result. Allowing sources into a network in violation of the stability theorems can cause significant instability.

Despite the fact that we have demonstrated that worst case traffic in an overbooked network can cause instability, there is evidence that the traffic that causes the network to exhibit this instability occurs infrequently. In this context, an overbooked network is one that does not meet the preconditions of Theorem 3 using effective average rates calculated by Theorem 4 at each switch. To show this, we replaced the worst case sources from our earlier simulation with packet train sources having a mean burst size of 0.8 Mb and a mean intertrain time of 8 ms, but having no correlation or synchronization. We

Cross–traffic Sources at Second Switch	Maximum Busy Period of First Switch (seconds)	Maximum Busy Period of Second Switch (seconds)
10	0.0498	0.905
9	0.0498	0.221
8	0.0498	0:112
7	0.0498	0.0662
6	0.0498	0.0560
5	0.0498	0.0200
4	0.0498	0.0200
3	0.0498	0.0200
2	0.0498	0.0200
1	0.0498	0.0200

Table 5.2 : Summary of busy times at overloaded switchs with worst case traffic

used the exact configuration shown in Figure 5.2, *i.e.*, ten cross traffic sources at each switch, and changed no parameters except the sending source processes, and the duration. We ran this simulation for 500 seconds, the results of which are shown in Table 5.3.

The worst case traffic reported in Table 5.2 that caused the network to exhibit unstable behavior did so very quickly. A violation of DTW stability was recorded in less than a second. The second simulation using packet train sources never violated DTW stability.

Simulations like those reported above support the intuition that alternative methods of computing a source's effective average rate, as suggested in Chapter 4, may be useful.

Maximum Busy Period of First Switch	Maximum Busy Period of Second Switch
0.0246 sec.	0.0448 sec.

Table 5.3 : Summary of busy times of overloaded switches under random traffic

A more aggressive approach to permitting circuit establishment may allow networks to use more resources than under Theorem 4, and display DTW stability.

With few exceptions, the remainder of the simulation studies we will use configurations that are overbooked in terms of the multiple switch stability criteria described in Chapter 4. This is a key point. All the systems discussed here behave much more reliably at lower network utilizations. We describe the systems at the edge of stability because it is critical they function well under those conditions.

5.4. Smoothing Traffic at Sources

A critical element of DTW is sources smoothing their traffic to meet the time window criterion. To do this, they use the regulators defined above. These experiments explore the effect of this regulation on source throughput and source delay.

Source throughput is affected since cells can be lost at the regulator. The regulator buffer size must be large enough that losses are not excessive. In a loss free regulator, losses are tied to the buffering since only a full regulator can suspend the sending process. The other side of this tradeoff is that the more buffering that is placed in the regulator to reduce loss, the higher the potential delay through it.

5.4.1. Experimental Method

The network has the configuration shown in Figure 5.3 and the parameters shown in Table 5.4.

The sources use a packet train process having mean burst sizes of 0.4, 1.6, and 3.2 megabits. The mean intertrain time is the burst size divided by the source peak rate, since the peak rate to average rate ratio is 2:1. Since the ratio is 2:1, a source must be

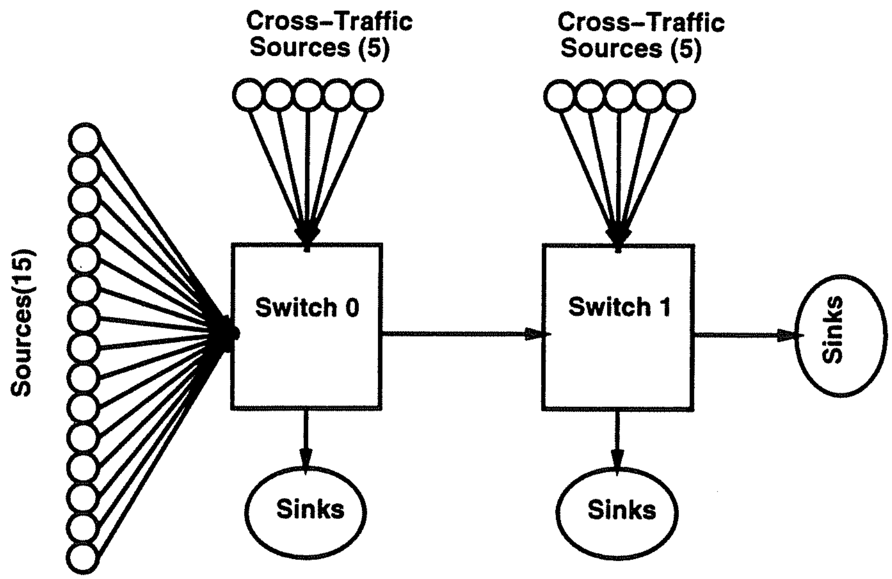

Figure 5.3 : Simulation Configuration for Regulator Experiments

Parameter	Value
Switch Buffering	25 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	varies
Congestion Threshold	90%
Mean Burst Size	varies
Mean Intertrain Time	varies
Duration	500 seconds

Table 5.4 : Simulation Parameters for Regulator Experiments

idle as long as it sends, on the average. In each simulation all sources use the same packet train parameters. We varied the buffer sizes in the regulators from 1 megabit to 100 megabits, although most measurements were taken at the low end of that spectrum. The switches sent feedback normally.

5.4.2. Results

Mean Burst Size = 0.4 Mb			
Buffers (megabits)	Avg. Delay (msec)	Avg. Loss Rate (Mb/sec)	Avg. Throughput (Mb/sec)
0.5	0.0819	7.10	42.9
0.8	0.850	1.14	48.8
1.0	1.3	1.08	48.9
1.6	3.36	0.968	49.0
2.0	5.09	0.885	49.1
3.0	10.6	0.723	49.3
3.2	11.8	0.695	49.3
5.0	29.5	0.554	49.4
10.0	77.0	0.294	49.7
100.0	507.0	0.095	49.9

Table 5.5 : Summary of regulator buffering experiment

Table 5.5, Table 5.6, and Table 5.7 summarize the experiments. The average delay is the average delay of a cell passing through the regulator. The average loss rate represents the loss in source throughput due to the regulator action. This action is either the loss of packets, or the imposed additional idle time, depending on the regulator implementation. The throughput is the end–to–end throughput of the source. Network losses are small enough that this is essentially the throughput out of the regulator.

The results tell us that there is a basic throughput versus delay tradeoff that occurs in the regulator. Although it is theoretically possible for a source to achieve its negotiated throughput under the time window criterion, this is practically limited by the buffering available in the regulator. Furthermore, increasing the buffering in the regulator increases delay, especially for bursty traffic.

For large enough bursts and buffering, the delay through the regulator can be significant. The end–to–end propagation delay is 30 msec in this network, but a source with 10 megabits of buffering experiences an average packet delay of a full round trip time before the packet enters the network in any of the simulations.

Mean Burst Size = 1.6 Mb			
Buffers (megabits)	Avg. Delay (msec)	Avg. Loss Rate (Mb/sec)	Avg. Throughput (Mb/sec)
0.5	0.0800	35.7	14.3
0.8	0.0800	28.1	21.9
1.0	0.0800	23.6	26.4
1.6	0.0858	12.5	37.5
2.0	0.204	7.05	42.9
3.0	6.73	2.75	47.2
3.2	7.84	2.60	47.3
5.0	19.9	2.11	47.7
10.0	64.0	1.22	48.8
100.0	842.7	0.159	49.8

Table 5.6 : Summary of regulator buffering experiment

Mean Burst Size = 3.2 Mb			
Buffers (megabits)	Avg. Delay (msec)	Avg. Loss Rate (Mb/sec)	Avg. Throughput (Mb/sec)
0.5	0.0800	43.2	7.57
0.8	0.0800	38.9	11.9
1.0	0.0800	36.1	14.6
1.6	0.0800	28.5	22.2
2.0	0.0801	24.0	26.8
3.0	1.0869	14.3	36.5
3.2	1.54	12.7	38.1
5.0	12.6	4.99	45.8
10.0	60.3	2.92	47.9
100.0	1322.2	0.876	49.9

Table 5.7 : Summary of regulator buffering experiment

Any system that shapes traffic suffers from the regulation tradeoff described above. Our simulations of Leaky Bucket control and TCP control exhibit similar losses and delays at the regulator. A regulator will inevitably disrupt traffic, but it is the changes that such a regulator makes to the traffic stream that prevent similar problems in the network itself. The cost of the regulator must justify itself by the positive effects in the network. Under DTW these positive effects include DTW stability, which allows the system to react well to congestion even in the presence of changing network state, and the service tailoring discussed in Chapters 7 and 8. Although we would prefer a system where the shaping could be performed transparently, this is simply impossible.

Throughout this chapter, simulation summaries for all systems note the loss and delay contributed by regulators.

Although distortion of traffic due to the regulator is inevitable, this distortion need not be catastrophic. As we mentioned earlier, loss free regulators can be used for traffic that requires them. Many operating systems' networking code implements a loss free regulator. Furthermore, the tradeoff between packet delay and throughput in the regulator is simple and direct. Sources that require high throughput require more buffering at the regulator; sources seeking low delay require less.

5.5. DTW Feedback System

This section explores the operation of DTW's feedback system in both static and dynamic network configurations. A static network configuration is one in which no source alters its behavior during the simulation. Dynamic configurations are characterized by sources becoming idle or becoming active. DTW controls and avoids congestion in both environments.

5.5.1. DTW in a Static Network

This set of simulations demonstrates that DTW's feedback system responds to network congestion, and that, in the absence of radical changes in network state, the system remains stable in a traditional sense.

In this stable state, the time windows of the sources oscillate about a fixed point. Source throughput is high and end–to–end delay is low. The oscillation of the time windows represents DTW's constant probing to determine if the network can accommodate additional burstiness. Since we assume traffic is inherently unpredictable, the way DTW determines if the network can support additional burstiness is to have each switch periodically increase the burstiness sources using it are allowed, by increasing their MTW and observing the results of this change. When a change results in congestion, the switch's MTW is reduced.

5.5.1.1. Experimental Method

The network has the configuration shown in Figure 5.4 and the parameters shown in Table 5.8.

5.5.1.2. Results

Table 5.9 summarizes the simulation. Only statistics for the 15 sources that use both switches are reported. The "Delay" columns represent the per–packet average delay through the regulator and from the regulator to the sink, respectively. The "Total Delay" column is their sum. Network delay includes propagation delay (30 ms) and queueing delay. Similarly, the "Loss" columns represent the average percentage of packets lost in the regulator and in the network by each source. The "throughput" column is the average end–to–end throughput of each source.

Parameter	Value
Switch Buffering	25 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	1.6 Mbits
Congestion Threshold	90%
Mean Burst Size	0.8 Mb
Mean Intertrain Time	8 ms
Duration	500 seconds

Table 5.8 : Simulation Parameters for Feedback Behavior Experiments

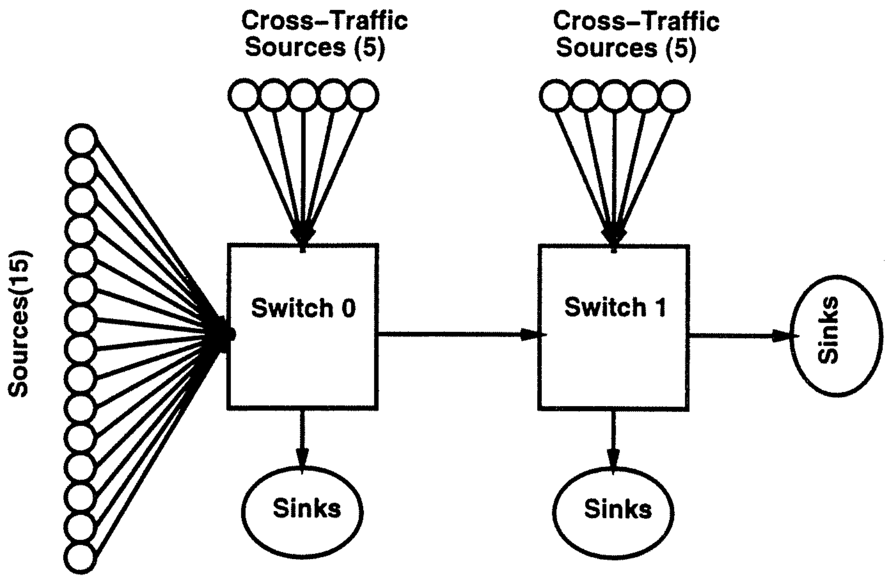

Figure 5.4 : Simulation Configuration for Feedback Behavior Experiments

Src. Delay (msec)	Net. Delay (msec)	Total Delay (msec)	Src. Loss (pct.)	Net. Loss (pct.)	Throughput (Mb/sec)
2.88	33.7	36.6	4.11	0.000253	47.9

Table 5.9 : Summary of Simple Feedback Simulation

Most losses occur at the regulator, but can be avoided by a loss free regulator. DTW avoids network losses well, and maintains a 95% utilization of the switch. These results will be compared with other systems later in the chapter. They are presented here to show that DTW controls congestion while maintaining reasonable values of several figures of merit.

We repeated the simulation using the configuration in Table 5.4 and the parameters in Table 5.8, with the exception of the source sending process. This simulation uses sources sending worst case traffic, *i.e.*, maximum sized bursts as fast as the source control allows them.

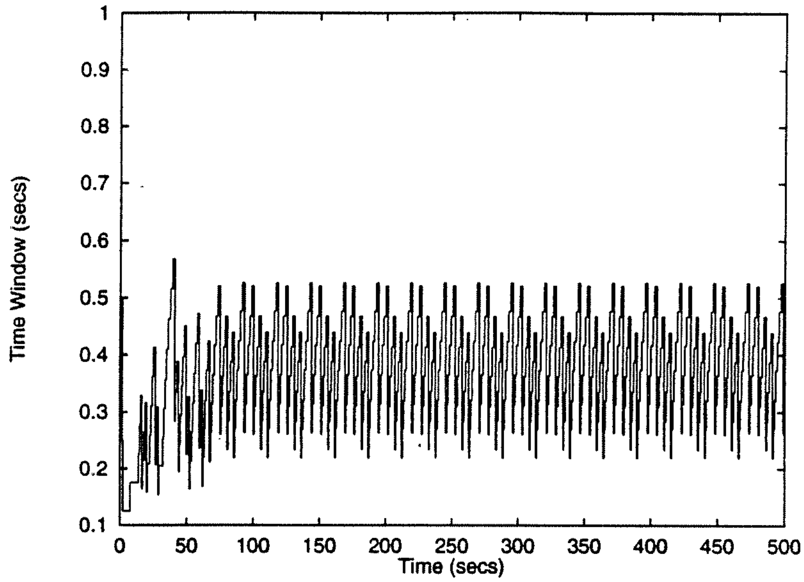

Figure 5.5 : Time Window versus Time (Worst Case Sources)

Figure 5.5 shows the evolution of the time window for a source with a ratio of 1 during the simulation of worst case traffic. Since the source's ratio is 1, the time window plotted represents the minimum of the two switch MTWs. Figure 5.5 shows the periodic behavior of a source's time window as it seeks the appropriate value for these network conditions. After the simulation achieves a steady state, the source time window oscillates regularly. This regular period shows that DTW achieves a steady state in a static network.

Figure 5.6 shows a the time window of a source in a network with the same configuration as the simulation plotted in Figure 5.5, populated by packet train sources. This simulation uses all the parameters in Table 5.8. The source whose time window is plotted in Figure 5.6 also has a ratio of 1. The time window displays periodicity, but not exact periodicity. Since there are random fluctuations in the traffic, the oscillations are of somewhat differing size. These fluctuations in the periodicity of the time window

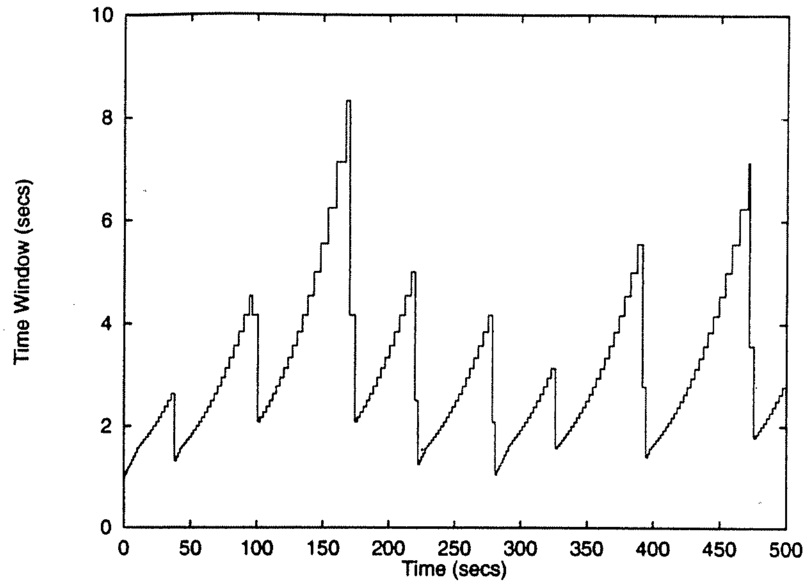

Figure 5.6 : Time Window versus Time (Packet Train Sources)

display DTW's sensitivity to variations in source traffic.

The time window plotted in Figure 5.6 is generally larger that the one plotted in Figure 5.5. Packet train sources have different idle times between their packet trains, and their bursts are of varying length. The bursts of packet train sources are less likely to arrive together at switches than those of the worst case sources. When the packet train source's bursts arrive together, they are less likely to be maximum length bursts. The result of this is a higher degree of stochastic multiplexing at the switches, reflected in a higher time window.

5.5.2. DTW and the Congestion Threshold

These simulations demonstrate the effect of the congestion threshold on DTW's performance. The congestion threshold is the level of buffer occupancy at which a switch considers itself congested, and reduces the MTW of sources using it. The congestion threshold is given as a fraction of a switch's buffers. The following simulations vary the congestion threshold, leaving all other parameters fixed.

5.5.2.1. Experimental Method

The network has the configuration shown in Figure 5.7 and the parameters shown in Table 5.10. The threshold is always the same at both switches.

5.5.2.2. Results

The results of the threshold adjustment experiments are given in Table 5.11. The experiments show the effect of a range of threshold values. The values cover a wide range of switch behavior. A switch using a threshold of 0.5 will reduce its MTW when half of its buffers are in use, while a switch with a 1.0 threshold will not reduce its MTW until losses occur.

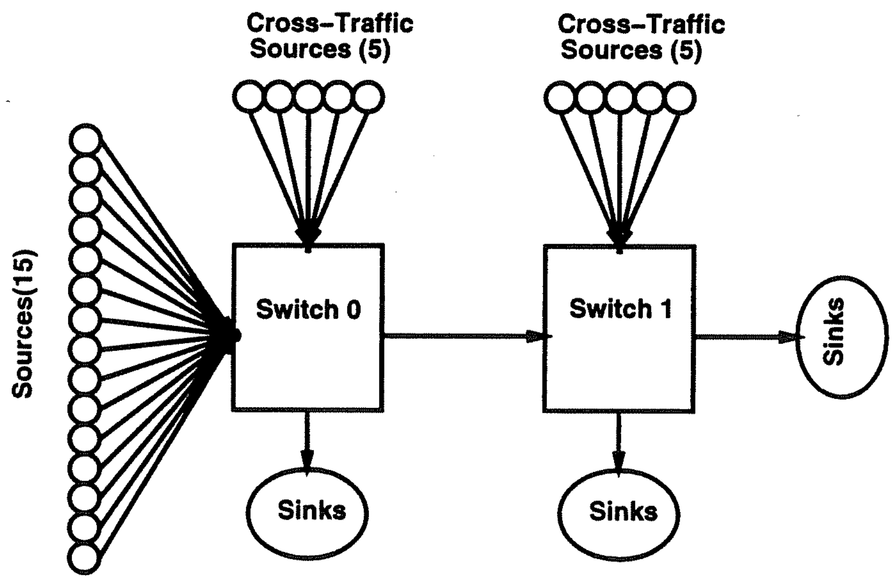

Figure 5.7 : Simulation Configuration for Threshold Study Experiments

Parameter	Value
Switch Buffering	25 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	3.2 Mbits
Congestion Threshold	varies
Mean Burst Size	1.6 Mb
Mean Intertrain Time	16 ms
Duration	500 seconds

Table 5.10 : Simulation Parameters for Threshold Study

Thresh–hold	Source Delay (msec)	Network Delay (msec)	Total Delay (msec)	Source Losses (pct.)	Network Losses (pct.)	Through-put (Mb/sec)
0.50	11.2	33.1	44.3	7.40	0	46.5
0.60	10.3	33.5	43.8	6.94	0	46.8
0.70	9.49	34.0	43.5	6.49	0.000154	46.9
0.80	8.75	34.4	43.2	6.10	0.000638	47.1
0.90	8.28	34.7	43.0	5.79	0.00285	47.3
1.00	7.65	35.1	42.8	5.44	0.00909	47.4

Table 5.11 : Results of Threshold Study

Table 5.11 shows that source delays and source losses decrease with increasing threshold. A higher threshold results in larger time windows at the sources, and therefore less smoothing delays in the regulator. Conversely, queueing delays, reflected in the network delay, increase with increasing threshold. Queueing delay rises since the average queue occupancy in switches is higher. The decrease in source delay is greater than the increase in queueing delay, so total delay is reduced by increasing the threshold. This can be seen graphically in Figure 5.8.

Increasing the threshold results in additional network losses. Each increase of 0.1 in the threshold results in roughly a five fold increase in network losses. Network losses may be more costly to a source than source losses, and the needs of sources that require low network losses must be considered when setting a threshold value for the system. The reduction in delay gained by increasing the threshold is paid for by an increase in

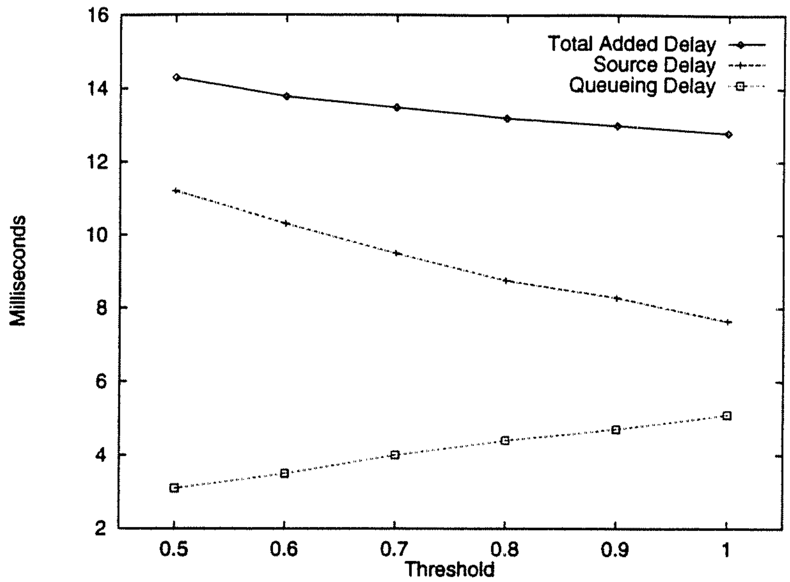

Figure 5.8 : Delay Components vs. Threshold

network losses.

