

Real-Time Implementation of an Adaptive Traffic Shaper for Dynamic Bandwidth Allocation in ATM Networks

Vinai Rajendran Sirkay
Douglas Niehaus
Joseph B. Evans

TISL Technical Report TISL-9770-28

Prepared for:

Defense Advanced Research Projects Agency/CSTO

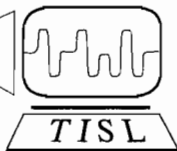
Research on Gigabit Gateways
AARPA Order No. 8634

Issued by EDS/AVS under Contract #F19628-92-C-0080

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. government.

December 1994

Telecommunications and Information Sciences Laboratory
The University of Kansas Center for Research, Inc.
2291 Irving Hill Drive Lawrence, Kansas 66045



Abstract

Traffic contracts between the customers and network providers encourage customers to produce traffic conforming to the negotiated parameters. In order to provide conforming traffic a customer may *shape* the traffic between the source and the network. Traffic shaping refers to the queueing of ATM cells and releasing cells so that they meet the specifications of the contract negotiated.

In this work, a real-time hardware-software implementation of an adaptive shaper is presented. The overall goal of such a shaper is to provide an algorithm that will change the rate assigned to a VC according to source characteristics obtained via measurements in hardware. The algorithm used to provide this control is the *i*-out-of-*m* adaptive algorithm which allows a VC to transmit *i* back to back cell in an *m* slot interval. Since the customer will not be able to accurately characterize the traffic it is necessary for the adaptive shaper to dynamically change the *i* and *m* parameters to follow the trends of the customer traffic. The adaptive shaper task is a part of the system software for the 622 Mb/s ATM/SONET Gateway Card.

Contents

1	Introduction	1