Chapters 7 and 8 describe methods of using the congestion threshold in the context of service tailoring to affect the network performance of individual sources.

5.5.3. DTW in a Dynamic Network

The simulations described above show that DTW runs well in the absence of changes in network state, while remaining sensitive to changes in that state. To be certain that the system could accommodate sources entering and leaving the network, we simulated several sources entering a network with the capacity to accommodate them.

5.5.3.1. Experimental Method

The network has the configuration shown in Figure 5.9 and the parameters shown in Table 5.12. The shaded sources are all idle initially, and begin sending 200 seconds into the simulation. At the beginning of the simulation, the bottleneck switch will be the second switch, and sources using both switches will derive their time window from it. When the five additional sources enter, the first switch becomes the bottleneck, and the sources begin deriving their time windows from it. This change in time window size represents DTW detecting the new source and avoiding congestion in the new configuration.

Table 5.13 summarizes a typical simulation from this experiment. These are the same parameters reported in Table 5.9. All the values reported here are close to the values given in Table 5.9, which shows that the source addition does not unbalance the system significantly.

In fact, throughput is higher and source losses are lower than those in Table 5.9, primarily because for the first forty percent of the experiment the network is running with less traffic. This allows the sources to maintain larger time windows, and therefore losses and delays at the regulators are lower. Notice that network losses are somewhat higher, reflecting the losses incurred in the network when the idle sources begin transmitting. The five idle sources begin transmitting at the same time, which is a particularly

Parameter	Value
Switch Buffering	25 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	1.6 Mbits
Congestion Threshold	90%
Mean Burst Size	0.8 Mb
Mean Intertrain Time	8 ms
Duration	500 seconds

Table 5.12 : Simulation Parameters for Adding Sources

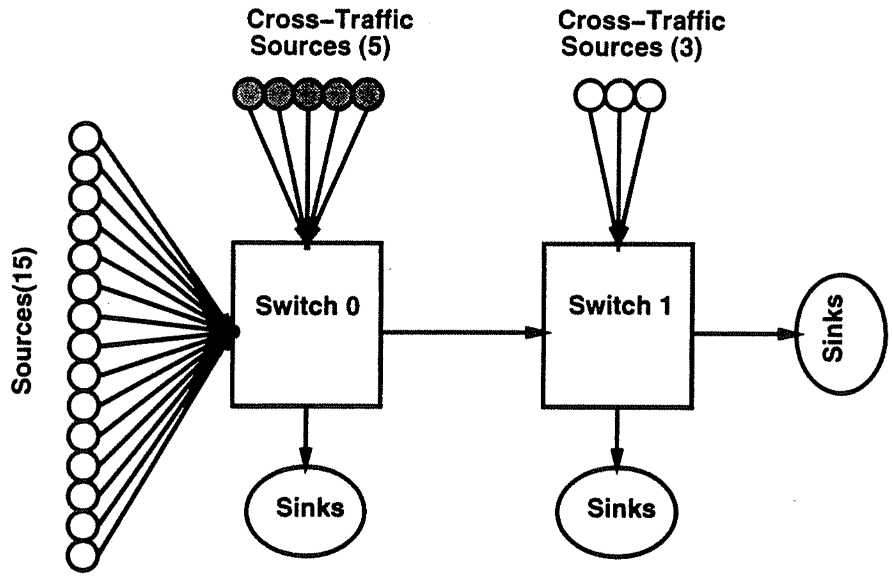

Figure 5.9 : Simulation Configuration for Adding Sources

Src. Delay (msec)	Net. Delay (msec)	Total Delay (msec)	Src. Loss (pct.)	Net. Loss (pct.)	Throughput (Mb/sec)
2.67	31.8	34.5	3.86	0.00108	48.1

5.5.3.2. Results

Table 5.13 : Summary of Source Addition

sudden change in the network state.

Figure 5.10 shows a plot of the time window of one of the sources using both switches. Figure 5.11 is a plot of the time windows of one of the cross traffic sources from each switch. The sources plotted have ratios of 1, so Figure 5.11 is a plot of the MTWs of the two switches. Initially, the second switch is the bottleneck switch, and the time window of the source using both switches follows that time window. When the new sources enter the network, the first switch becomes the bottleneck, and the source in Table 5.10 follows the new bottleneck switch's MTW. This is the exact behavior predicted, and Table 5.13 indicates that it causes the system to behave well.

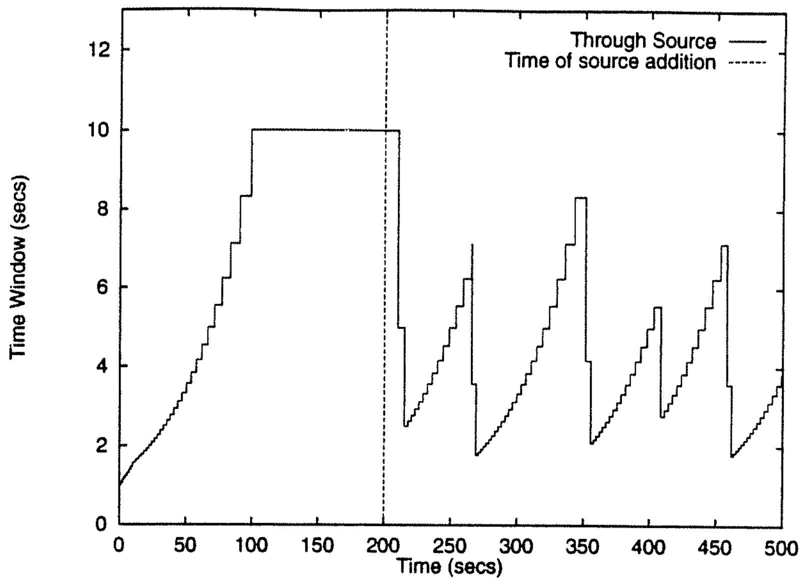

Figure 5.10 : Time Windows for Through Sources During Source Addition

We have simulated the deletion of sources from the network as well, but the results are essentially the same as for addition in reverse. The former bottleneck source becomes less constrained, and the other switch becomes the bottleneck, forcing the through traffic's time windows to follow the pattern of the new bottleneck.

These simulations, and those of the previous section, have demonstrated the function of the DTW feedback algorithms. They have shown how the switches adjust source time windows in response to traffic in a static network, and sources entering the network. The following section presents a more quantitative evaluation of DTW's performance by comparison with other congestion control and avoidance systems.

5.6. Comparison to Other Systems

This section compares DTW to two other systems, a feedback congestion control using packet windows, and an allocation system based on Leaky Bucket and WFQ. The intent is to compare DTW's congestion avoidance and control mechanisms to these

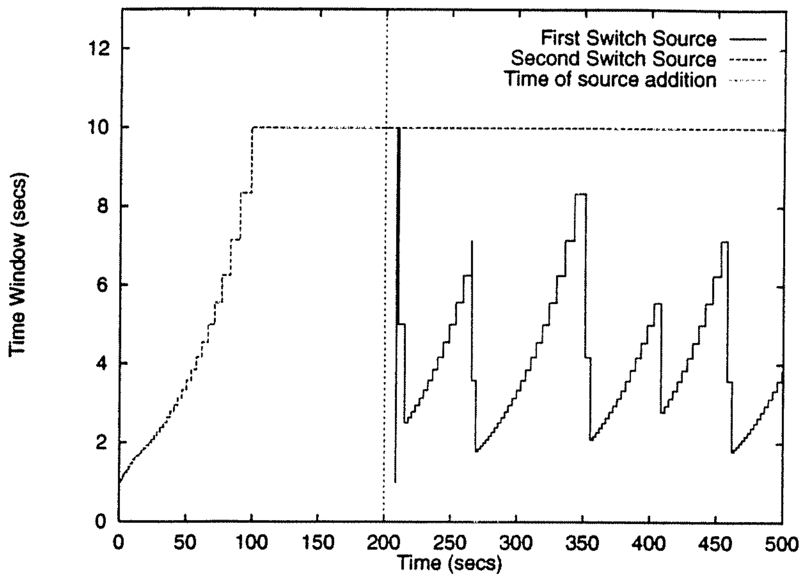

Figure 5.11 : Time Windows for Cross Traffic Sources During Source Addition

systems, so none of the experiments described in this section make use of the features for service tailoring described in Chapter 7. They are simulated in Chapter 8. This section explores the benefits and drawbacks of DTW compared to other systems for congestion control and avoidance.

The packet feedback system to which we compare DTW is closely modelled on TCP, and uses TCP's methods to detect network congestion. The number of unacknowledged packets that a source can have outstanding at any time is its packet window size. Under the feedback system, each source reacts to congestion by multiplicatively decreasing the size of its packet window. When a source has received acknowledgements for a full window, it additively increases its packet window. A source detects congestion whenever it fails to get an acknowledgement of a packet within a preset time, which is twice the round trip time in these simulation.

This feedback system also employs a Slow Start mechanism as congestion control[8]. Each source defines two window sizes, one is the congestion window size, which represents the number of outstanding packets allowed when the source is operating normally, and the other is the Slow Start window size, which is the number of packets that a source can have outstanding during a Slow Start. The smaller of these two quantities is the packet window size the source uses. In general they are equal. When congestion is detected, the congestion window is reduced multiplicatively, and the Slow Start window is set to 1 packet. The Slow Start window is increased by 1 for every acknowledged packet until the Slow start window is equal to the congestion window. When the two are equal the linear increase/multiplicative decrease algorithm is used to adjust both.

The net effect of Slow Start and feedback congestion avoidance is that a source linearly increases its packet window until congestion is detected. Then the congestion window is multiplicatively decreased, and the Slow Start window is reduced to its smallest size and increased exponentially until it reaches the value to which the congestion window was multiplicatively decreased. This period of reduced throughput enforced by Slow Start gives congested switches a chance to clear their queues.

We chose to study a system closely modelled on TCP rather than a system like DECbit[2] that receives feedback directly from switches because the former is more widely used. In fact, the TCP congestion avoidance and control systems are arguably the most widely used in the world, and we felt it would be instructive to compare DTW to them.

Although TCP is the model for the feedback system we simulate, the feedback system is not TCP. Notably, no retransmissions occur, and the acknowledgements are sent only to allow the sources to detect congestion. The throughput values reported reflect how many packets traversed the network, not whether they had useful data in them. We simulate only the congestion control and avoidance mechanisms of TCP, not its reliable delivery system.

The allocation strategy modelled is one where sources are policed by Leaky Bucket and send packets into a network of WFQ switches. The Leaky Bucket sources we use are equivalent to the DTW sources, where the equivalence is defined by equation (4.9). The allocation system does not attempt to sense network state in any way.

We chose Leaky Bucket as a representative allocative enforcement mechanism since it is often proposed as a congestion control scheme for ATM traffic[76]. For example, the ATM forum suggests it as a policing mechanism[17].

5.6.1. DTW vs. Classic Feedback (Static Network)

In order to compare DTW to the TCP–like packet feedback system (called ''the feedback system''), we simulated the two using identically distributed traffic in a static network and recorded several figures of merit. Again, no retransmissions were sent, although acknowledgements were sent for use by the feedback system in determining the congestion window size. Queueing delay for the acknowledgements was neglected, and they were never lost.

5.6.1.1. Experimental Method

The network has the configuration shown in Figure 5.12 and the parameters shown in Table 5.14.

The feedback system's window was incremented by 10 packets (0.08 Mb) every time a full window was successfully acknowledged. When the feedback system detected

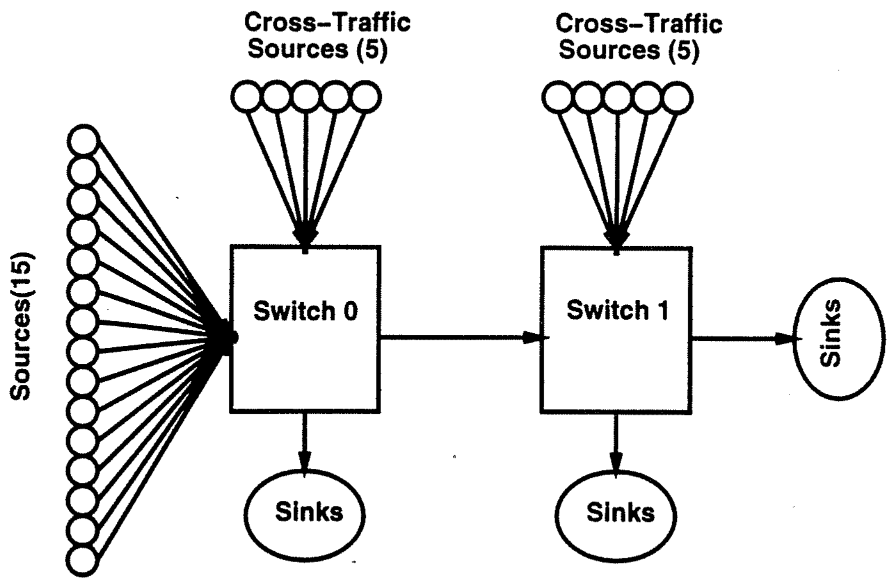

Figure 5.12 : Simulation Configuration for DTW vs. Feedback Experiments

Parameter	Value
Switch Buffering	10-25 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	twice the burst size
Congestion Threshold	90%
Mean Burst Size	0.2 Mb - 6.4 Mb
Mean Intertrain Time	2 ms - 64 ms
Duration	500 seconds
TCP Window Increase	10 packets/window
TCP Window Decrease	0.5

Table 5.14 : Simulation Parameters for DTW vs. Feedback Experiments

congestion it cut its window to half its previous size, and initiated a Slow Start as described above.

5.6.1.2. Results

Table 5.15 and Table 5.16 summarize the simulations. The tables report the same parameters as Table 5.9.

Burst Size (Mb)	Source Delay (msec)	Network Delay (msec)	Total Delay (msec)	Source Losses (pct.)	Network Losses (pct.)	Through-put (Mb/sec)
0.2	0.350	32.7	33.1	1.64	0	49.2
0.4	0.844	33.7	34.5	2.31	0	48.9
0.8	2.60	34.2	36.8	3.71	0.00154	48.2
1.6	8.28	34.7	43.0	5.79	0.00284	47.3
3.2	27.9	35.2	63.1	9.15	0.00383	46.6
6.4	73.4	35.7	109.1	10.6	0.0190	45.8

Table 5.15 : DTW Summary (Switch Buffering = 25 Mb)

Burst Size (Mb)	Source Delay (msec)	Network Delay (msec)	Total Delay (msec)	Source Losses (pct.)	Network Losses (pct.)	Through-put (Mb/sec)
0.2	0.129	44.6	44.7	0.308	0.0234	49.9
0.4	0.342	43.3	43.6	0.712	0.0438	49.6
0.8	1.23	42.1	43.3	1.39	0.0884	49.3
1.6	5.68	40.3	46.0	3.02	0.145	48.5
3.2	19.8	41.1	60.9	4.82	0.173	48.4
6.4	65.5	36.9	102.4	9.06	0.250	46.5

Table 5.16 : Packet Feedback Summary (Switch Buffering = 25 Mb)

DTW gives sources a shorter average end–to–end packet delay than the feedback system, as well as lower network losses by at least an order of magnitude. The throughput figures do not reflect any retransmissions, so sources that have to retransmit will lose more effective throughput to these losses. Since DTW sources' average throughput is more than 96% of the feedback system sources' throughput, the difference made by network losses may be telling for sources requiring reliable delivery.

DTW has higher source losses and delays than the feedback system. However, that performance is offset by the superior performance of DTW traffic in the network. Source losses can generally be avoided by a loss free regulator, and are comparable to the feedback system's source losses. The higher source delay of DTW traffic is easily offset by the improved network performance, until burst sizes become so great that source delays dominate the total delay for both systems.

We repeat the simulations for switches with 10 Mb of buffering, to study the effect of switch buffering on the network. We expect that reducing it will increase the effect of the bandwidth–delay product on the systems. With 25 Mb of buffering, the first switch can buffer a full pipe of traffic from all the sources the first switch (20 sources × 100 Mb/sec/source × 10 msec = 20 Mb). Reducing this buffering affects both systems, but we expect it to be more disturbing to the purely reactive feedback system.

Table 5.17 and Table 5.18 summarize simulations with the parameters in Table 5.14, except that the switches have 10 Mb of buffering.

Burst Size (Mb)	Source Delay (msec)	Network Delay (msec)	Total Delay (msec)	Source Losses (pct.)	Network Losses (pct.)	Through-put (Mb/sec)
0.2	0.482	31.8	32.3	2.43	0.000658	48.8
0.4	1.51	31.9	33.4	4.24	0.00155	48.0
0.8	4.57	32.0	36.6	6.24	0.00583	47.0
1.6	13.8	32.3	46.1	8.53	0.0139	46.1
3.2	37.8	32.6	70.4	10.8	0.0660	45.8
6.4	84.9	32.8	117.7	11.6	0.130	45.4

Table 5.17 : DTW Summary (Switch Buffering = 10 Mb)

Burst Size (Mb)	Source Delay (msec)	Network Delay (msec)	Total Delay (msec)	Source Losses (pct.)	Network Losses (pct.)	Through put (Mb/sec)
0.2	0.377	34.3	34.7	1.85	0.0626	49.1
0.4	1.46	33.6	35.1	3.96	0.113	48.0
0.8	4.58	33.2	37.8	5.8	0.137	47.2
1.6	16.2	32.6	48.8	10.7	0.216	45.1
3.2	35.9	33.0	68.9	10.7	0.214	45.8
6.4	123.9	31.5	155.4	22.3	0.319	41.5

Table 5.18 : Packet Feedback Summary (Switch Buffering = 10 Mb)

The most telling result of this experiment is that the throughputs of the systems are within a half percent for most cases, while DTW's network losses remain nearly a factor of ten lower. Sources retransmitting lost traffic will behave much better under DTW.

Since there is less buffering in the network, congestion occurs at lower throughput and burstiness levels. This causes the source control algorithms to be more active, which restricts traffic more, and makes losses at the regulator more likely. Although these losses can be eliminated by a loss free regulator, their effect on throughput is notable.

As expected, without considerable switch buffering, feedback based congestion control suffers under the high bandwidth delay product of the simulated network.

5.6.2. DTW vs. Classic Feedback (Dynamic Network)

The previous simulations studied the two congestion control systems in a static network. It is also important to see how well a congestion control system adapts to a changing network state. To investigate this, we simulate a less loaded network than in the previous section, and then repeat the simulation for the same network with several sources entering and leaving repeatedly. We summarize how much the destabilizing sources affect the two systems.

5.6.2.1. Experimental Method

The network has the configuration shown in Figure 5.13 and the parameters shown in Table 5.19. The shaded sources are idle for the initial simulation of the less loaded network. In the second simulation, they are idle for fifteen seconds and then send at their full rate for five seconds, and repeat the cycle. We report the percentage degradation in the figures of merit between the two simulations for a range of burst sizes.

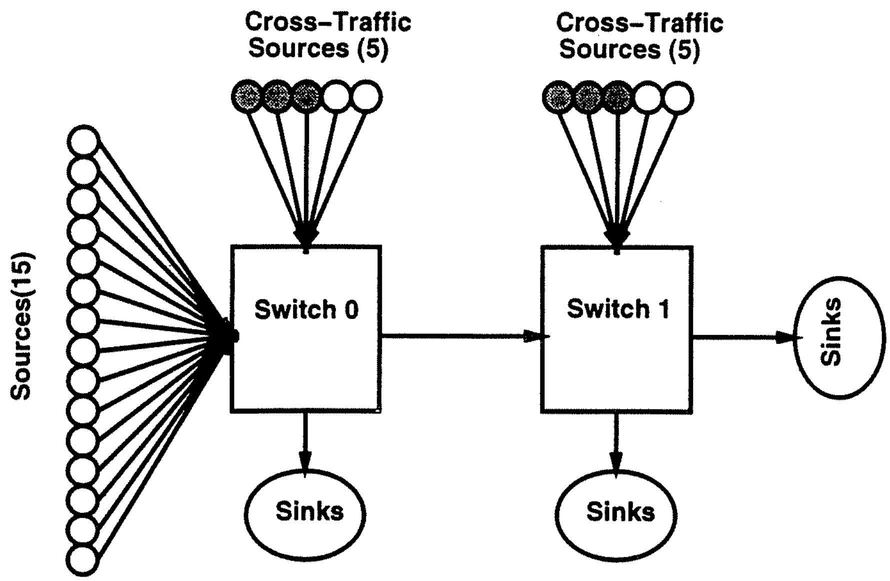

Figure 5.13 : Simulation Configuration for Changing Network Experiments

Parameter	Value
Switch Buffering	10 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	twice the burst size
Congestion Threshold	90%
Mean Burst Size	0.2 Mb-1.6 Mb
Mean Intertrain Time	2 ms-16 ms
Duration	500 seconds
TCP Window Increase	10 packets/window
TCP Window Decrease	0.5

Table 5.19 : Simulation Parameters for Changing Network Experiments

5.6.2.2. Results

Table 5.20 and Table 5.21 summarize the effect that introducing these destabilizing sources had on each system.

Each column represents the percentage increase in the named performance measure from the unloaded to the loaded network. We report the increases to show how each

Burst Size (Mb)	Change in Source Delay (pct.)	Change in Network Delay (pct.)	Change in Total Delay (pct.)	Change in Source Losses (pct.)
0.2	31.9	1.57	1.92	41.2
0.4	63.8	1.61	3.24	67.3
0.8	93.2	1.54	7.47	84.8
1.6	119.2	1.23	19.5	94.5

Table 5.20 : Perturbation of DTW in an unstable network

Burst Size (Mb)	Change in Source Delay (pct.)	Change in Network Delay (pct.)	Change in Total Delay (pct.)	Change in Source Losses (pct.)
0.2	55.3	4.16	4.30	932.0
0.4	330.1	3.15	4.06	2978.0
0.8	1162.3	2.64	6.08	4592.8
1.6	3181.3	1.98	13.2	8158.4

Table 5.21 : Perturbation of the feedback system in an unstable network

system reacts to repeated, significant changes in network state.

Table 5.22 and Table 5.23 summarize the changes in network loss of the two systems.

Both systems avoid loss almost completely in the stable network, but in the dynamic configuration losses occur. The DTW network experiences an order of magnitude less losses. An increase in network losses combined with a loss in source

Avg. Burst Size (Mb)	Unloaded Losses (pct)	Loaded Losses (pct)
0.2	0	0.0116
0.4	0	0.0101
0.8	0	0.0277
1.6	0	0.140

Table 5.22 : Summary of DTW Losses in a Dynamic Network

Avg. Burst Size (Mb)	Unloaded Losses (pct)	Loaded Losses (pct)
0.2	0	0.204
0.4	0	0.417
0.8	0	0.536
1.6	0.0339	1.008

Table 5.23 : Summary of Feedback System Losses in a Dynamic Network

throughput due to increasing source losses, means that the feedback system will be unattractive to sources that require reliable delivery. DTW reacts to an changing network with a much smaller degradation of performance in these areas.

As expected, the feedback system suffers under the rapid changes. Since the feedback system is a purely reactive system facing a high bandwidth–delay product, a violently changing system is particularly disturbing for it. The feedback system is ineffective because too much traffic enters the network before the feedback control can take corrective action. Increasing the burstiness of the sources only makes the disturbance greater, since a larger burst represents a larger change in network state to which the system must react in a fixed time.

As the mean burst size increases, DTW degrades much more gracefully than the feedback system. This is due to the fact that DTW's congestion control system is not reactive. It suffers a performance degradation since its reactive avoidance mechanism has similar problems to those of the feedback system, but that degradation is kept in check by the non–reactive congestion control (DTW stability).

In summary, the DTW system is less prone to losses due to changes in network state than the feedback based system. This is due to the integration of the feedback and packet admission system that enforces DTW stability at the switches.

5.6.3. DTW vs. Allocation (Fully Loaded Network)

To compare DTW and the allocation system, in which source traffic is policed by Leaky Bucket, we performed a set of experiments similar to those performed on the feedback system. We simulated equivalent Leaky Bucket and DTW systems for a range of sending processes.

5.6.3.1. Experimental Method

The network has the configuration shown in Figure 5.14 and the parameters shown in Table 5.24.

The Leaky Bucket sources were given the parameters of equivalent DTW sources from the comparison simulations. The parameters of the Leaky Bucket sources were derived

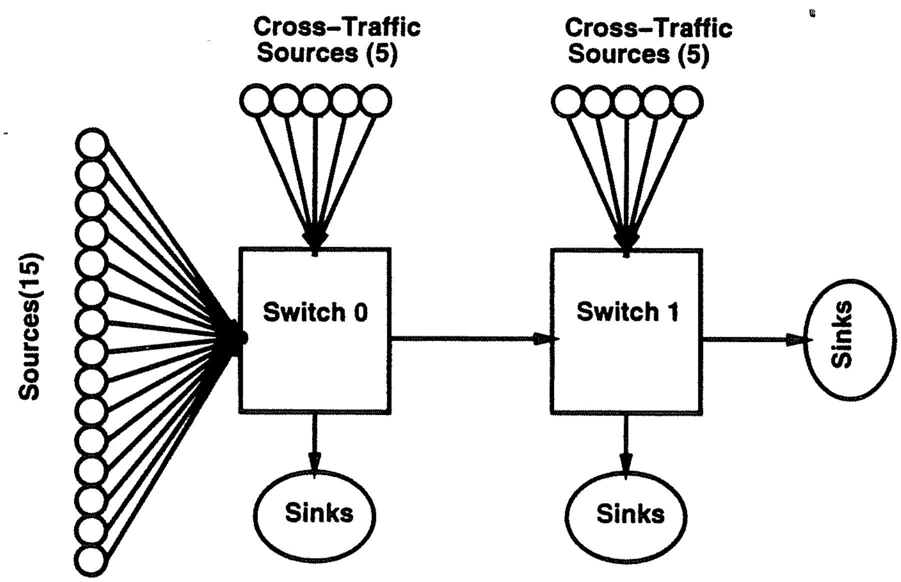

Figure 5.14 : Simulation Configuration for DTW vs Allocation Experiments

Parameter	Value
Switch Buffering	10-25 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	twice the burst size
Congestion Threshold	90%
Mean Burst Size	0.2 Mb-6.4 Mb
Mean Intertrain Time	2 ms-64 ms
Duration	500 seconds

Table 5.24 : Simulation Parameters for DTW vs Allocation Experiments

using equation (4.9) using a time window of 1 second multiplied by each source's ratio. This provides a range of Leaky Bucket parameter values. The same source ratios are used here as were used by sources in the simulations reported in Table 5.15 and Table 5.17.

5.6.3.2. Results

The simulations are reported in Table 5.25, for switch buffers of 25 Mb. Comparing this summary with that in Table 5.15, we see that DTW reports significantly smaller network losses, and lower total delays. DTW's throughputs are somewhat lower, but this is largely compensated for by the lower network losses especially for sources sending

Burst Size (Mb)	Source Delay (msec)	Network Delay (msec)	Total Delay (msec)	Source Losses (pct.)	Network Losses (pct.)	Through-put (Mb/sec)
0.2	0.166	38.8	39.0	0.361	0.0137	49.8
0.4	0.415	39.2	39.6	0.701	0.0151	49.6
0.8	1.38	38.7	40.1	1.33	0.0383	49.2
1.6	4.88	38.7	43.6	2.52	0.0528	48.8
3.2	18.5	39.3	57.8	4.95	0.0960	48.3
6.4	50.9	38.2	89.1	6.86	0.120	47.6

Table 5.25 : Allocation (Leaky Bucket) Summary (Switch Buffering = 25 Mb)

reliable data streams.

Table 5.26 Summarizes a similar set of experiments with switch buffers set to 10 Mb. Decreasing the switch buffering continues the trends seen in Table 5.25. In general DTW serves traffic better in the two scenarios. However, once source delays become the dominant factor in end–to–end delays, Leaky Bucket shows improvement over DTW. Since Leaky Bucket's source control is independent of the state of the network, it does not become more restrictive in the more congested networks, like DTW's control does.

That there is no attempt by allocation systems to sense network state is a major difference between them and DTW. It is possible to pick different bucket sizes for the Leaky Bucket sources and achieve similar or better performance to DTW in the simulations we have shown, but such tuning needs to take into account the traffic profile. DTW senses the traffic profile and seeks the appropriate source control parameters.

Leaky Bucket's source control must be configured for the worst case when a source establishes its virtual circuit. If the Leaky Bucket parameters are chosen too optimistically, sources may lose data if they send traffic that is too bursty. If the parameters are

Burst Size (Mb)	Source Delay (msec)	Network Delay (msec)	Total Delay (msec)	Source Losses (pct.)	Network Losses (pct.)	Through-put (Mb/sec)
0.2	0.166	35.9	36.1	0.361	0.148	49.7
0.4	0.415	36.1	36.5	0.701	0.271	49.5
0.8	1.39	35.8	37.2	1.34	0.472	49.0
1.6	4.89	35.7	40.6	2.52	0.738	48.4
3.2	18.7	35.7	54.4	4.95	1.08	47.9
6.4	51.0	35.0	86.0	6.86	1.12	47.1

Table 5.26 : Allocation (Leaky Bucket) Summary (Switch Buffering = 10 Mb)

chosen too pessimistically, the network may be underutilized. In either case this choice is made without knowledge of the traffic that will be in the network, or even the number of sources currently active. DTW senses network state and adapts to it.

5.6.4. DTW vs Allocation (Underloaded Network)

Since Leaky Bucket is tuned to the worst case, we expect it to perform well in loaded networks. However it restricts sources sending in less loaded networks as much as if they were sending in a loaded one. An advantage of DTW is that it senses this available capacity, and allows sources to utilize it. The following simulations are of a less lightly loaded network, which are compared with the equivalent fully loaded networks (those that produced Table 5.25 and Table 5.15). To examine the difference in source regulation of the two algorithms under the lighter load, the perturbation in source loss and delay between the two configurations are reported.

5.6.4.1. Experimental Method

The network has the configuration shown in Figure 5.15 and the parameters shown in Table 5.27. The configuration above is for the less loaded network to which the earlier Leaky Bucket and DTW simulations are compared below.

5.6.4.2. Results

Table 5.28 summarizes the change in regulation performance.

The columns are the percent change between the fully loaded system and the less loaded system, expressed as a percentage decrease. A positive percentage indicates that the metric improved by that percentage of the worse measurement. DTW reduces delay and losses at the regulator, while the allocation system doesn't change its performance

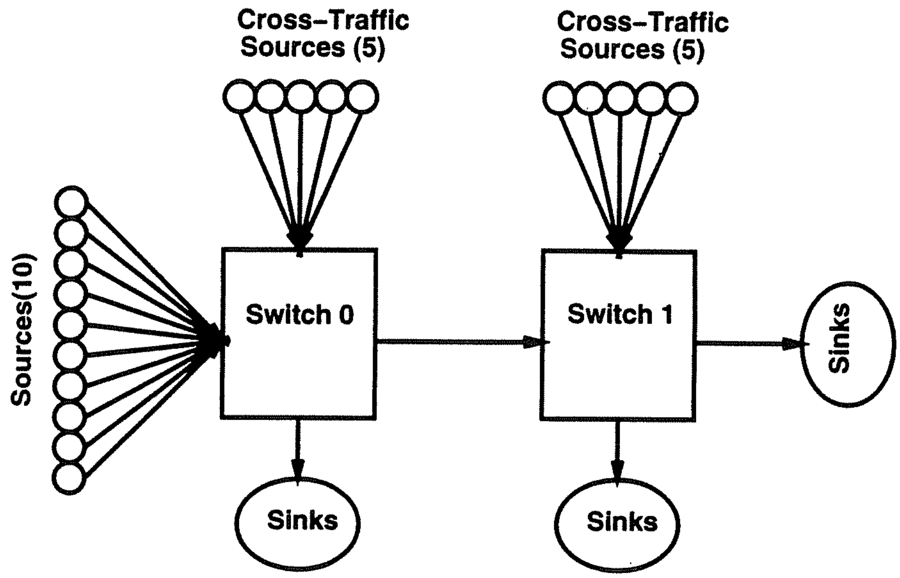

Figure 5.15 : Simulation Configuration for DTW vs Allocation Experiments

Parameter	Value
Switch Buffering	10 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	1.6 Mb
Congestion Threshold	90%
Mean Burst Size	0.8 Mb
Mean Intertrain Time	8 ms
Duration	500 seconds

Table 5.27 : Simulation Parameters for DTW vs Allocation Light Load Experiments

Control System	Improvement in Source Delay (pct)	Improvement in Source Losses (pct.)
Allocation	-2.64%	-3.88%
DTW	19.6%	18.7%

Table 5.28 : Comparison under light load (Allocation vs DTW)

appreciably. (Some of the change in performance of the allocation system is due to the fact that fewer packets are sent in the less loaded simulation, because there are fewer sources, which leads to a difference in averaging.)

The difference in performance is because DTW's probing of the network allows it to adjust its regulation of source burstiness. Leaky Bucket cannot, and must smooth traffic for the worst case. DTW sources are less restricted in the less congested network. DTW only constrains their behavior enough to avoid and control congestion in the current condition, while Leaky Bucket always constrains sources to the worst case.

If other sources begin sending traffic, DTW will adapt to the new state by the process we have seen for adding sources, but Leaky Bucket will remain unchanged, eternally prepared for a worst case.

5.7. Simulation Conclusions

First, we have confirmed the stability results from Chapter 4. The simulations confirmed the analysis, and have also shown that in reality the bounds are somewhat loose. This shows that the analysis reflects a worst case, while showing that worst case is uncommon.

The simulations have shed quantitative light on the problem of forcing sources to conform to the time window criterion. We have explored the problem of using a buffered regulator to control traffic, and seen that this problem is a general one. Although DTW is more strict than many disciplines, as we saw in Chapter 4, the corresponding increase in delay and loss at the regulators is not excessive. We see that the distortions of traffic induced by DTW are comparable to those of other disciplines, and not only pay for the additional stability of DTW, but also for the extensions discussed in Chapters 7 and 8.

DTW's performance was qualitatively and quantitatively analyzed. We showed it to be stable in a static network, and to take advantage of stochastic multiplexing. It reacts well to changes in network state, like sources entering the network. Such entries cause only small perturbations in DTW's performance.

DTW was compared to a feedback congestion control and avoidance system. They provide comparable performance in static networks with enough buffering to allow the feedback system time to react to congestion. DTW sources have an order of magnitude less packet losses in such a network, although the feedback system has a somewhat higher throughput. The higher throughput is offset by the higher network losses for sources that must retransmit lost packets. In networks with less source buffering, DTW performs better, especially in regard to sources that resend lost packets. The throughputs of the systems are very close in these simulations, while DTW maintains its factor of ten edge in avoiding packet loss.

In dynamic networks, the feedback system experiences a larger degradation in all performance metrics. Since it only relies on feedback to control congestion, it is swamped by the rapid changes in network state. DTW experiences performance degradation as well, but significantly less. DTW's superior performance is due to its non–reactive congestion control system.

DTW was also compared with an allocation system. Although for a given network configuration, it is possible to configure an allocation system to deliver comparable performance to DTW, such a configuration must be reconfigured for each traffic pattern and set of circuits. DTW seeks the optimal parameters to its source control by design. DTW allows sources to take advantage of a lightly loaded network and avoid congestion in a heavily loaded one, without any change of parameters.

These simulations have demonstrated DTW's performance quantitatively and shown many of its useful features. It avoids and controls congestion well in a high bandwidth–delay product network, even one that is changing rapidly. In less heavily loaded networks, sources are allowed to use the burstiness capacity of the network.

Chapter 6

Implementation

''In the discovery of secret things and in the investigation of hidden causes, stronger reasons are obtained from experiments and demonstrated arguments than from probable conjectures and the opinions of philosophical speculators of the common sort.''

— *William Gilbert,* **De Magnete (On the Magnet)**

This chapter discusses the prototype implementation of DTW on the XUNET experimental network[6]. We begin with a brief overview of the XUNET, and then describe the implementation of DTW on the XUNET switch and XUNET router. The remainder of this chapter describes several experiments that show that the implementation behaves as predicted by simulation.

6.1. The XUNET Network

The XUNET, a nationwide, high speed ATM network created by AT&T, is part of the BLANCA gigabit testbed sponsored by the National Science Foundation (NSF) and the Advanced Research Projects Agency (ARPA). The testbed is administered by the Corporation for National Research Initiatives (CNRI). XUNET consists of two network elements, XUNET switches and XUNET routers. For brevity we will occasionally refer to these as switches and routers throughout this section. The switches are fast, programmable ATM switches. The routers act as gateways from local area networks to the XUNET. The network consists of ten XUNET switches connecting seven endpoints

Endpoints are sites with routers. Figure 6.1 shows the locations of the switches and routers. The sites are connected by 45 Mb/sec lines. A 622 Mb/sec line also exists between Wisconsin and Illinois, but was unavailable for these experiments. Although XUNET is an ATM network in the sense that it sends 53 byte cells across virtual circuits, it does not conform to the ATM Forum standards nor to the CCITT standards. This has little bearing on our experiments.

The XUNET switch is a descendent of the Datakit switch[87]. The switch architecture is shown in Figure 6.2. It consists of a fast backplane with several cards attached to it. The cards are used to transmit and receive cells, queue cells, and switch cells. The cards used for queueing cells are called queue cards, and the single card used to switch them is called the translation module. Cards used to transmit and receive cells are called

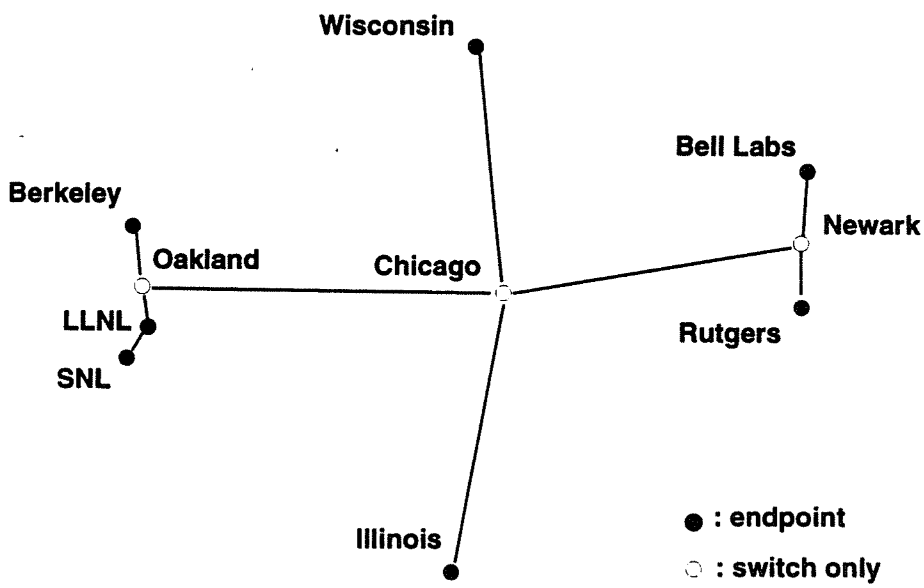

Figure 6.1 : XUNET Topology

line cards, and are always associated with a queue card. The backplane is divided into four busses, the arbitration bus, the contention bus, the broadcast bus, and the maintenance bus. Only two of these appear in Figure 6.2, namely the contention and broadcast busses, which are used to transfer cells between cards. The arbitration bus is used by the queue cards to secure access to the contention bus. The maintenance bus is used internally by the switch and does not impact our experiments.

Attached to these busses are up to eight queue card/line card pairs and one ATM translation module. The queue card/line card pairs transmit and receive traffic on the associated line, as well as queueing cells until they can be switched or transmitted. The queue cards contain a flexible queueing engine, capable of implementing several types of prioritized cell scheduling. The ATM translation module translates cells' source VCIs (virtual circuit identifiers) to their destination VCIs.

The operation of the switch can be best understood by tracing a cell through it. A cell enters the switch when a line card receives it from the attached line. The cell is passed to the attached queue card, which queues the cell until it can be switched. Once a queue card has a cell that needs to be switched, it begins participating in a distributed algorithm [88] on the arbitration bus with all other queue cards that have cells to be switched. This algorithm determines fairly which queue gets access to the contention bus. The queue card that was granted use of the contention bus puts one cell on it, bound for the translation module. The translation module changes the cell's current VCI, which is the virtual circuit it arrived on, to the VCI of the outgoing virtual circuit, and places the cell on the broadcast bus. The cell is removed from the broadcast bus by the appropriate queue card. The cell is queued until the line card is able to send it.

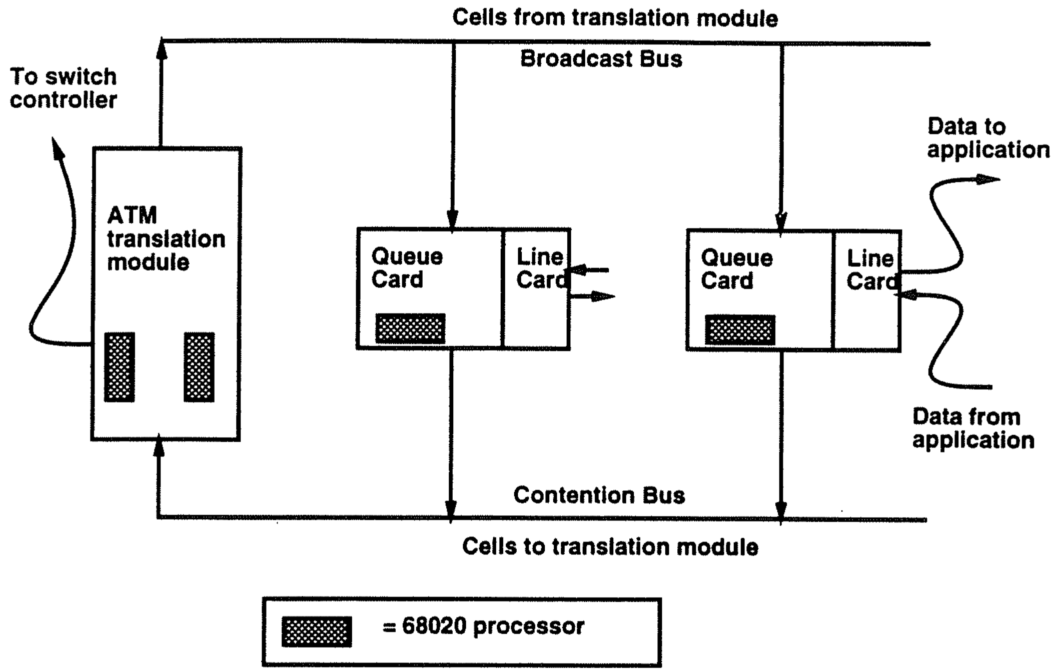

Figure 6.2 : XUNET Switch Architecture

There are several Motorola 68020 processors in the switch. The cell handling is done by custom hardware in the translation module and in the queue cards. The parameters of the algorithms used by that hardware can be adjusted by programs running in the processors. During normal operation, the processors have few tasks. The processors on the queue card are essentially inactive, except for exchanging occasional cells with the switch controller (described below) to assure the controller that each card is functioning normally. One processor in the translation module runs an Ethernet interface, which allows the translation module to communicate with the controller and other switch elements. The other processor scans the address translation table, which controls the mapping of source VCI to destination VCI, for errors. We use these processors to implement DTW feedback algorithms.

Attached to each switch is a general purpose computer, used as a switch controller. The switch controller is responsible for making certain that all elements of the switch are operational. It also runs a protocol to establish and remove virtual circuits. This protocol is currently unimplemented. The controller communicates with the switch across a dedicated Ethernet. Either the switch or the controller can send information on the Ethernet.

The other element of the XUNET network is the router. The router is a general purpose computer used as a gateway between a Fiber Distributed Data Interface (FDDI) local area network and the XUNET. It has a custom kernel, which includes modifications to allow processes to send ATM cells directly. The modified kernel also includes a device driver to communicate with a high speed interface that sends ATM cells suitable for XUNET. This device is known as the Hobbit board[89], named for the AT&T processor used as a DMA engine on it. The Hobbit board and device driver take a buffer of data from the kernel and divide it into ATM cells which are then sent out into the XUNET. They can also reverse the process to receive data and forward it to the kernel of the router. In general, this data is an IP packet, although it may also be raw data to be sent directly in ATM cells using a variant of AAL5 packet framing. The device driver also contains the appropriate knowledge to map an IP address to a VCI and back.

Once a virtual circuit exists, one computer sending an IP packet to another on the XUNET consists of the following steps. A packet is routed from the sending computer to the nearest router across the LAN connecting the two. The router decides, based on the address of the packet, which virtual circuit to forward the packet on, and sends it via the Hobbit board. The Hobbit board divides the packet into ATM cells, which are passed from switch to switch based on VCI, until they reach the destination router. This router's Hobbit board reconstructs the packet, and delivers it to the router's kernel. The kernel then forwards the packet to the destination computer via the LAN connecting them.

6.2. The DTW Implementation

Implementing the DTW prototype involved implementing the monitoring and feedback algorithms in the switch, and the source control algorithms in the sending host. We implemented the feedback system on the processors in the switch, and the source control algorithms in the router. The decision to implement the source control at the router was made to avoid having to send data across the local area net connecting the router to another host. It was also useful that the router can send and receive ATM cells directly, which simplified the design of the feedback system. We gratefully acknowledge Lui Chan's help in implementing initial designs of the prototype.

6.2.1. The Switch Feedback System

The switch has two responsibilities under DTW. It must implement Weighted Fair Queueing, and run the MTW adjustment algorithms. This section describes how each of these is implemented.

The XUNET switch cannot implement WFQ, but the programmable queueing hardware can trivially implement weighted round robin. Under weighted round robin, a weight is associated with each queue. All the queues with cells on them are served in order, and each time a queue is served, a number of cells equal to its weight are removed. If a queue has fewer cells than its weight queued, those cells present when it is served are sent. Over the long run, the weights in weighted round robin approximate the service shares in WFQ.

To be certain that the XUNET switch can approximate WFQ, we did the following experiment. Using code written by Robert Olsen, we established 10 virtual circuits through one queue module, to the backplane and back to the queue module. Cells on these circuits left the queue module, passed through the translation module, where they received the same VCI, and were passed back to the originating queue module in an endless loop. These ten circuits each were assigned the weights from 1 to 10. An eleventh circuit was started, and its throughput measured at the translation module by one of the processors there. Table 6.1 summarizes the results.

The ''predicted throughput'' column in Table 6.1 is the throughput predicted by equation (3.2), given an experimentally observed backplane speed of 495.8 Mb/sec. Measurements were taken for several seconds with the switch fully loaded. Approximating WFQ by weighted round robin over a period of seconds is justified by Table 6.1.

Implementing the feedback algorithm was more complicated. The majority of the feedback algorithm, which is described in Chapter 3, is implemented on the 68020 which is installed on the queue card. This processor is referred to as the queue processor. The queue processor has access to the per–virtual circuit queue lengths maintained by the queueing hardware, as well as the ability to inject cells into the network as feedback. Unfortunately, it does not have access to an accurate timer. Fortunately, there is a timer

Weight of Test Circuit	Observed Throughput (Mb/sec)	Predicted Throughput (Mb/sec)
1	8.81	8.85
2	17.3	17.4
3	25.5	25.6
4	33.4	33.6
5	41.1	41.3
6	48.5	48.8
7	55.7	56.0
8	62.6	63.0
9	69.4	69.7
10	75.9	76.3

Table 6.1 : Weighted round robin vs. WFQ

on the translation module with a 125 ns resolution. The implementation uses the timer on the translation module to keep time, and a processor on the translation module sends cells to the queue processor periodically to let it know time is passing.

This method is practical since whenever the processor on the translation module that is responsible for communication between switch elements sends a cell, that cell is sent immediately. We use a timer resolution of one cell per millisecond. Considering that the timer has a 125 ns resolution, a cell will cross the backplane in 770 ns, and sending a cell requires only a few 68020 machine instructions, the overhead of getting these cells to the queue processor is negligible.

The queueing engine does not notify the queue processor when cells are lost, nor does it record such losses. To monitor virtual circuit queue lengths, the queue processor must poll them. In an ideal system, the queueing engine would interrupt the queue processor when a cell was lost. Minimally, the queueing hardware should record losses. However, the prototype used in these experiments serves few enough sources that polling is adequate.

The partition of functionality in the feedback software is shown in Figure 6.3. The program running in the queue processor knows the VCIs and congestion thresholds of the queues to monitor. It monitors the length of these queues and follows the algorithm in Chapter 3 to adjust its MTW. When the queue processor sends the new MTW as feedback, it also sends the maximum queue length recorded over the current monitoring interval.

Two programs run in the translation module, one to construct a cell to be sent to the queue processor every millisecond, and one to forward the cell to the queue processor. This unusual partition of functionality is driven by the hardware. One processor can

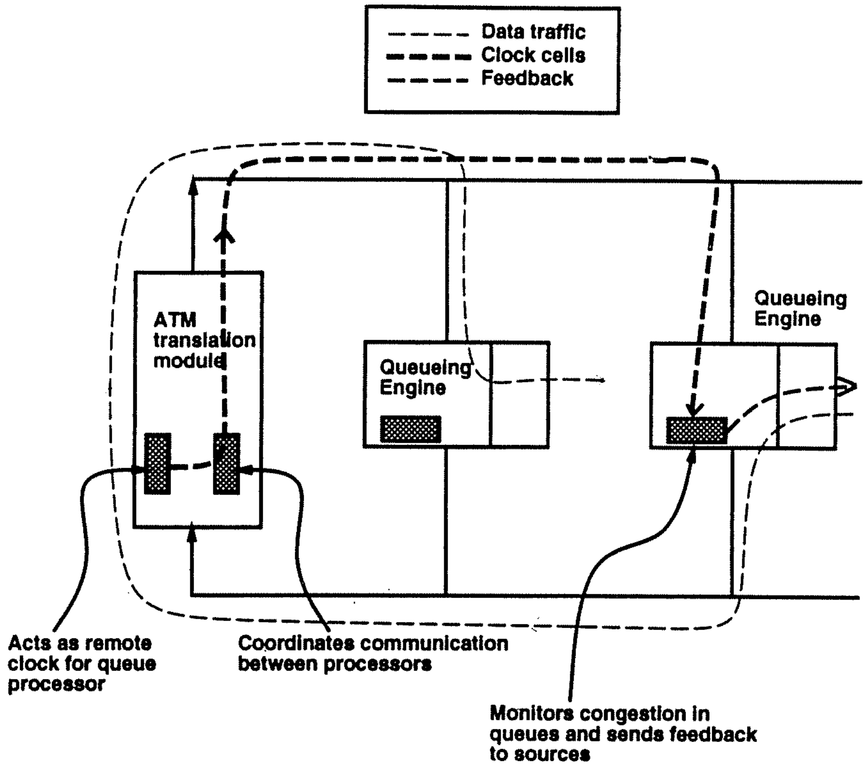

Figure 6.3 : XUNET Switch Architecture

access the timer, and one can access the backplane.

6.2.2. The Source Control Algorithms

Implementing the source control algorithms was straightforward. We implemented the source control on the XUNET router, so that we could take advantage of the ATM transmission capabilities, and to avoid the address translations and traffic distortions of crossing the attached FDDI network. The time window enforcement algorithms are implemented in user space, as a library.

The decision to implement the source control algorithms in user space was to avoid having to work in the system kernel. Implementing in user space made development easier, since rewriting networking code in an operating system is notoriously difficult.

The second motivation was that user space code is more portable. A user library similar to the one described here has been ported to several different architectures by Lin Cui.

Once the library was written, we tested the outgoing traffic stream, and found that it less than 5% of all packets sent violated the time window criterion. The packet streams in question crossed a local area network (FDDI), and may have been somewhat distorted. It should be stressed that 5% is the highest deviation we saw even after crossing the LAN, and most cases had no violations.

The library does suffer from the problem that for small time windows, the resolution of the clocks available to the user level process is not fine enough to allow a smooth enforcement of the time window criterion. The clock resolution is about 10 ms, and for time windows smaller than that value, the system loses throughput. The magnitude of this loss will be described later in this chapter when we describe the experiments performed with the prototype.

6.3. Experiments

This section describes the experiments done using the prototype. First, we describe the configuration of the network used for the experiments. We show evidence that the prototype operates as predicted by the earlier simulation studies. We then show that the system adjusts to various values of congestion threshold, system load, and negotiated source throughput. Finally, we show that DTW responds to changing network state.

6.3.1. The Experimental Configuration

We chose to test the implementation on the link between the University of Wisconsin at Madison and the University of Illinois at Champaign–Urbana. See Figure 6.4 There are three switches along the path, but for simplicity of gathering experimental results, we ran the feedback algorithms only on the bottleneck queue module. For traffic travelling from Wisconsin to Illinois, the bottleneck queue module is the queue module connected to the 45 Mb/sec line at Wisconsin. We have observed the Hobbit board sending bursts of data at more than 60 Mb/sec, although it can sustain only about 30 Mb/sec. Bursts of data sent at 60 Mb/sec will meet their first bottleneck at the 45 Mb/sec line mentioned above.

The XUNET router in Wisconsin is the source of traffic. We limit ourselves to one source on one router for several reasons. Having one source allows us to measure and record its responses to experiments easily. Sources at other routers are likely to have other bottlenecks, and synchronizing them to produce interesting traffic patterns is difficult.

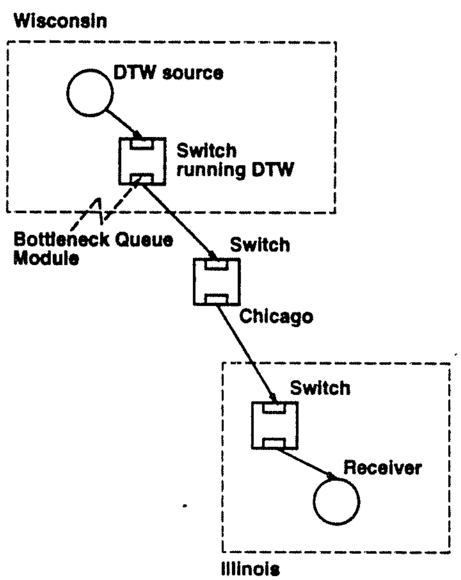

Figure 6.4 : The Experimental Environment

We vary several network parameters in the experiments. It is possible to apply an artificial load to the switch, which can be better controlled than real cross traffic. The amount of artificial load in the network is varied. We vary the queue length that the switch must observe to reduce its MTW, and therefore the source's time window. This value is called the congestion threshold. The source's negotiated throughput is also varied. We show that DTW responds to these different network configurations.

The test application is a process sending 5000 byte packets as fast as it can and still meet the time window criterion. It always uses a time window ratio (r) of 1, so its time window is always the MTW of the switch.

Certain parameters remain the same across all experiments. The AMTW is always 50 seconds, the additive increase unit of the time window is 5 ms, and the multiplicative decrease factor is 2. The minimum MTW the switch can have is 2 ms, and the maximum is 10 seconds.

The switch MTW is always initially 1.0 sec. This is larger than the stable time window value for most of the configurations tested, and the MTW is quickly closed to a reasonable value. The large initial value of the MTW causes some edge effects in the results, mostly visible as a burst of losses at the beginning of each experiment. In a production system, each new virtual circuit would receive the switch's current MTW, rather than an arbitrary value.

To load the network, we established another circuit looping back on itself through the bottleneck queue card. This virtual circuit has no endpoint. It is a path that forms a closed loop between the switch in Wisconsin and the switch in Chicago. The source's virtual circuit and the load circuit have the same weight in the weighted round robin queueing at all queue modules. The load was varied on the network by varying the number of cells in the looped circuit.

Other experimental parameters that are varied are the negotiated average rate of the source and the congestion threshold of the source's queue at the switch. The values of these parameters will be explicitly stated in the descriptions of the experiments below.

6.3.2. Basic DTW Behavior

Our first concern was that the implementation should behave as DTW did under simulation. In a stable system, the time window should oscillate about a fixed value. The queue length at the switch should respond to changes in the time window, and it should rarely exceed the congestion threshold.

The experiment that follows is a single source sending through an unloaded switch. The congestion threshold is 500 cells. The source negotiated an average rate of 15 Mb/sec. We report a typical experiment.

Figure 6.5 plots the source's time window for the duration of a two minute run. The height of the curve in Figure 6.5 represents the size of the time window, and the length of the line segments represent the time that the source used that time window. The horizontal line is the average of the time window over the length of the experiment. This value is the sum of the product of the time window values and their duration of use divided by the total time of the experiment. For a stable network, the average time window is a reasonable approximation of the point about which the time windows are oscillating.

Figure 6.5 shows that the implementation displays the behavior predicted by simulation for DTW in a stable network. The time window oscillates about a fixed point. The source's throughput, recorded at the source, is 13.9 Mb/sec. The loss in throughput is primarily due to the inaccurate clock on the source.

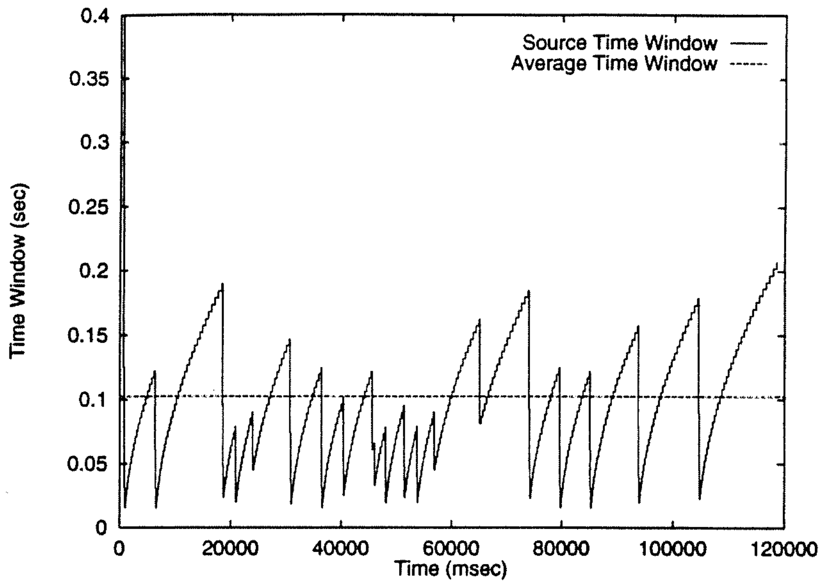

Figure 6.5 : Source Time Window vs Time (Unloaded Switch)

Figure 6.6 is a plot of the source's maximum queue length for the same two minute run. Since the traffic is obeying the time window criterion, the queue length is returning to zero many times throughout the experiment. The lower horizontal line is the congestion threshold, and the upper line is the maximum possible queue length.

From Figure 6.6, we see that the queue length closely follows the time window plotted in Figure 6.5. The time axes are the same, and it is easy to determine that the queue length crossing the congestion threshold results in a reduction in the time window. The reduction in the time window reduces the maximum queue length over the following intervals. Notice that other than the beginning of the experiment, the queue length rarely passes the congestion threshold. The feedback is causing the system to avoid congestion. Observing the cell counts at the last switch in the network show that losses are negligible in this configuration, and generally occur only during the initial search for an appropriate time window.

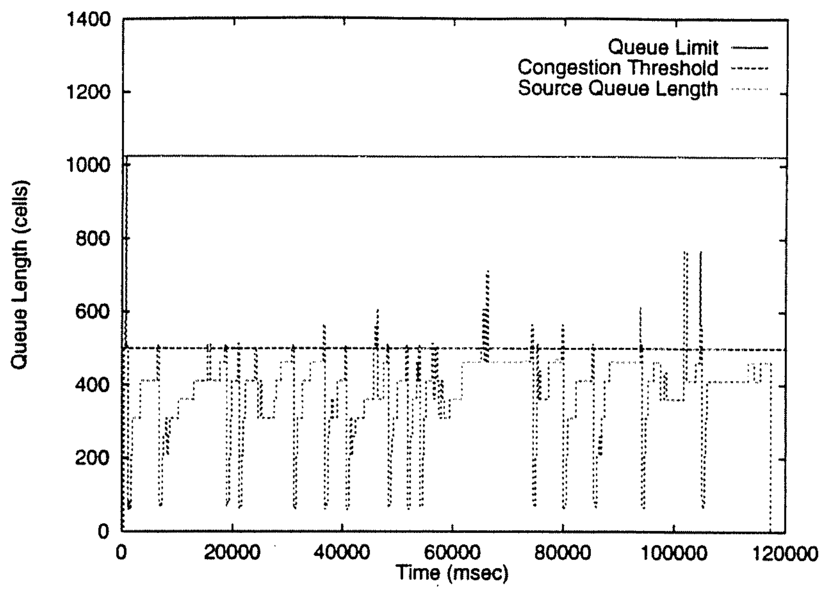

Figure 6.6 : Source Queue Length vs. Time (Unloaded Switch)

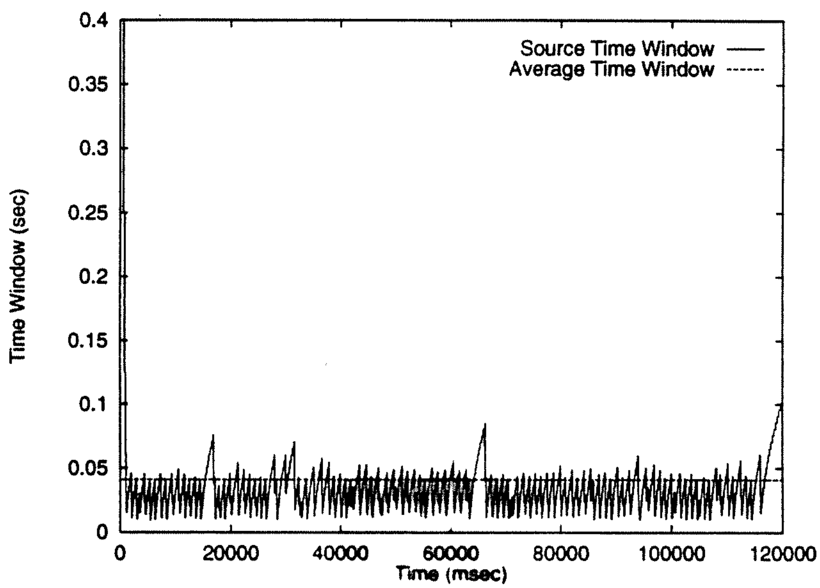

Figure 6.7 : Source Time Window vs Time (Loaded Switch)

To be certain that the system was responding to the state of the network, we loaded the switch, and repeated the procedure above. All parameters are the same except the network load, which is 250 cells. We plot data for a typical experiment. Figure 6.7 and Figure 6.8 plot the same values as Figure 6.5 and Figure 6.6 respectively, on the same axes. The throughput in this experiment was 12.7 Mb/sec. We notice that both the period and the amplitude of the time window oscillations are smaller. The test source must be less bursty in this network to avoid loss. The queue length tracks the time window behavior. Queue lengths exceed the congestion threshold more often, but the system continues to avoid losses. DTW adapts to the more loaded network by choosing smaller time window values. This limits the source burstiness and avoids congestion.

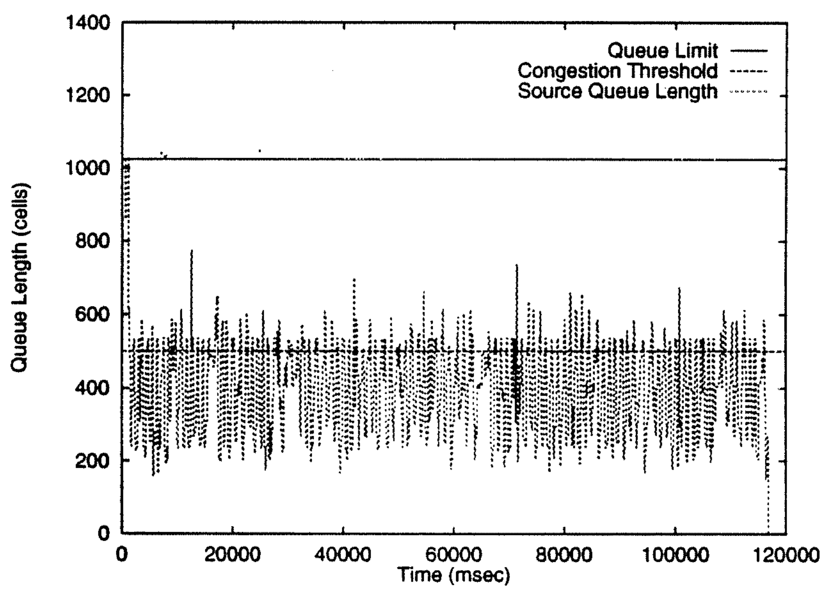

Figure 6.8 : Source Queue Length vs. Time (Loaded Switch)

6.3.3. Responding to Network Load

This section describes experiments that show more quantitatively how DTW responds to network load. Each experiment is a two minute run. The source negotiates an average rate of 15 Mb/sec, and the first set of experiments uses a congestion threshold of 500 cells. The network load is varied from 0 to 450 cells.

Table 6.2 summarizes the results of these experiments. The ''load'' column is the number of cells looping in the load circuit. Throughput is measured at the source, and the average time window is calculated as defined above. Losses are reported in absolute terms and as a percentage of total transmitted cells.

The increasing load on the network causes the source's queue to be more full, resulting in the MTW being decreased more often. The adjustment algorithms cause the time window to oscillate about a lower average. DTW effectively senses the load, and adjusts source burstiness to meet it.

The losses are low. The majority of the losses for runs at low load occur during the initial adjustment period of the time window. Roughly a thousand cells are lost in the

Load (cells)	Throughput (Mb/sec)	Average Time Window (sec)	Losses (cells)	Losses (pct)
0	13.9	0.103	1048	0.0235
50	13.7	0.0795	1297	0.0295
150	12.9	0.0442	1522	0.0314
250	12.7	0.0409	6278	0.15
350	8.33	0.0247	28455	1.24
450	3.41	0.0172	623181	55.5

Table 6.2 : Summary of Network Load Experiments (Threshold = 500 cells)

initial time window adjustment. Beyond that the losses represent occasional congestion. Until the network load climbs to the point of congestion collapse, DTW performs well in avoiding loss.

When the network is loaded by a 450 cell burst, it has been pushed beyond DTW's ability to control it to congestion collapse. The time windows of the sources in those experiments are at their lowest value and only oscillating slightly. The network is overbooked at this point. The load circuit alone is exhibiting enough burstiness to congest the network, and DTW cannot adjust its burstiness.

At a load of 450 cells, the average rate of the load circuit alone exceeds 45 Mb/sec. The line card operating at 45 Mb/sec can drain a queue of 425 cells in the 3.81 ms that it takes for a cell travel round trip to Chicago and back from Wisconsin (the round trip time was observed by Robert Olsen). Since there are more than 425 cells in the queue, it will never empty, so the load circuit presents an effective rate of 45 Mb/sec to the queue module. This violates Theorem 3, given in Chapter 4, so the system is not guaranteed to be stable.

Table 6.3 summarizes the same experiment for a higher congestion threshold, 800 cells. The network is equivalently loaded, but since the source is willing to let the switch get more congested, the average time window is larger. Again the network experiences congestion collapse at a load of 450 cells. We have shown that the network is uncontrollably congested at this point.

From Figure 6.3, it is apparent that raising the congestion threshold allows a source to maintain a larger time window. In this implementation, this raises the throughput that the source can achieve as well. However the price is that the source's queue length at the

Load (cells)	Throughput (Mb/sec)	Average Time Window (sec)	Losses (cells)	Losses (pct)
0	14.7	0.326	2215	0.0457
50	14.4	0.169	901	0.0191
150	13.9	0.0886	4586	0.0998
250	13.6	0.0697	46325	1.05
350	10.23	0.0251	53100	1.96
450	3.73	0.0178	918360	64.8

Table 6.3 : Summary of Network Load Experiments (Threshold = 800 cells)

switches is higher on the average, and its losses are higher.

6.3.4. Enforcing Negotiated Throughput

These experiments reflect DTW's ability to detect and control burstiness from one source. We induce additional burstiness into the network by increasing the negotiated average rate of the sources. This results in a larger burst size for a given time window.

We expect DTW to handle this traffic with less losses than the artificially loaded network, since the switch feedback will be adjusting the burstiness of all traffic using the switch. In the previous experiments, DTW could not adjust the burstiness of the load circuit.

The experiments consist of varying the source's negotiated throughput in an unloaded network. We run several two minute experiments using source average rates between 5 and 30 Mb/sec. We have observed that sending at a sustained rate much above 30 Mb/sec results in losses at the Hobbit board. The experiments are carried out for congestion thresholds of 500 and 800 cells.

Table 6.4 summarizes the experiments for a congestion threshold of 500 cells, and Table 6.5 summarizes them for the 800 cell congestion threshold. Actual throughput suffers due to the limitation on the source's clock. This affects the sources with a lower congestion threshold more, because their average time window is lower. Since those sources want to keep a lower queue length at the switch, they have send smoother traffic, which appears as a smaller time window.

The losses for sources negotiating a small average rate in the experiments with the higher congestion threshold are somewhat artificial. Most of those losses occur at the start of the experiment when sources are finding the appropriate time window. Roughly the first thousand cells or so are lost during that initial burst. Since this is more or less independent of the negotiated throughput, this explains why the percentage losses are reduced by increasing the negotiated rate. The total number of packets forwarded rises,

Negotiated Throughput (Mb/sec)	Actual Throughput (Mb/sec)	Average Time Window (sec)	Losses (cells)	Losses (pct)
5	4.91	0.801	0	0
10	9.83	0.438	0	0
15	13.86	0.102	1048	0.0235
20	16.86	0.0472	1818	0.0428
25	19.19	0.0356	8812	0.140
30	19.45	0.0281	16074	0.246

Table 6.4 : Summary of Negotiated Throughput Experiments (Threshold = 500 cells)

Negotiated Throughput (Mb/sec)	Actual Throughput (Mb/sec)	Average Time Window (sec)	Losses (cells)	Losses (pct)
5	4.95	1.48	1207	0.0739
10	9.81	0.367	1825	0.0342
15	14.65	0.326	2215	0.0457
20	18.55	0.104	2825	0.0453
25	22.20	0.0632	8817	0.121
30	24.49	0.0418	16287	0.203

Table 6.5 : Summary of Negotiated Throughput Experiments (Threshold = 800 cells)

and this fixed burst of losses at startup remains constant.

In this case, the higher congestion threshold has significantly less effect on losses. The losses remain somewhat higher for sources with the higher threshold, but not as markedly as in the previous section. This is because DTW is able to control the burstiness of all sources. The previous experiments featured a source sending at a fixed burstiness that DTW was unable to smooth as necessary. In this case DTW is capable of controlling all the contributions to network congestion, and therefore the network is more stable.

6.3.5. Dynamic Adaptation to Changing Network State

The experiments we have reported so far have all involved a static network. Various parameters were changed, but the network maintained the state defined by those parameters throughout the experiment. This experiment shows DTW's ability to detect and respond to a change in network state.

The experiment consisted of a source sending at a negotiated rate of 15 Mb/sec with a congestion threshold of 800 cells for ten minutes. The network was initially unloaded. After the source had been sending for approximately five minutes, 300 cells were injected into the load circuit. Recall that the load circuit is a loop, so the network remained loaded for the remainder of the experiment. We observed the source time window and switch queue length.

The source maintained a throughput of 14.0 Mb/sec. We will see that the average time window is of little interest in this case, as the source's time window is not oscillating about the same value in the first five minutes as it is in the last five minutes.

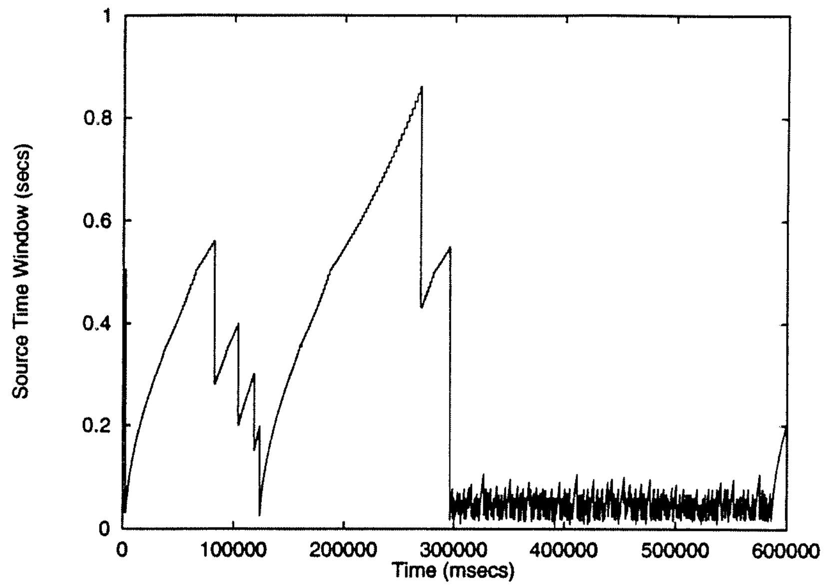

Figure 6.9 : Time Window of a Source in a Changing Network

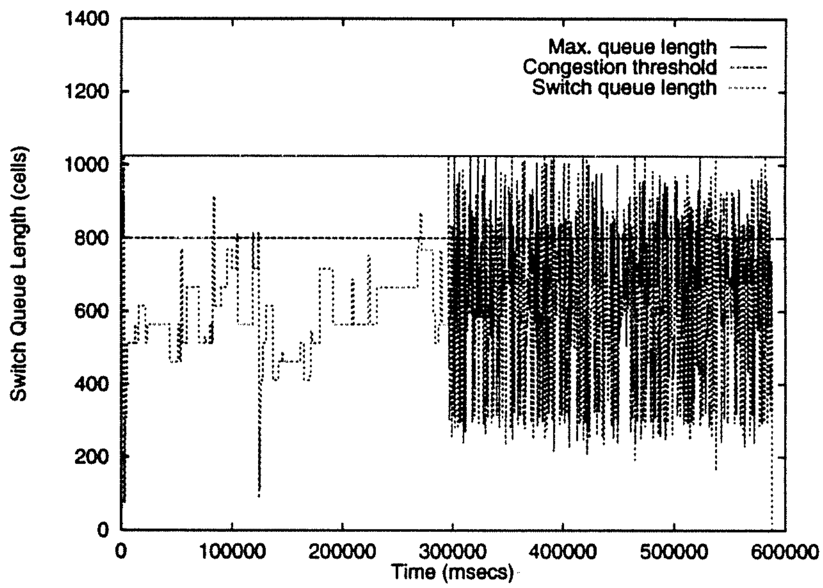

Figure 6.10 : Queue Length of a Switch in a Changing Network

Figure 6.9 and Figure 6.10 show the time window and queue length of the source during a typical run of this experiment. The switch detects the change in network load, and the source's time window is adjusted. As we have seen in earlier experiments, a load of 300 cells on the network is a heavy one, and the source has to constrain its behavior significantly. This makes the change in source behavior evident in Figure 6.9.

As the load increases, it is also reflected in the switch queue length. The increase in network load increases the maximum queue lengths reported by the switch. Losses occur in the second five minutes, and are negligible in the first five minutes. Under added load the switch queue lengths are more bursty, but become centered about the congestion threshold. Although losses occur, they are rare.

This experiment shows that DTW quickly senses a change in network congestion and takes effective measures to combat the detected congestion. This agrees with the behavior observed in simulation (see Figure 5.10), and is an excellent property in a congestion control and avoidance system.

6.3.6. Summary of Experiments

The experiments described in this chapter are useful because they show that DTW is implementable on existing hardware, and show how DTW functions on real traffic. DTW places some demanding requirements on the hardware necessary to implement it, and we have shown that such hardware exists today. Furthermore, the system functions well, even though the hardware does not meet the exact needs of DTW. An approximation is made to WFQ, and the switch algorithms must poll the hardware to monitor switch state, yet the prototype avoids and controls congestion.

The prototype provides evidence that the simulations and analytical work rest on reasonable assumptions. The fact that the system is not forced to congestion collapse until Theorem 3 is violated provides evidence that despite the assumptions made in that theorem, it is applicable to a real system. The similarity of Figure 6.9 and Figure 5.10

show that the simulations predict the behavior of the real system well.

Much needs to be done before the prototype described in this chapter represents a real system. However, the results here indicate that DTW is effective at controlling congestion in a real network.

Chapter 7

Service Tailoring in DTW

"He picks up scraps of information –

He's adept at adaptation''

— Neil Peart, ''Digital Man''

This chapter describes features of DTW that allow sources to request different types of service from the network. The process of serving traffic in a way designed to meet those requests is called *service tailoring*. We begin by explaining why there is a need for these features in DTW and then describe the algorithms used to implement them. In closing, we describe how these algorithms can be used to provide best effort delivery.

7.1. Motivation for Service Tailoring

Traditionally network applications fall into two main categories, file transfer and remote login. File transfer applications move files from computer to computer, and require high throughput with no losses. Such applications commonly achieve loss free delivery to the destination computer by resending data lost by the network, so they have a higher throughput of useful information if losses in the network are low. Remote login applications are users accessing computers remotely. They use little bandwidth compared to file transfer sources, but because of the interactive nature of the application, they expect a fairly short delay.

To serve these two types of applications, current congestion control and avoidance strategies keep throughput high, and avoid pushing the network to congestion collapse.

The benefit of maximizing throughput to file transfer sources is obvious. Remote login users are willing to tolerate fluctuations in service, assuming the worst case is not excessively bad. Maintaining a high throughput for all sources is sufficient to keep response time low for the small messages of remote login sources in today's networks.

This simple environment is being disturbed by the arrival of new applications. The most commonly cited, and perhaps most disruptive of these, is real–time traffic. Other new sources include distributed scientific computations, which send infrequent large bursts of data, and remote monitoring and control systems which send low bandwidth continuous streams of data that may have associated timing constraints. Other new applications are appearing regularly. We believe that the combination of increasing bandwidth and growing availability of networks will continue to attract new applications. Many of these will have performance requirements that today's network designers have not predicted.

7.1.1. An Example – A Real–time Source

A real–time source prefers its traffic to experience low delay and jitter. Bounds are often placed on quantities like delay and jitter by the application and enforced by the network. These bounds are called real–time performance bounds. Certain real–time sources require such bounds, and some operate without them. Examples of real–time sources include video teleconferencing and audio conferencing.

The delay constraints exist because the source's traffic contains a record of some event that occurred at the sender that is being recreated at the receiver. The intent is that the receiver be able to react to the remote event as though it were happening locally. If the event is not reproduced quickly, the information is useless. For example, in order for two people to converse in an audio conference, each person's comments need to be delivered to the other promptly.

Jitter is the amount consecutive cells become separated in time or compressed in time while crossing the network. This can affect reproduction of an event, by distorting the temporal component of data. Sounds that form words in one reproduction can form gibberish when they are grouped differently by jitter. Jitter is removed by recording the temporal information in the data, and using buffering to reconstruct the event at the receiver before replaying it. Bounding jitter allows applications to bound their buffering at the receiver.

Jitter and delay are related. If there is a bound on the amount that two cells can be separated in time, there is necessarily a delay on the length of time it will take a cell to traverse the network. A jitter bound implies a delay bound and vice versa.

The jitter and delay of a real–time source determine the playback point of the source. The playback point is the time by which the next event must be reconstructed at the destination to maintain a desired quality of the event reproduction. For example, an application to transmit interactive full–motion video at 60 frames per second has playback points spaced 16 ms apart. In order to keep the quality of the event reproduction, *i.e.,* the video quality, at the desired level, each event, *i.e.,* each video frame, must be recreated at the destination 16 ms after the previous event. As we describe below, the quality of event reproduction can either be determined *a priori* and the real–time performance bounds and playback point calculated from it, or the playback point can be set to a value reasonable for the current observed network state, and the quality of event reproduction determined by the network state.

7.1.2. Serving Real–time Traffic

Several systems have been proposed to provide networks that enforce real–time performance bounds. Such systems constrain a source's traffic to meet certain criteria, and guarantee that the real–time bounds on the source's traffic will be met through the use of specialized queueing systems. Examples are the Tenet System[90, 91], and Stop–and–Go[23]. Such systems are basically tuned allocation congestion control and avoidance systems. The constraints on traffic and associated queueing disciplines in the network combine to ensure that the networks will meet the requested bounds. They suffer from the same shortcoming of other allocation systems, namely underbooking the network.

Clark, Zhang, and Shenker[92] describe an alternative to the guarantees provided by these systems. They describe a type of service called predictive service. Under predictive service, applications that can adjust their playback point do so based on the observed performance of the network. For example, a video application that was observing too much jitter in its traffic to show 60 frames per second may adjust its rate to 45 frames per second to allow more time for smoothing the jitter. In this model, the destination observes the current maximum or average jitter in the network, and sends feedback to the source indicating what frame rate it should use. Clark *et. al.* observe that in many cases the actual value of the performance metric is so much better than the worst case bound, that predictive service is the best way to serve sources with real–time constraints.

7.2. Service Tailoring Under DTW

Although bounds on DTW's delay and jitter can be directly calculated (see Appendix A), we share the view that predictive service is an effective way to serve sources with special needs. The calculations of worst case delay are very similar to those performed in Chapter 4 to calculate a source's effective average rate. In the same way that these calculations consistently overestimate the change in that source's average rate, calculations of the worst case delay under DTW consistently overestimate the usual delay experienced by traffic.

Instead of calculating and enforcing such bounds, DTW allows sources describe how the network is to queue the traffic at switches, and what conditions are to be considered in the generation of feedback by the switches. Specifically, the source can request resources in the network, and choose from several sets of feedback algorithm parameters. Each of these methods of tailoring DTW's performance is provided by a separate subsystem. There is a resource allocation subsystem to allocate service and buffer shares at switches, and a selective feedback subsystem to allow a source to choose the parameters of a feedback algorithm that suits the source's needs. These subsystems combine to provide the source with performance tailored to its needs. Such a system is ideal for predictive sources, since it allows them to communicate the service they require to the network, and the base their operation on the resulting service.

Resource allocation is performed at switches, and consists of changing the amount of buffering or service rate that a source's traffic gets at the switch. These allocations are given in terms of the source shares and buffer shares defined in Chapter 3. Allocating resources unequally to sources directly affects the performance of their traffic. All other parameters being equal, sources with a larger service share at a switch will be likely to have a lower per–cell average delay than sources with smaller service shares. A larger buffer share may protect a source against loss, or increase its throughput, depending on how its feedback is structured.

Selective feedback parameterizes the feedback algorithms described in Chapter 3 to take into account the differing requirements of sources. Instead of all sources receiving the same MTW from switches, sets of sources that have requested similar service are controlled as a group. Each group has a separate MTW, controlled independently of the other groups' MTWs. All MTWs are controlled by switches. Each switch runs an MTW adjustment algorithm for each group, tailored to the group's needs.

These two mechanisms are used together to tailor network performance. For example, delay reducing sources increase their service share at switches, and request feedback tailored to keep their queue occupancy low. Loss avoiders request additional buffering and use feedback to keep some of their buffers in reserve.

DTW is not the only system that could provide resource allocation as a service tailoring method. Any system using WFQ, or a similar queueing discipline, to mete out switch service rate could use the techniques described here. Allocating buffer space has similar requirements. The selective feedback techniques are less general. As we shall show the use of selective feedback enhances resource allocation based strategies.

DTW was designed as a general system to allow sources to describe the service they want the network to provide, rather than providing bounds on a few performance metrics. We believe that allowing applications to adjust their behavior in response to network performance is reasonable, since they have more knowledge of how to adapt themselves to changing network state than the network provider. Furthermore, we feel that whatever metrics one chooses to bound, other metrics may be important to some future application. Providing a system through which sources can describe how the network should handle their traffic allows DTW to effectively serve the future generations of applications.

7.2.1. Resource Allocation

Given the algorithms in Chapter 3, the implementation of resource allocation is straightforward. The switches can allocate two quantities, service rate and buffer space. A source requests its allocations by specifying a service share (s_i is the service share for source i) and a buffer share (b_i) in its connection establishment message. If granted by the switch, the allocations are enforced by the WFQ algorithm and the custom buffering algorithm described in Chapter 3.

Deciding what values are appropriate for a given source may not be straightforward. A source needs to be aware of the nature of its traffic and the amount of resources available at switches. Not all network users are sophisticated enough to acquire and interpret this knowledge, so we envision a system, perhaps as simple as a run–time library, that will pick appropriate parameters for common cases. For example, all video traffic sending a particular frame size will share the same parameters.

In addition to knowing the traffic type, DTW will have to interact with the virtual circuit routing system. Source requirements may determine the route of the virtual circuit, causing the routing system to route the circuit through switches that have enough resources to meet the source's demands. In other cases, the routing algorithm may adjust connection establishment parameters to effectively use switches with less capacity. Although finding the correct parameters for classes of traffic is an interesting problem, we content ourselves with examining how well DTW tailors service given those parameters. To that end, we do not consider how source resource requests are computed.

7.2.2. Selective Feedback

The other component of service tailoring is selective feedback. Selective feedback is the gathering of sources with similar requirements into groups that receive similar feedback. All members of a feedback group receive feedback from switches generated by an MTW adjustment algorithm with the same parameters. The names of feedback groups are known to the whole network, and a source specifies which group it wishes to join when it establishes a connection. Each switch represents a feedback group by the parameters to the group's algorithm, the group name, and a list of the current members of the group using that switch.

All sources sharing a feedback group do not necessarily share a route. At any switch, all members of the same group that are routed through the switch receive the same feedback. For example, the sources in two video conferences, one from Chicago to St. Louis, and one from New York to Los Angeles, may share the same feedback group, but they are unlikely to share more than a few switches. A switch used by a circuit in either conference or both will use the same MTW adjustment algorithm for the group serving the video conference, but each switch's knowledge of the group's membership is limited to the members using that switch.

When feedback groups are in use, each switch maintains an MTW for each group, and manipulates that MTW using the algorithms described in Chapter 3. The switch uses a set of parameters to tune its instantiation of the MTW adjustment algorithm for each group. These parameters are the percentage of buffers that must be in use before the switch reduces the group's MTW (the congestion threshold), the additive increase unit and multiplicative decrease factors to use in adjusting the group's MTW (the AIU and MDF), and a feedback group to which feedback generated by this group's instance of the MTW adjustment algorithm will be delivered (the destination group). Each group also specifies whether switches running the group's algorithm consider the total queue length of all sources using the switch (a global threshold) or just the queue lengths of members of the group using the switch (a local threshold) when determining the group's MTW size. This array of parameters provides a highly customizable feedback system.

Setting each group's congestion threshold differently allows some groups to keep their queue occupancy low, while allowing others to keep higher queue occupancy. The threshold is given as a percentage of buffers, rather than an absolute number, to allow the group size to vary easily. When the given percentage of buffers are full, the switch sends a lower MTW to the sources in the group, otherwise the source sends a larger MTW after a monitoring interval elapses. Sources interested in low per–packet delay or in reserving buffer space to avoid losses will set a low congestion threshold. This parameter has a different effect when a group uses a local threshold, as opposed to a local threshold. The choice of global or local threshold determines what set of buffers is monitored, and the value of the congestion threshold determines how full that set of buffers must be before the MTW is reduced.

There is a subtle difference between a congestion threshold, which is an element of the selective feedback system, and a buffer share, which is an element of the resource allocation system. A buffer share represents the portion of a switch's buffers devoted to a source's traffic. A congestion threshold is the level of occupancy of those buffers that will cause the switch to reduce the MTWs of members of a group. The buffer share is resources allocated to a source, the congestion threshold is an occupancy level that indicates congestion.

Adjusting the AIU and MDF determines how aggressive a source is about assuming that the network has capacity for burstiness. A group with a high AIU, compared to its MDF, will open its time windows quickly and close them slowly. Sources in such a group assume that there will be room in an uncongested network for a large increase in burstiness. A large MDF, compared to the AIU, indicates a group that is willing to cut its burstiness sharply to avoid congestion losses. The general scaling of these parameters depends on how often the network state is likely to change. A static network will use small values of each, so that the time windows oscillate tightly about a stable point. Larger values of these parameters can be useful in more frequently changing networks.

The destination group is used to route feedback from a group that cannot adjust its time window, to one that can. A source's time window determines its maximum burst size, as well as the maximum jitter its traffic will experience in the network (see Appendix A). Some sources must bound these quantities, and therefore maintain constant time windows. DTW provides the destination group parameter to allow this. If congestion is detected in a group that is preserving its time window, it is assumed that the group to which the feedback is routed is willing to reduce its burstiness to avoid network congestion caused by members of the group preserving their time windows. This can be an effective way to smooth bulk traffic, like USENET news, to preserve the performance of real–time traffic, like video. The destination group parameter provides flexibility in routing feedback.

Whether a switch uses a local or global threshold depends on how much a group can afford to depend on the opportunistic nature of the buffer allocation. Another factor is the effect of congestion caused by other groups on the traffic of the group in question. Sources that are very loss–sensitive will use information based only on a local threshold, since they must be cautious in using unallocated buffers. Other sources may view congestion as a phenomenon that depends on all traffic to a switch, and react to it regardless of its cause. Such sources will use a global threshold.

These parameters allow for the specification of a range of adjustment algorithms. Although there is a great temptation to define many groups for a given network, establishing feedback groups has a cost to the switch. The processor on the switch must run an MTW adjustment algorithm for each group configured. In many cases this will be the determining factor of how many groups a network may have. It is better to have a few groups serving well defined types of traffic, than a group configured for each source.

7.2.3. Integration of Methods

In order to specify an effective service tailoring strategy, both of the mechanisms described above must be used. Even simple strategies, for which one mechanism or the other seems sufficient, frequently show dramatic improvement in simulations when both mechanisms are used. Some surprisingly simple strategies depend on both subsystems being present.

Consider a source that wishes to avoid loss by receiving a large buffer share allocation. If the source obtains that share without the switch adjusting its feedback to the source to leave some of the source's new allocation empty, the source will not see as large an improvement in loss protection as possible. If the switches are sending feedback as though the source is maximizing throughput, the additional buffers will simply be full most of the time. When a surge in traffic appears, as for a new source beginning to send data, the loss avoiding source has the same amount of free buffers as any other source, and experiences similar losses. On the other hand, if switches are aware that the source is avoiding loss by its presence in a loss avoidance feedback group, they will send feedback designed to leave some of the source's buffer allocation empty. These empty buffers now form a safety margin. The allocation of additional buffers allows the source to smooth its traffic to a similar degree as other sources, and still have extra buffers available in the event of a traffic surge. The selective feedback allows the source to specify that it wants to use the extra buffers as a safety factor. The two mechanisms mesh to provide a more effective loss avoidance service than either alone could.

DTW's service tailoring algorithms are simple to describe, but form a general mechanism for providing service to sources that meets their requirements. Resource allocation at switches adjusts the service that traffic receives at switches. Selective feedback adjusts source burstiness to make the best use of those resources to meet the source's requirements. The system is not tied to any particular metric, and can implement many known types of service, and new types of service for future applications. Chapter 8 discusses simulation case studies that show the power and versatility of DTW service tailoring. The remainder of this chapter discusses how minor changes to WFQ can provide a zero–allocation virtual circuit to deliver cells on a best effort basis.

7.3. Best effort Traffic

Under DTW, sources are defined in terms of their sending rates and time windows so the network can provide them with service based on those parameters. Some sources would prefer to send traffic into the network arbitrarily rather than pay the overhead of a full DTW connection. With minor changes to the DTW algorithms, DTW can support such traffic.

The changes need to be made in the resource allocation algorithms. We relax the constraints that s_i and b_i, source i's service share and buffer share, must be greater than zero. A circuit with a zero service share is defined to have an infinite virtual clock, and a zero increment. Similarly, a circuit with a zero buffer share has no buffering reserved.

A circuit with a zero service share and a zero buffer share reserves no resources at switches, and, assuming that the VCI space is sufficiently large, can always be established. Such a circuit is a zero–cost, or best effort circuit. It allows a source to send traffic without reserving resources, but without any of the benefits conferred by DTW.

Such a circuit may be acceptable for unreliable datagram traffic, and has the advantage that it may be established quickly, since no confirmation of the resource reservations are necessary.

When all switch queues serving circuits with a non–zero service share are empty, the switch may serve cells from best effort circuits. Cells queued for best effort circuits are always discarded before cells queued for standard virtual circuits, even if there are standard virtual circuits that are using more than their buffer share. In short, the traffic is excluded from having any effect on DTW traffic whatsoever. Best effort circuits receive no feedback.

The zero service and buffer shares also suggest possibilities for the creation of interesting traffic classes. For example, a source that makes a zero service allocation, but makes a non–zero buffer allocation would be able to store its traffic in switches, while that traffic is forwarded only as extra switching capacity is available in the network. A source with a non–zero service share but a zero buffer share receives a share of service, and therefore a low delay for cells that are delivered, but can be preempted by traffic from other sources.

True best effort traffic has no guarantees whatsoever associated with it. In the simulations described in Chapter 5, it was not uncommon for switches to have utilizations of 95% or higher. In such a harsh environment, traffic losses of best effort traffic would be high, and the class may not be useful. Despite all of these drawbacks, a common criticism of systems like DTW that require negotiation of source parameters and subsequent resource reservation is that sources cannot simply send and hope that their traffic gets through. This best effort class provides sources with that choice, without disturbing sources that use DTW.

Chapter 8

Service Tailoring Case Studies

General propositions do not decide concrete cases. The decision will depend on a judgement or intuition more subtle than any articulate major premise''

— *Oliver Wendell Holmes, Jr.,* **Lochner v. New York, 198 U.S. 45, 78**

This chapter continues the exploration of service tailoring begun in Chapter 7. We present three case studies of service tailoring configurations, each designed to adapt network service to meet different source requirements. We consider sources that minimize network delay, sources that reduce their losses, and sources that maintain constant time windows. In each case, we explain the configuration of the subsystems used to deliver that service. We then present simulation results that show the effectiveness of each subsystem, and the effectiveness of the integrated subsystems. These subsystems are the resource allocation and selective feedback systems described in Chapter 7.

8.1. Simulation Parameters

Like Chapter 5, much of this chapter consists of reporting simulations. The relevant parameters are given for each simulation, but there are some that are the same across all simulations. The parameters that were constant for all simulations in Chapter 5 are constant for all simulations in this chapter, except where explicitly stated otherwise. See section 5.2 for the specific values. We also use the policy of randomly picking a set of r

values for a given experiment and using the same values for comparisons within that experiment. In this chapter this means that the simulations of each subsystem and the integrated system all share the same randomly picked source ratios for the entire study of a tailoring method.

8.2. Delay Reducing Sources

The goal of delay reducing sources is to send traffic from source to destination with the lowest delay possible. A source wants to minimize delay so that some entity at the sink can react to the events occurring at the source as though the events were occurring at the sink. The events are the data being transmitted from source to sink, for example, frames of video, or alarm indications. Delay reducing sources include real–time audio or video conferences, network games, distributed control systems, and remote login sessions.

8.2.1. Service Tailoring for Delay Reducers

Delay reducers want to reduce network queueing delay while maintaining low source delay. Two ways for sources to reduce their per–cell queueing delay are to increase the rate at which their traffic is served in the switches, and to reduce their traffic's queue length at the switches. Increasing the rate at which the source's traffic is served reduces queueing delay directly. Reducing the source's queue length at a switch reduces each cell's delay, since fewer cells will be in the source's queue in front of it.

Maintaining a low source delay requires a source's time window to be as large as possible. The larger the time window the less a source has to smooth its traffic, and the lower its source delay will be. Selective feedback allows a source to request that the switches modulating its time window only consider the resource usage of the source's group when adjusting the group's MTW by selecting a local threshold. This allows a source whose group is not overusing its allocation of switch resources to reduce its source delay. A local threshold causes the time windows of the group to more strongly reflect the group member's usage of resources than the overall usage of switch resources. This distinction makes selective feedback useful to delay reducers in reducing their source delay.

Increasing the service rate of a source's traffic at the switch is a straightforward application of resource allocation. Delay reducing sources specify higher service shares when they establish their connections. This results in the delay reducers getting a higher guaranteed rate at the switch than other traffic.

Adjusting the queue length at the switches must be done via feedback. The delay reducers are assigned to the same feedback group. This feedback group is configured so that switches reduce the group's MTW when the queue length in the group's buffers passes a threshold. Since selective feedback can take into account a group's buffer usage as opposed to the aggregate queue length of the switch, it is effective in reducing that group's queue length. Adjusting time windows based on a global threshold not permit the differentiation. Using a local threshold may also have the beneficial effects on source delay mentioned above.

Each of these measures can reduce network delay when used independently. We take the further step of integrating them. The following section describes experiments exploring the effectiveness of this strategy.

8.2.2. Simulation Studies of Delay Reduction

The proposed delay reduction strategy was studied through simulation. We simulate each subsystem separately, and then combine the subsystems, to provide a more effective integrated system.

8.2.2.1. Delay Reduction by Resource Allocation

The following simulations used the configuration in Figure 8.1 and the parameters in Table 8.1. Basic sources and delay reducing sources send through both switches. Delay reducing sources implement the policies described for delay reduction. All service share and threshold values for sources using both switches are the same at both switches. Cross traffic sources use the same parameters as basic sources. Since service shares must sum to one, basic sources use reduced service shares when delay reducers use increases service shares. All sources use the packet train sending process described in Chapter 5, with the parameters given in Table 8.1. All simulations of loss avoidance use the

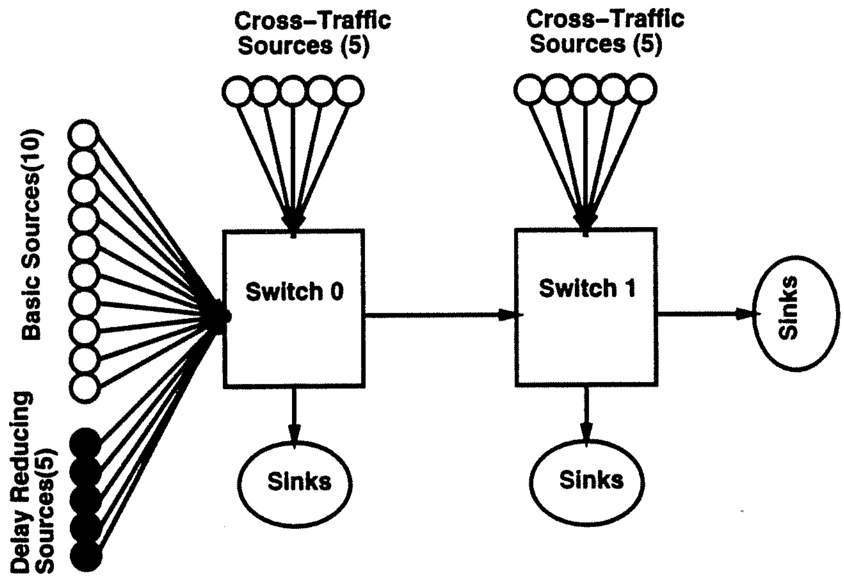

Figure 8.1 : Simulation Configuration for Delay Reduction (Resource Allocation)

parameters above unless explicitly mentioned in the experimental description. All statistics given are only for sources sending through both switches.

In the experiment summarized in Table 8.2, the delay reducing sources implemented their policy by negotiating a higher service share than the other sources. Experiments are presented that used three different service shares.

In these experiments, the service shares always sum to be at most 1, but many sources have not negotiated a guaranteed minimum rate equal to their average rate. A source with less than a 0.05 service share has not negotiated its average rate at the switch. This violates one of the preconditions of Theorem 4, described in Chapter 4. In this sense, the simulations are overbooked, but no violations of DTW stability were observed. This is another case where random traffic does not realize worst case behavior. It is important that in this simulation, and those that follow, the system is operating at the edge of stability, and still performing well.

Parameter	Value
Switch Buffering	25 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	3.2 Mb
Mean Burst Size	1.6 Mb
Mean Intertrain Time	16 ms
Delay Reducers Threshold	0.90
Delay Reducers Service Share	varies
Delay Reducers Buffer Share	0.05
Basic Sources Threshold	0.90
Basic Sources Service Share	varies
Basic Sources Buffer Share	0.05
Duration	500 seconds

Table 8.1 : Simulation Parameters for Delay Reduction (Resource Allocation)

The values reported in Table 8.2 are defined the same way as the values reported in Chapter 5. From Table 8.2, we see that allocation of additional switch processing power to some sources reduces their delays, while increasing the delay of others. Source losses and delays do not increase appreciably when resource allocation is used to reduce delay. This is because the feedback is based on the aggregate queue length at the switches in these experiments. Reallocating service shares changes only the order cells are sent from a switch, not the maximum queue length at that switch. There may be secondary effects at the second switch, but they are small. Both the average network and average total delays are reduced by increasing source service shares.

Resource allocation provides a way to allocate the average delay between groups. The basic tradeoff is that to improve one group's performance, another group's must be degraded. Assuming such a tradeoff is acceptable, resource allocation is an effective way to provide sources with a reduced delay.

Source Type	Service Share	Source Delay (msec)	Network Delay (msec)	Total Delay (msec)	Source Losses (pct)	Network Losses (pct)	Through–put Mb/sec
Basic	0.05	8.24	34.7	42.9	5.75	0.00295	47.3
D. red.	0.05	8.35	34.8	43.2	5.86	0.00265	47.3
Basic	0.04	8.39	35.7	44.1	5.92	0.00274	47.2
D. red.	0.075	8.49	30.6	39.1	5.96	0	47.2
Basic	0.03	8.14	35.9	44.0	5.69	0.00284	47.3
D. red.	0.10	8.18	30.6	38.8	5.83	0	47.3

Table 8.2 : Summary of Delay Reduction by Resource Allocation

8.2.2.2. Delay Reduction by Selective Feedback

The following simulations used the configuration in Figure 8.1 and the parameters in Table 8.3.

Table 8.4 summarizes a set of experiments in which the queue lengths of the delay reducing sources are kept low by selective feedback. The feedback threshold, reported in the ''threshold'' column, is the fraction of the buffers allocated to a group that must be filled for the switch to reduce that group's MTW. In these simulations, a group's feedback is based on the number of buffers in use by the group, not those of the entire switch.

Selective feedback has some positive effects on the network, but is ineffective at reducing the total delay. Table 8.4 shows that it reduces the delay at the switches (network delay), but has the negative effect that source delays for delay reducers are significantly higher. The lower threshold for decreasing the group's MTW causes more negative feedback, which reduces the source time windows so much that the smoothing

Parameter	Value
Switch Buffering	25 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	3.2 Mb
Mean Burst Size	1.6 Mb
Mean Intertrain Time	16 ms
Delay Reducers Threshold	varies
Delay Reducers Service Share	0.05
Delay Reducers Buffer Share	0.05
Basic Sources Threshold	0.90
Basic Sources Service Share	0.05
Basic Sources Buffer Share	0.05
Duration	500 seconds

Table 8.3 : Simulation Parameters for Delay Reduction (Selective Feedback)

Source Type	Thresh– old	Source Delay (msec)	Network Delay (msec)	Total Delay (msec)	Source Losses (pct)	Network Losses (pct)	Through– put Mb/sec
Basic	0.90	8.19	34.4	42.6	5.68	0.000446	47.3
D. red.	0.90	9.86	34.3	44.2	6.82	0.000553	46.8
Basic	0.90	7.45	34.4	41.9	5.28	0.000222	47.5
D. red.	0.65	12.3	33.8	46.1	8.03	0.000159	46.3
Basic	0.90	6.86	34.5	41.4	4.94	0.00514	47.6
D. red.	0.475	14.5	33.5	48.0	8.96	0.00119	45.9

Table 8.4 : Summary of Delay Reduction by Selective Feedback

delays at the sources dominate the reduced network delays. Source losses become high enough to reduce the delay reducers' average throughput appreciably.

Comparing the the first two lines of Table 8.4 with Table 8.2, it is apparent that just placing sources into groups has an effect on their performance. This is because grouping traffic and basing their feedback on fewer buffers causes sources to be less tolerant of buffer sharing between groups. Each group can only share buffering among sources in that group, as opposed to being able to share all switch buffers.

In this case, selective feedback seems ineffective. However, when integrated with resource allocation, the combination provides better service than resource allocation alone.

8.2.2.3. Delay Reduction by Integrated Methods

The following simulations used the configuration in Figure 8.1 and the parameters in Table 8.5. This experiment is a straightforward combination of the two previous experiments. Delay reducers both have higher service shares, and belong to a feedback group with a lower threshold. Simulations of this policy are summarized in Table 8.6

Parameter	Value
Switch Buffering	25 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	3.2 Mb
Mean Burst Size	1.6 Mb
Mean Intertrain Time	16 ms
Delay Reducers Threshold	varies
Delay Reducers Service Share	varies
Delay Reducers Buffer Share	0.05
Basic Sources Threshold	0.90
Basic Sources Service Share	varies
Basic Sources Buffer Share	0.05
Duration	500 seconds

Table 8.5 : Simulation Parameters for Delay Reduction (Selective Feedback)

Source Type	Service Share	Thresh–old	Source Delay (msec)	Network Delay (msec)	Total Delay (msec)	Source Losses (pct)	Network Losses (pct)	Through–put Mb/sec
Basic	0.05	0.90	8.19	34.4	42.6	5.68	0.000447	47.3
D. red.	0.05	0.90	9.86	34.3	44.2	6.82	0.000553	46.8
Basic	0.04	0.90	10.4	35.1	45.5	6.91	0.00120	46.7
D. red.	0.075	0.65	5.63	30.5	36.1	4.15	0	48.0
Basic	0.03	0.90	10.4	35.0	45.4	6.99	0	46.7
D. red.	0.10	0.475	5.63	30.5	36.1	4.14	0	48.0

Table 8.6 : Summary of Delay Reduction by Integrated Methods

Combining the two subsystems results in a system with better performance than either alone. In fact, the combined system is so much more effective than either subsystem, that the combined system has improved performance by as much as it is able to when the subsystems are each using parameters that were not optimal for the subsystem.

The higher service shares of the delay reducers make the more cautious selective feedback thresholds effective, because the delay reducers' queue lengths are kept lower by the higher service rate that their traffic is getting at switches, rather than by overly restrictive time windows. There was no selective feedback in the resource allocation experiment, so time windows of sources did not reflect the lengths of their group's queues, but of the switches' aggregate queues.

Selective feedback gives sources the ability to detect the effect their group's traffic is having on their queue lengths at the switch. When selective feedback is used alone as in the previous experiment, it forces the delay reducers to use less switch buffering by smoothing their traffic. When delay reducer's service shares are high, their queue lengths go down as a matter of course. In this network, selective feedback is sensing an already low utilization of buffers and allowing the delay reducing sources underusing their allocation to be burstier. The addition of the resource allocations changes the role of the selective feedback from constraining burstiness to create low queue length to freeing time windows as an indication of slack in the network.

Used with an appropriate resource allocation, selective feedback results in a reduction in source delay, while maintaining the reduction in network delay that the resource allocation created. Each subsystem is effective in reducing a component of the end–to–end delay, resulting in a more powerful system.

8.3. Loss Avoiding Sources

Loss avoiding sources seek to send their traffic with as few losses as possible. This type of source is generally trying to get a sizable amount of information across the network in a small time, without errors. Frequently, delay reducing sources are willing to lose some percentage of their cells to insure timely delivery of the rest. Loss avoiders are willing to accept a higher delay in return for lower loss rates. File transfer applications, transaction systems, and remote procedure call packages are sources that need to avoid loss.

Since loss avoiders must get their data to the sink without error, they frequently rely on retransmission strategies to resend lost cells. Resending data requires an additional round trip delay, so loss avoiders are willing to pay in queueing time to avoid resending time.

Some applications implement error–free delivery through forward error correction. This amounts to sending redundant data in such a way that if a certain percentage of the data is lost, the sink can reconstruct the missing data from the redundant information. The cost of forward error correction is the extra information sent. If enough data is lost, such sources must resort to retransmission. It is in these sources' best interest to keep their network losses low so that their forward error correction remains effective. If it fails, these sources are reduced to the case above.

8.3.1. Service Tailoring for Loss Avoiders

Like delay reducers, loss avoiders can use both of the service tailoring mechanisms to achieve their goal. Loss avoiders can use resource allocation to acquire a larger share of buffers at the switches. Since such sources are entitled to more switch buffers, the chance that one of their cells will not have a buffer reserved for it during times of switch congestion is lower. Another option is to use selective feedback to reduce the occupancy of the buffers that a source has allocated to it at a switch. In this case, some of the source's buffers will be empty under stable traffic, allowing the source's cells to use the empty buffers in times of congestion.

Acquiring additional buffering is a simple application of resource allocation. Unlike redistributing service shares, this allocation cannot upset DTW stability, even theoretically.

Lowering the congestion threshold of a feedback group devoted to loss avoiders will allow members of that group to keep some of their buffer allocation in reserve. Of course, this will force the loss reducers to smooth their traffic more strictly than other sources and may subject them to additional smoothing delays. More importantly, in longer periods of congestion, loss avoiders will have no more buffers allocated to them than any other sources, so this method is less likely to be effective if the network is prone to long bouts of congestion.

These two systems can be combined. In this case, loss avoiders will acquire a larger pool of buffers at each switch, but only use some fraction of them in uncongested times. The remainder is used to hold traffic that would otherwise be lost when congestion appears. Resource allocation provides the buffer space, and selective feedback regulates its use.

8.3.2. Simulation Studies of Loss Avoidance

The approach taken to studying loss avoidance is similar to the one taken in the previous section on delay reduction. We have simulated using each subsystem to avoid losses, and compared these simulations to the simulation of the integrated system.

8.3.2.1. Loss Avoidance by Resource Allocation

The following simulations used the configuration in Figure 8.2 and the parameters in Table 8.7.

All sources not indicated as cross traffic sources are sending through both switches. Shaded sources using both switches are loss avoiding sources and implement the loss avoidance policies described in each simulation. Other sources using both switches are called basic sources, and implement no particular service tailoring method. Cross traffic uses basic source parameters. All sources using both switches make the same resource reservations at both switches. As in the simulations of delay reducers, only statistics for the sources using both switches will be presented.

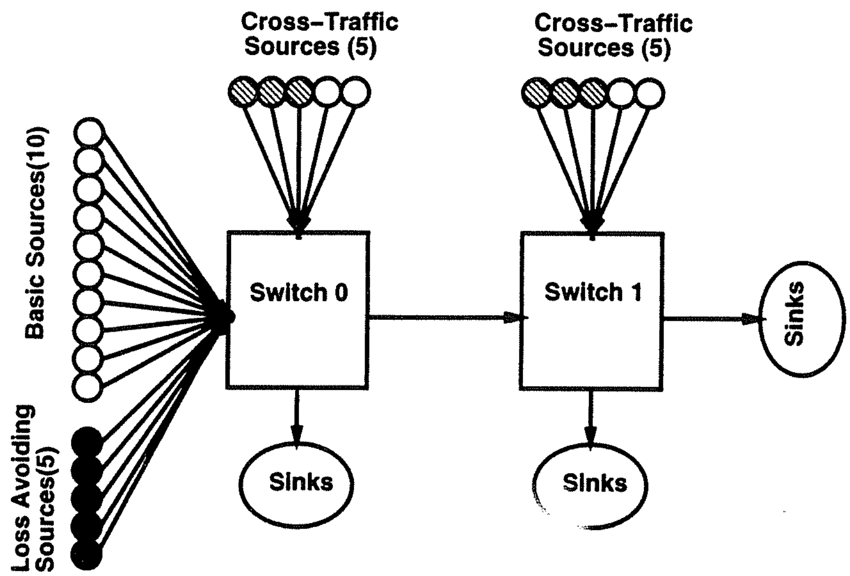

Figure 8.2 : Simulation Configuration for Delay Reduction (Resource Allocation)

Parameter	Value
Switch Buffering	10 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	5.0 Mb
Loss Avoiders Threshold	0.90
Loss Avoiders Service Share	0.05
Loss Avoiders Buffer Share	varies
Basic Sources Threshold	0.90
Basic Sources Service Share	0.05
Basic Sources Buffer Share	varies
Duration	500 seconds

Table 8.7 : Simulation Parameters for Delay Reduction (Resource Allocation)

In the loss avoidance simulations, the network is configured to induce as many losses as possible. All sources use the worst case traffic model described in Chapter 5, *i.e.*, they send a packet whenever they are permitted to by the source control. Furthermore, the striped cross traffic sources follow a cycle of being idle for fifteen seconds and sending worst case traffic for five seconds. This destabilizes the network by having it change state throughout the simulation. This network configuration will be used for all simulations of loss avoiding sources.

Causing significant losses in a DTW network is difficult. The per–source losses reported in this section are about an order of magnitude higher than those reported for static networks in Chapter 5 (Table 5.17), and considerable disruption to the network had to be caused. Even under these challenging conditions, without loss avoidance, DTW reports roughly the same losses at the packet feedback system simulated in Chapter 5 in the static configuration reported in Table 5.18.

The first policy simulated is the straightforward application of resource allocation to loss avoidance. Sources avoiding loss allocate more buffers than basic sources. We ran three simulations, increasing the buffer shares of the loss avoiders in each one. Table 8.8 summarizes these simulations.

We report only network losses and throughputs for these experiments. The worst case source model presents the regulator with a constant stream of packets, and allows the regulator to discard those it cannot send. This ensures that a packet is always sent when it can be. This means that all source losses and delays are as large as they can be

Source Type	Buffer Share	Network Losses (pkts)	Through–put Mb/sec
Basic	0.05	2931	49.9
Loss Avoiders	0.05	3278	49.9
Basic	0.04	4320	49.9
Loss Avoiders	0.075	979	50.0
Basic	0.03	7160	49.9
Loss Avoiders	0.10	405	50.0

Table 8.8 : Summary of Loss Avoidance by Resource Allocation

for all simulations.

Table 8.8 shows that resource allocation provides protection for loss avoiders. The sources with higher buffer allocations lose fewer packets than basic sources. As with the other resource allocation solution described in this chapter, gains in loss protection for the source avoiding losses are paid for with reduced protection for the basic sources. Resource allocation provides a simple and effective loss avoidance strategy, if it is acceptable to penalize other traffic.

8.3.2.2. Loss Avoidance by Selective Feedback

The following simulations used the configuration in Figure 8.2 and the parameters in Table 8.9.

Table 8.10 summarizes the use of selective feedback to leave a fraction of the buffer allocations of each loss avoiding source at the switches empty as a hedge against loss. Sources all have the same buffer allocations, but are placed in feedback groups, where

Parameter	Value
Switch Buffering	10 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	5.0 Mb
Loss Avoiders Threshold	varies
Loss Avoiders Service Share	0.05
Loss Avoiders Buffer Share	0.05
Basic Sources Threshold	0.90
Basic Sources Service Share	0.05
Basic Sources Buffer Share	0.05
Duration	500 seconds

Table 8.9 : Simulation Parameters for Delay Reduction (Selective Feedback)

each switch sends feedback to the group based on the occupancy of that group's buffers. Loss avoiders have a lower feedback threshold, in order to leave some of their buffers unused except during times of congestion. We vary the thresholds to gauge the effectiveness of the strategy.

As in the case of using selective feedback alone to reduce delay, using it to avoid loss produces mixed results. The groups with lowered thresholds experience fewer losses than those with high thresholds, but the improvement is not as significant as under the resource allocation system.

Although selective feedback can avoid the congestion that causes loss, when losses occur, their magnitude is largely determined by the actual resource reservations. Loss avoiders can use their unoccupied buffers when the congested period begins, but if the congestion lasts, they run out of unused buffers and have to face losses as well. This mechanism would be even less effective in a network without bounded congestion times.

Selective feedback can be used to avoid loss in networks with short congestion periods. We combine this mechanism with resource allocation in the next study.

Source Type	Thresh–old	Network Losses (pkts)	Through–put Mb/sec
Basic	0.90	2852	49.9
Loss Avoiders	0.90	2607	49.9
Basic	0.90	3509	49.9
Loss Avoiders	0.65	2416	49.9
Basic	0.90	4111	49.9
Loss Avoiders	0.475	2012	49.9

Table 8.10 : Summary of Loss Avoidance by Selective Feedback

8.3.2.3. Loss Avoidance by Integrated Methods

The following simulations used the configuration in Figure 8.2 and the parameters in Table 8.11.

Table 8.12 summarizes the experiments using the combined service tailoring. In these experiments, both subsystems are in use. A larger buffer share is allocated to each loss avoiding source, to provide more buffering to the source in times of congestion. The selective feedback thresholds are set to keep buffers unused in uncongested operation.

Parameter	Value
Switch Buffering	10 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	5.0 Mb
Loss Avoiders Threshold	varies
Loss Avoiders Service Share	0.05
Loss Avoiders Buffer Share	varies
Basic Sources Threshold	0.90
Basic Sources Service Share	0.05
Basic Sources Buffer Share	varies
Duration	500 seconds

Table 8.11 : Simulation Parameters for Delay Reduction (Integrated Methods)

Source Type	Buffer share	Thresh –old	Network Losses (pkts)	Through– put Mb/sec
Basic	0.05	0.90	2853	49.9
Loss Avoiders	0.05	0.90	2608	49.9
Basic	0.04	0.90	3803	49.9
Loss Avoiders	0.075	0.65	1313	50.0
Basic	0.03	0.90	1791	49.9
Loss Avoiders	0.10	0.475	304	50.0

Table 8.12 : Summary of Loss Avoidance by Integrated Methods

Table 8.12 shows that the combined subsystems are effective. Losses are cut dramatically and consistently. The combination of subsystems is more effective than either alone. Selective feedback maintains the unused buffer pool for the loss avoiders that is effective during brief congestion periods, and the resource allocation reduces their losses over long congestion periods.

Although the integrated system consistently reduces losses for the loss avoiders, its effect on the basic sources is less uniform. As we might predict, the improvement in loss avoidance from the first to second line in Table 8.12 is paid for with a decrease in the loss avoidance of the basic sources. However between the second and third lines of the same table, the loss avoidance of the basic sources improves. As the occupancy of the buffers of the loss avoiders is reduced by the selective feedback, the opportunistic buffer allocation algorithm allows the basic traffic to make use of the unused buffer pool when the loss avoiders are not using it. When the occupancy of these buffers is low enough as in the simulation with the lowest threshold, there is almost always some space for the basic traffic to use. This second order effect is not visible in the second simulation, since more of the smaller unused buffer pool is needed by the loss avoiders.

The combined system works well, with each system addressing one component of the desired performance. In this case, each defends against congestion losses on a different time scale, and together they reduce the total losses.

8.4. Constant Time Window Sources

Constant time window (CTW) sources are sources that do not vary the size of their time window. They may not have sufficient buffering or intelligence to smooth their traffic. Since the time window is related to the bounds on jitter and delay (see Appendix A), sources that must have bounds on those quantities can achieve them by choosing and maintaining a constant time window. Examples of constant time window sources include low intelligence peripherals, like cameras. Other real time sources may exhibit this property, especially those sending constant sized bursts like video coders.

DTW's congestion avoidance system relies on sources adapting their burstiness to changing network state. Switches serving constant time window sources adjust other willing sources' burstiness when they detect congestion among the CTW sources. This approach is effective as long as there are enough sources changing their time windows to offset those sources that are remaining constant. The sources that receive additional feedback to offset the CTW sources' contribution to network congestion are designated by their membership in an appropriate feedback group. This brand of service tailoring allows some sources to step outside DTW's feedback system, but it is most effective when only a few sources do so.

DTW stability is unaffected by sources maintaining constant time windows.

8.4.1. Service Tailoring for CTW Sources

Since CTW sources are seeking to maintain a condition at the regulator, namely a constant time window, rather than at the switches, changing their resource allocation at switches is of no use. These sources rely on a redirection of feedback by switches to some other willing group to avoid congestion in the network. An example of sources that may be relied upon to accept such feedback are bulk mail sources or USENET news feeds. The information such applications are propagating does not need to meet hard performance constraints. Since CTW sources will be exhibiting a fixed level of burstiness, the total network burstiness will be reduced by reducing the burstiness of the willing group.

Although CTW sources cannot use resource allocations at the switch to keep their time windows constant, they make resources allocations. Their cells need to be served and buffered, so they must negotiate service and buffer shares. The switch will monitor their queue lengths, and adjust source time windows based on their resource utilization. However, the sources that will have their time window adjusted will be in a different feedback group than the CTW sources.

All CTW sources are put in a feedback group that has its destination group parameter set to a group willing to respond to CTW group feedback. No feedback is directed to the CTW sources. Messages to decrease the MTW of the willing group based on the buffer utilization of the CTW group are routed to the willing group, and no message to increase the MTW of the willing group based on the buffer utilization of the CTW group is ever generated. All other feedback is sent normally.

Notice that allowing sources to retain constant time windows does not entitle them to any priority for their traffic once it has entered the network. Constant time window sources continue to avoid congestion by having switches sense an overbooking of their buffers and sending feedback to a willing group. Without additional resource allocations, constant time windows sources have the same chance for losses or delay as other sources.

Of course, constant time window sources that wish to use resource allocations to avoid loss or reduce delay may do so. They may also adjust the feedback thresholds of their redirected feedback.

8.4.2. Simulation Studies of Constant Time Windows

The following simulations used the configuration in Figure 8.3 and the parameters in Table 8.13.

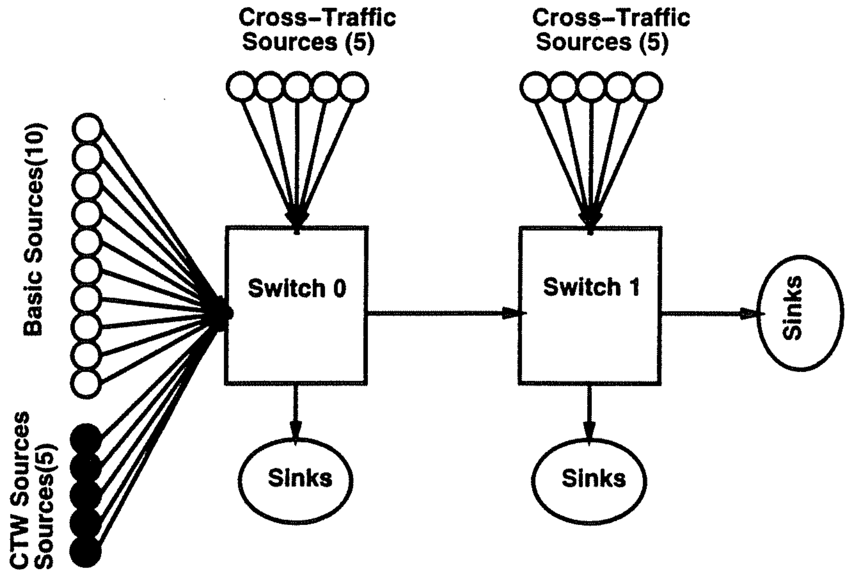

Figure 8.3 : Simulation Configuration for CTW experiments

Parameter	Value
Switch Buffering	25 Mb
Source Average Rate	50 Mb/sec
Source Peak Rate	100 Mb/sec
Regulator Buffering	1.6 Mb
Mean Burst Size	0.8 Mb
Mean Intertrain Time	16 ms
CTW Sources Threshold	varies
CTW Sources Service Share	0.05
CTW Sources Buffer Share	0.05
CTW Sources Time Windows	0.50
Basic Sources Threshold	0.90
Basic Sources Service Share	0.05
Basic Sources Buffer Share	0.05
Duration	500 seconds

Table 8.13 : Simulation Parameters for CTW experiments

The studies of constant time window sources reported here show that redirection of feedback is effective in controlling congestion while allowing sources to maintain constant smoothing of their traffic. As before, five of the sources using both switches were picked as constant time window sources and placed in the CTW feedback group. All other sources were used as willing recipients of the CTW group's feedback, and are called basic sources. The simulation results reflect only sources sending through both switches. The feedback recipients were receiving feedback based on the occupancy of their group's buffers, and any messages to reduce the time window generated by a switch for the CTW group. All sources negotiate the same resource allocations with both switches. All sources are packet train sources, with the parameters in Table 8.13. Table 8.14 summarizes the experiments.

In the ''CTW'' rows are the results for the constant time windows sources. Network losses were all zero, and are not included. Notice that despite the change in feedback threshold, the CTW sources maintain a constant time window. This is reflected in the constant values of source loss and source delay. The other sources adapt to the network state and avoid congestion. The network continues to show the positive effects of DTW despite the fact that sources are maintaining constant time windows.

Redirecting feedback provides a mechanism for sources that cannot dynamically smooth their traffic to make use of a DTW network. This provides an important

Source Type	Thresh– old	Source Delay (msec)	Network Delay (msec)	Total Delay (msec)	Source Losses (pct)	Through– put Mb/sec
Basic	0.90	2.27	33.8	36.1	0.00326	48.4
CTW	0.90	4.12	33.5	37.6	0.00579	47.2
Basic	0.90	3.48	32.7	36.2	0.00491	47.6
CTW	0.65	4.12	32.7	36.8	0.00579	47.2
Basic	0.90	4.44	32.2	36.6	0.00606	47.1
CTW	0.475	4.12	32.3	36.4	0.00579	47.2

Table 8.14 : Summary of Constant Time Window Experiment

mechanism that allows sources that must fix their burstiness to make use of a DTW network.

8.5. Summary of Service Tailoring

Chapter 7 described the resource allocation and selective feedback subsystems used for service tailoring. This chapter presented several case studies that show how these subsystems can be used to provide different types of service. Specifically we have demonstrated that these two systems can be used to provide sources with lower average per–cell delays, and lower average loss rates. These are two types of service that are of great importance to traffic today. We have also demonstrated a way to allow sources that cannot dynamically smooth their traffic to reap the benefits of DTW.

Although the subsystems defined in Chapter 7 are useful by themselves, integration produces a more powerful system than either separately. In some cases, the integration causes one of the subsystems to assume a different role in the service provision, like selective feedback reducing source delay in the delay reduction experiments. Integration can also be a simple combination of the two systems strong points, as in the case of loss avoidance. In either case integrating the systems provides better service than the subsystems themselves.

These case studies demonstrate the flexibility and power of the two subsystems, and their effectiveness when integrated. We have proven that they are useful in providing the types of service that today's traffic requires. More types of service can be constructed from the building blocks provided by DTW. The robustness of the service tailoring system will enable it to be used in tomorrow's networks as well as today's.

Chapter 9

Conclusions and Future Work

''Only on the edge of the grave can man conclude anything.''

— *Henry Brooks Adams,* **The Education of Henry Adams, ch. 6**

This thesis described the Dynamic Time Windows congestion control and avoidance system, and explored its performance through analysis, simulation and implementation. DTW is unique in its approach of directly controlling source burstiness to control and avoid congestion in high speed networks. By combining the best parts of allocation congestion control and avoidance systems and feedback congestion control and avoidance systems, DTW is effective in environments where these systems fail. DTW adapts to changing network state to use resources more efficiently than allocation systems. Because it decouples congestion control and avoidance, DTW performance does not degrade in the face of a high bandwidth–delay product, as feedback systems do. In addition to avoiding and controlling congestion, DTW allows sources to tailor network performance to meet their applications specific needs.

Analytically, we have shown that using DTW's source control algorithms, which are based on controlling source burstiness using the time window criterion, produce a network with bounded congestion times. This property, DTW stability, is extended to multiple switches by the addition of Weighted Fair Queueing to the switches. We show that a system using WFQ and DTW distorts traffic only as much as the bottleneck switch, and we are able to bound this distortion. DTW stability allows congestion control to be divorced from congestion avoidance. Switch queues empty in a predictable, bounded time without feedback. This allows DTW to operate in networks where a high bandwidth–delay product makes conventional feedback based systems unstable.

We have also explored the relationship between DTW source control and Leaky Bucket source control. Both enforce an average rate in the limit. DTW source control realizes this limit an infinite number of times, unlike Leaky Bucket. This repeated enforcement of the average rate is shown to cause DTW stability.

Through simulation we have shown that DTW's adjustment of time windows is an effective way to utilize the network efficiently while avoiding congestion. We show that time windows oscillate about a stable point in a static network. Switches are able to sense changes in network state by observing their queue lengths, and communicate this change to sources by adjusting their time windows. This process adapts source burstiness to the changing network state. Adjusting time window sizes only changes the utilization of the network; it does not interfere with DTW stability. The decoupling of DTW stability from feedback allows DTW to use feedback to tune source burstiness to avoid congestion in the face of a high bandwidth–delay product.

Simulations were also used to compare DTW to conventional allocation and feedback congestion control and avoidance systems. DTW provides comparable or better service than these systems under steady load. We show that DTW performs better in a rapidly changing network than a feedback system, and that it senses and adapts to network state better than the allocation system.

The analysis of DTW stability for multiple switches is also confirmed by simulation. Besides confirming the analysis, these experiments showed how difficult it is to force traffic to realize a worst case. In most cases the stability bounds are more constraining than necessary. To take advantage of this slack, heuristic bounds on a source's effective average rate may be used.

Using the XUNET network, we have prototyped a basic DTW system which shows that it can be built using today's hardware. The system avoids congestion and detects changes in network state as predicted in analysis and simulation. The implementation enabled us to observe that weighted round robin scheduling is a good approximation to WFQ, and that DTW is effective on real hardware.

DTW's service tailoring system is shown to adapt parameters of the DTW algorithms well to sources with various service requirements. We show that service tailoring can be used to reduce source's delays, avoid losses, and allow sources that cannot dynamically smooth their traffic to use DTW. By allowing sources to establish zero–cost virtual circuits, we provide a means for DTW to serve best effort traffic. We describe how the configurations of the tailoring parameters are derived, so that these descriptions may act as guides for future implementors of various service types. The service tailoring system is powerful enough to provide the service required for sources today, and flexible enough to be configured for tomorrow's traffic as well.

As described in this work, DTW is a system that approaches congestion control avoidance in a new way, by controlling source burstiness. It has shown this to be an effective method, which is the equal of conventional congestion avoidance and control systems under mundane conditions, and superior to them under extreme conditions. We believe that the conditions under which DTW excels are the conditions that will prevail in tomorrow's networks. These conditions include a high bandwidth–delay product, and changing network state. Furthermore DTW allows sources to tailor network performance to their specific requirements. This combination will makes DTW an attractive congestion control and avoidance system for coming high speed networks.

9.1. Future Work

Although this work describes many aspects of DTW, there are still open areas for research. These topics can be explored through analysis, simulation and implementation.

The bounds on source's effective average rate after passing through a switch could be probabilistically calculated for traffic sources of various distributions. We know the worst case bounds are realizable, but analysis may reveal how likely they are to be met. The analysis should be performed for several source models, and the results compared. One of the challenges is to find a model simple enough to perform a meaningful analysis upon that captures the key attributes of source behavior.

An analysis of source traffic distortion by the enforcement of the time window criterion at the source would be useful as well. We have explored this through simulation, but an analysis may provide a deeper understanding. This analysis requires modelling the regulator accurately, and then analyzing its behavior given a model of source behavior. The model of traffic leaving the regulator is necessary to the modelling of traffic at switches described above.

Models of average behavior of tailored services will also be useful. We have explored the effects of service tailoring in a very practical way, through simulation. Providing an analysis that could reasonably predict the behavior of traffic under the tailoring mechanisms would be useful. This analysis may lead to new mechanisms to provide service tailoring.

Simulation will remain a useful tool for evaluating new service tailoring mechanisms. Although analysis remains interesting, it is plagued with the difficulties of modelling source tractably, and finding models that reflect real sources. This difficulty is increased by the fact that sources are changing their behavior as networks change theirs. Simulation provides a shorter path than analysis to test algorithms. As new applications appear, it will be possible to define and test new service tailoring configurations that serve the applications through simulation.

The implementation needs to be made more robust, and tested in a more general environment under real traffic. The current system is a prototype, and is not ready for a general user. Providing a system closer to a production level will allow us to observe DTW's effect on a wide range of traffic profiles, as more people use the system. This will allow us to begin tuning those system parameters that can only be tuned by experience.

The service tailoring features need to be added to the prototype, and then to the production system. This will allow us to test DTW on a range of sources requiring tailoring. Again, this will give us experience estimating the parameters needed to tailor real sources.

Porting DTW to other environments would be instructive as well. The current implementation is tied to the XUNET hardware. It would be instructive to implement the algorithms in another system, both to confirm their robustness, and as an opportunity to optimize the code. Moving DTW to another network may also make it accessible to more users, which will provide us more opportunities to observe the system under real traffic.

Although we have demonstrated that DTW is a powerful system for controlling and avoiding congestion, there are many opportunities to study the system further, make it accessible to users, and enhance the system.

Appendix A

Delay and Jitter Bounds for DTW

The the delay and jitter of a virtual circuit's traffic under DTW can be determined directly from the time window criterion and the properties of WFQ. This appendix performs those calculations, and comments on their significance. We assume the traffic and switch models used in Chapter 4.

Let the last bit of a burst be the last instant the burst is sent or received, and define the sending and receiving time of the first bit of that burst similarly.

In order to bound the delay for a DTW network, one needs to calculate the worst case queueing delay that a burst will encounter as it crosses the network, and add that value to the propagation delay. We calculate the queueing delay of the last bit of a burst at the bottleneck switch.

Delay is a distortion of traffic by a switch, and as we saw in chapter 4, these distortions are not additive in a DTW network. Therefore only the bottleneck switch's distortion needs to be considered. In other words, after the bottleneck switch has delayed the last bit of a source's largest burst by its maximum amount, no switch will queue it. If switches before the bottleneck queue it, that reduces the delay that the bottleneck adds to it, but not the total delay.

Consider an empty switch, with a burst arriving at it from source i. Let the switch serve that burst at minimum rate that source i is guaranteed. The queue length will grow at a rate of $\hat{\lambda}-\frac{s_i\mu}{\sum_j s_j}$. Since the source can only send a burst of $\bar{\lambda}I$ bits, and that burst will be completely cleared before the source can send the next burst, due to DTW stability. The maximum queue length will be:

$$\frac{\bar{\lambda}I}{\hat{\lambda}}\left(\hat{\lambda}-\frac{s_i\mu}{\sum_j s_j}\right)$$

$$=\bar{\lambda}I\left(1-\frac{s_i\mu}{\hat{\lambda}\sum_j s_j}\right)$$

Assuming that the switch clears this queue at its slowest rate, the delay for the final cell is:

$$\frac{\bar{\lambda}I\left(1-\frac{s_i\mu}{\hat{\lambda}\sum_j s_j}\right)}{\frac{s_i\mu}{\sum_j s_j}}$$

$$=\bar{\lambda}I\left(\frac{\sum_j s_j}{s_i\mu}-\frac{1}{\hat{\lambda}}\right)$$

Bounding that delay consists of placing an upper bound on I, which can be done with a minor modification to the feedback algorithms described in Chapters 3 and 7.

The delay through the regulator must be considered as well. The maximum delay is experienced by the last cell queued in a full regulator. Every I time units $\bar{\lambda}I$ bits will be sent, and in the worst case the process will take $I(1-\frac{\bar{\lambda}}{\hat{\lambda}})$ time units to start.

In the worst case, the first bit in the regulator has no credit available to send it, and has become the first bit after the regulator has just sent a maximum sized burst at the highest speed. This means there are $\bar{\lambda}I$ credits in the credit queue, each separated by $1/\hat{\lambda}$ time units. The first credit entered the credit queue $\frac{\bar{\lambda}I}{\hat{\lambda}}$ time units ago, and since credits are restored after I time units, it will be available in $I-\frac{\bar{\lambda}I}{\hat{\lambda}}$ time units. So the total time the final bit waits in a b bit regulator is

$$I(1-\frac{\bar{\lambda}}{\hat{\lambda}}) + \left\lceil \frac{b}{\bar{\lambda}I} \right\rceil I$$

Total end–to–end queueing delay is therefore

$$I\left(1-\frac{\bar{\lambda}}{\hat{\lambda}}\right) + \left\lceil \frac{b}{\bar{\lambda}I} \right\rceil I + \bar{\lambda}I\left(\frac{\sum_j s_j}{s_i\mu} - \frac{1}{\hat{\lambda}}\right) \tag{A.1}$$

Enforcing this delay on a source requires bounding the time window and picking an appropriate sized regulator.

This bound on end to end delay, is also the bound on jitter. Consider two bursts sent so that the last bit of the first bit is sent t time units before the first bit of the second burst. where t is given by equation (A.1). If the first burst meets the maximum delay, and the second burst is not delayed, the last bit of the first burst and the first bit of the second burst will have moved together by the maximum delay. Therefore equation (A.1) is a jitter bound as well. Sh 2 "The Bounds in Perspective"

We mentioned in Chapter 7 that these bounds were of practical limited use, and we must stress that. It is easy to construct the worst case traffic as a thought experiment, but in reality, achieving those bounds requires some collusion. Unless the application has an extremely hard failure mode, it seems sensible to use predictive service. If an application absolutely requires strict bounds on delay and jitter, the solution may be to use an isolated network for that traffic. Because we believe such traffic is exceedingly rare, this solution may be the most fair to both predictive sources using the network and the application that requires the hard bounds.

References

1. R. Jain and K. Ramakrishnan, "Congestion Avoidance in Computer Networks with a Connectionless Network Layer: Concepts, Goals, and Methodology," *Proc. IEEE Symposium on Computer Networks*, pp. 134-143 IEEE, (1988).

2. R. Jain, "A Delay Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks," *Computer Communications Review* **19**(5) pp. 56-71 ACM SIGCOMM, (Oct. 1989).

3. K. Ramakrishnan and R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer," *Proc. SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 303-313 ACM SIGCOMM, (Aug 16-19 1988).

4. Jeffrey Jaffe, "Bottleneck Flow Control," *IEEE Transactions on Communications* **COM-29**(7) pp. 954-962 IEEE, (July 1981).

5. B. Sanders, "An Incentive Compatible Flow Control Algorithm for Fair Rate Allocation in Computer/Communication Networks," *IEEE Transactions on Computers* **37**(9) pp. 314-320 IEEE, (Sept. 1990).

6. A. G. Fraser, C. R. Kalmanek, A. E. Kaplan, W. T. Marshall, and R. C. Restrick, "XUNET 2: A Nationwide Testbed in Gigabit Networking," *Proc. IEEE INFOCOM*, IEEE, (May 6-8, 1992).

7. David Clark, Bruce Davie, David Farber, Inder Gopal, Bharath K. Kadaba, W. David Sincoskie, Jonathan M. Smith, and David Tennenhouse, "An Overview of the AURORA Gigabet Testbed," *IEEE INFOCOM*, pp. 569-581 IEEE INFOCOM, (May 6-8, 1992).

8. Van Jacobson, "Congestion Avoidance and Control," *Proc. SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 314-329 ACM SIGCOMM, (Aug 16-19 1988).

9. T.J. Brenners-Lee, R. Cailliau, J-F Groff, and B. Pollermann, "World Wide Web : The Information Universe," *Electronic Networking: Research, Applications and Policy* **2**(1) pp. 52-58 Meckler Publishing, (Spring 1992).

10. W. Hibbard, D. Santek, and G. Tripoli, "Interactive atmospheric data access via high speed networks.," *Computer Networks and ISDN Systems* **22** pp. 103-109 ().

11. Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala, "RSVP: A New Resource ReSerVation Protocol," *IEEE Network Magazine* **9,#4** pp. 8-18 IEEE, (September 1993).

12. A. Banerjea and B. Mah, "The Real Time Channel Administration Protocol," *Proc. 2nd International Workshop on Network and Operating System Support for Digital Audio and Video*, (November 1991).

13. J. S. Turner, "New Directions in Communications (or Which Way to the Information Age)," *IEEE Communications Magazine* **24**(4) pp. 8-14 IEEE, (Oct. 1986).

14. H. Ahmadi, R. Guerni, and K. Sohraby, "Analysis of Leaky Bucket Access Control Mechanism with Batch Arrival Process," *Proc. IEEE Globecom*, IEEE, (Dec. 2-5, 1990).

15. K. Bala, I. Cidon, and K. Sohraby, "Congestion Control for High Speed Packet Switched Networks," *Proc. IEEE INFOCOM*, pp. 520-526 IEEE, (June 5-7, 1990).

16. K. Sohraby and M. Sidi, "On the Performance of Bursty and Correlated Sources Subject to Leaky Bucket Rate-Based Access Control Schemes," *Proc. IEEE INFOCOM*, IEEE, (Apr. 7-9, 1991).

17. The ATM Forum, *ATM User-Network Interface Specification Version 3.0,* The ATM Forum, Mountain View, CA (June 1993).

18. L. Zhang, "A New Architecture for Packet Switching Network Protocols," *Ph.D Thesis*, pp. 1-135 MIT, (Aug. 1989).

19. A. E. Eckberg, A. W. Berger, T. Hou, and D. M. Lucantoni, "Performance Characterizations of Traffic Monitoring and Associated Control, mechanisms for Broadband 'Packet' Networks," *Proc. IEEE Globecom*, IEEE, (Dec. 2-5, 1990).

20. A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Proc. SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 1-12 ACM SIGCOMM, (Sept 19-22 1989).

21. L. Zhang, "Virtual Clock : A New Traffic Control Algorithm for Packet Switching," *Proc. SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 19-29 ACM SIGCOMM, (Sept 24-27, 1990).

22. L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet-Switched Networks," *ACM Transactions on Computer Systems* **9**(2) pp. 101-124 ACM, (May 1991).

23. S. J. Golestani, "A Stop-and-Go Framework for Congestion Management," *Proc. SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 8-18 ACM SIGCOMM, (Sept 24-27, 1990).

24. J-C Bolot and A. Shankar, "Dynamical Behavior of Rate Based Flow Control Systems," *Computer Communications Review* **20**(2)ACM SIGCOMM, (Apr. 1990).

25. A. Mukherjee, L. Landweber, and T. Faber, "Dynamic Time Windows and Generalized Virtual Clock: Combined Closed Loop/Open Loop Congestion Control," *Proc. IEEE INFOCOM,* pp. 322-332 IEEE, (May 6-8, 1992).

26. T. Faber, L. Landweber, and A. Mukherjee, "Dynamic Time Windows: Packet Admission Control with Feedback," *Proc. ACM Symposium on Communications Architectures and Protocols,* pp. 124-135 ACM, (August 17-20, 1992).

27. Manolis Katevenis, "Fast Switching and Fair Control of Congested Flow in Broadband Networks," *IEEE Journal on Selected Areas in Communications* **SAC-5**(8) pp. 1315-1326 IEEE, (Oct. 1987).

28. R. Jain, "Congestion Control in Computer Networks: Issues and Trends," *IEEE Network* **4**(3) pp. 24-30 IEEE, (May 1990).

29. Andrew S. Tannenbaum, *Computer Networks, 2nd ed.,* Prentice Hall, Englewood Cliffs, NJ (1988).

30. Kadaba Bharath-Kumar and Jeffrey Jaffe, "A New Approach to Performance-Oriented Flow Control," *IEEE Transactions on Communications* **COM-29**(4) pp. 427-435 IEEE, (April 1981).

31. Jeffrey Jaffe, "Flow Control Power is Nondecentralizable," *IEEE Transactions on Communications* **COM-29**(9) pp. 1301-1306 IEEE, (Sept. 1981).

32. J. Selgar, "New Flow Control Power is Decentralizable and Fair," *Proc. IEEE INFOCOM,* pp. 87-96 IEEE, (Apr. 9-12, 1984).

33. J. M. Selgar, "A Class of Flow Control Algorithms that Maximize the New Flow Control Power," *Proc. IEEE Globecom*, IEEE, (Nov. 26-29, 1984).

34. Jum Matsumoto, "Flow Control in Packet-Switched Networks by Gradual Restrictions of Virtual Calls," *IEEE Transactions on Communications* **COM-29**(4) pp. 466-473 IEEE, (April 1981).

35. Z. Haas and J. Winters, "Congestion Control by Adaptive Admission," *Proc. IEEE INFOCOM*, pp. 560-569 IEEE, (Apr. 7-9, 1991).

36. Z. Haas, "Adaptive Admission Congestion Control," *Computer Communications Review* **21**(5) pp. 58-76 ACM SIGCOMM, (Oct. 1991).

37. J.-C. Bolot and A, Shankar, "Analysis of a Fluid Approximation to Flow Control Dynamics," *Proc. IEEE INFOCOM*, pp. 2398-2406 IEEE, (May 6-8, 1992).

38. Peter Harrison, "An Analytic Model for Flow Control Schemes in Communication Network Nodes," *IEEE Transactions on Communications* **COM-32**(9) pp. 1013-1019 IEEE, (Sept. 1984).

39. Samuel Morgan, "Window Flow Control on a Trunked Byte-Stream Virtual Circuit," *IEEE Transactions on Communications* **36**(7) pp. 816-825 IEEE, (July 1988).

40. Leonard Kleinrock and Parviz Kermani, "Static Window Flow Control in Store-and-Forward Computer Networks," *IEEE Transactions on Communications* **COM-28**(2) pp. 271-278 IEEE, (Feb. 1980).

41. Parviz Kermani and Leonard Kleinrock, "Dynamic Window Flow Control in Store-and-Forward Computer Networks," *IEEE Transactions on Communications* **COM-28**(2) pp. 263-271 IEEE, (Feb. 1980).

42. D. Mitra, "Optimal Design of Windows for High Speed Data Networks," *Proc. IEEE INFOCOM*, pp. 1156-1163 IEEE, (June 5-7, 1990).

43. D. Mitra and T. Seery, "Dynamic Adaptive Windows for High Speed Data Networks: Theory and Simulation," *Proc. SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 30-29 ACM SIGCOMM, (Sept 24-27, 1990).

44. D. Mitra and J. Seery, "Dynamic Adaptive Windows for High Speed Data Networks with Multiple Paths and Propagation Delays," *Proc. IEEE INFOCOM*, pp. 39-48 IEEE, (Apr. 7-9, 1991).

45. J. Fernow and M. El-Sayed, "Stability of Adaptive Controls in Packet Networks," *Proc. IEEE INFOCOM*, pp. 107-114 IEEE, (Apr. 18-21, 1983).

46. J. Fernow and M. El-Sayed, "Stability of Adaptive Congestion Controls in Packet Networks," *Computer Networks and ISDN Systems* **10**(1) pp. 7-18 International Council for Computer Communication, (Aug. 1985).

47. S. Pingali, D. Tipper, and J. Hammond, "The Performance of Adaptive Window Flow Controls in a Dynamic Environment," *Proc. IEEE INFOCOM*, pp. 55-61 IEEE, (June 5-7, 1991).

48. K. Ramakrishnan, "Analysis of a Dynamic Window Congestion Control Protocol in Heterogeneous Environments including Satellite Links," *Proc. IEEE Symposium on Computer Networks*, pp. 94-104 IEEE, (1986).

49. A. Thomasian and P. Bey, "Performance Analysis of Window Flow Control for Multiple Virtual Routes," *Proc. IEEE INFOCOM*, pp. 69-80 IEEE, (Apr. 9-12, 1984).

50. A. Thomasian, Burroughs, and P. Bey, "Analytical Solution of an Integrated Performance Model of a Computer Communication Newtork with Window Flow Control," *Proc. SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 225-233 ACM SIGCOMM, (June 7-8 1984).

51. Wang and Sengupta, "Performance Analysis of a Feedback Congestion Control Policy under Non-Negligible Propagation Delay," *Proc. SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 149-158 ACM SIGCOMM, (Sept. 3-6, 1991).

52. Felix Wong and Jose De Marca, "Fairness in Window Flow Controlled Computer Networks," *IEEE Transactions on Communications* **37** pp. 475-480 IEEE, (May 1989).

53. D. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," *Computer Networks and ISDN Systems* **17**(1) pp. 1-14 International Council for Computer Communication, ().

54. J. Nagle, "Congestion Control in TCP/IP Internetworks," *Computer Communications Review* **14**(4) pp. 11-17 ACM SIGCOMM, (Oct. 1984).

55. D. Clark, S.Shenker, and L. Zhang, "Some Observations on the Dynamics of a Congestion Control Algorithm," *Computer Communications Review* **20**(5) pp. 30-39 ACM SIGCOMM, (Oct. 1990).

56. L. Zhang, S. Shenker, and D. Clark, "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic," *Proc. Symposioum on Communications Architectures and Protocols*, pp. 133-148 ACM SIGCOMM, (Sept. 3-6 1991).

57. J. Mogul, "Observing TCP Dynamics in Real Networks," *Proc. Symposium on Communications Architectures and Protocols*, pp. 305-317 ACM SIGCOMM, (August 17-20, 1992).

58. Z. Wang and J. Crowcroft, "A New Congestion Control Scheme: Slow-Start and Search," *Computer Communications Review* **21**(1) pp. 32-43 ACM SICGOMM, (Jan. 1991).

59. S. Keshav, "A Control-Theoretic Approach to Congestion Control," *Proc. SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 3-16 ACM SIGCOMM, (Sept. 3-6, 1991).

60. S. Keshav, "Congestion Control in Computer Networks," *Ph. D Thesis*, pp. 1-109 University of California at Berkeley, (1991).

61. H. Zhang and S. Keshav, "Comparison of Rate-Based Service Disciplines," *Proc. SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 133-148 ACM SIGCOMM, (Sept. 3-6 1991).

62. S. Morgan, "Queueing Disciplines and Passive Congestion Control in Byte Stream Networks," *Proc. IEEE INFOCOM*, pp. 711-720 IEEE, (Apr. 23-24, 1989).

63. Abhay K. J. Parekh, *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*, MIT (January 24, 1992).

64. Abhay Parekh and Robert Gallager, "A Generalized Processor Sharing Approach to Flow Control In Integrated Services Networks - The Multiple Node Case," *Tech Report 2076*, MIT, (1991).

65. Abhay Parekh and Robert Gallager, "A Generalized Processor Sharing Approach to Flow Control In Integrated Services Networks - The Single Node Case," *Proc. IEEE INFOCOM* **2** pp. 915-924 IEEE, (May 6-8, 1992).

66. J. Darvin and A. Heybey, "A Simulation Study of Fair Queueing and Policy Enforcement," *Computer Communication Review* **20**(5) pp. 23-29 ACM SIGCOMM, (Oct. 1990).

67. S. J. Golestani, "Congestion Free Transmission of Real-Time Traffic in Packet Networks," *Proc. INFOCOM,* pp. 527-536 IEEE, (1990).

68. S. Golestani, "A Framing Strategy for Congetsion Management," *IEEE Journal on Selected Areas in Communications* **9**(7) pp. 1064-1077 IEEE, (Sept. 1991).

69. S. Golestani, "Congestion-Free Communications in High-Speed Packet Networks," *IEEE Transactions on Communications* **39**(12) pp. 1802-1812 IEEE, (Dec. 1991).

70. S. Golestani, "Duration-Limited Statistical Multiplexing of Delay Sensitive Traffic in Packet Networks," *Proc. IEEE INFOCOM,* pp. 323-330 IEEE, (Apr. 7-9, 1991).

71. Louis Pouzin, "Methods, Tools, and Observations on Flow Control in Packet-Switched Data Networks," *IEEE Transactions on Communications* **COM-29**(4) pp. 413-426 IEEE, (April 1981).

72. Alfred Giessler, Annemarie Jagemann, Ellen Maser, and Jurgen Hanle, "Flow Control Based on Buffer Classes," *IEEE Transactions on Communications* **COM-29**(4) pp. 436-443 IEEE, (April 1981).

73. Farouk Kamoun, "A Drop and Throttle Flow Control Policy for Computer Networks," *IEEE Transactions on Communications* **COM-29**(4) pp. 444-452 IEEE, (April 1981).

74. D. Luan and D. Lucantoni, "Throughput Analysis of a Window Based Flow Control Subject to Bandwidth Management," *Proc. IEEE INFOCOM*, pp. 411-417 IEEE, (Mar. 29-31, 1988).

75. A. Eckberg, D. Luan, and D. Lucatoni, "Bandwidth Management: a Congestion Control Strategy for Broadband Packet Networks - Characterizing the Throughput Burstiness Filter," *Computer Networks and ISDN Systems* **20**(1-5) pp. 415-424 International Council for Computer Communication, ().

76. A. Eckberg, D. T. Luan, and D. Lucantoni, "Meeting the Challenge: Congestion and Flow Control Strategies for Broadband Information Transport," *Proc. IEEE Globecom*, IEEE, (1989).

77. G. Ramamurthy and R. Dighe, "Distributed Source Control : A Network Access Control for Integrated Broadband Packet Networks," *Proc. IEEE INFOCOM*, pp. 896-907 IEEE, (June 5-7, 1990).

78. G. Ramamurthy and R. Dighe, "Distributed Source Control : A Network Access Control for Integrated Broadband Packet Networks," *IEEE Journal on Selected Areas in Communications* **9**(7) pp. 990-1001 IEEE, (Sept. 1991).

79. Harry Rudin and Heinrich Mueller, "Dynamic Routing and Flow Control," *IEEE Transactions on Communications* **COM-28**(7) pp. 1030-1039 IEEE, (July 1980).

80. G. Stassinopoulos and Panagiotis Konstantopoulos, "Optimal Congestion Control in Single Destination Networks," *IEEE Transactions on Communications* **Com-33**(8) pp. 792-800 IEEE, (Aug. 1985).

81. Guatam Thaker and J. Bibb Cain, "Interactions Between Routing and Flow Control Algorithms," *IEEE Transactions on Communications* **COM-34**(3) pp. 269-277 IEEE, (March 1986).

82. B. Sanders, "An Asynchronous Distributed Flow Control Algorithm for Rate Allocation in Computer Networks," *IEEE Transactions on Computers* **37**(7) pp. 779-787 IEEE, ().

83. Williamson and Cheriton, "Load-Loss Curves: Support for Rate Based Congestion Control in High Speed Datagram Networks," *Proc. SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 17-30 ACM SIGCOMM, (Sept. 3-6, 1991).

84. Carey L. Williamson, "Optimizing File Transfer Response Time Using the Loss-Load Curve Congestion Control Mechanism," *Proceedings of the Symposium on Communications Architectures, Protocols and Applications* **23, #4** pp. 117-126 ACM SIGCOMM, (September 13-17, 1993).

85. R. Cruz, "A Calculus for Network Delay, Part I, Network Elements in Isolation," *IEEE Transactions on Information Theory* **37**(1) pp. 114-131 IEEE, (Jan. 1991).

86. R. Cruz, "A Calculus for Network Delay, Part II: Network Analysis," *IEEE Transactions on Information Theory* **37**(1) pp. 132-141 IEEE, (Jan. 1991).

87. A. G. Fraser, "Datakit – A Modular Network for Synchronous and Asynchronous Traffic," *Conference Record of IEEE International Conference on Communications*, (June 1979).

88. Mary Vernon and U. manber, "Distributed Round–Robin and First–Come First–Serve Protocols and Their Applications to Multiprocessor Bus Arbitration," *15th IEEE International symposium on Computer Architecture*, (May 30 - June 2,

1988).

89. Alan Berenbaum, Joe Dixon, Anand Iyengar, and Srinivasan Keshav, "A Flexible ATM-Host Interface for XUNET II," *IEEE Network* 7(4) pp. 18-23 (July 1993). Special Issue: End-System Support for High-Speed Networks (Breaking Through the Network I/O Bottleneck)

90. P. DiGenova, "Real-Time Communication on Computer Networks: Introduction and Analysis of the Tenet Approach of Berkeley," *Tesi di Laurea*, University of Bologna, (December 1993).

91. Dominico Ferrari, Anindo Banerjea, and H. Zhang, "Network Support for Multimedia - A Discussion of the Tenet Approach," *Technical Report TR-92-072*, International Computer Science Institute, (October, 1992).

92. David Clark, Scott Shenker, and Lixia Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism," *Prec. ACM Symposium on Communications Architectures and Protocols* **22**(4) pp. 14-26 ACM, (August 17-20, 1992).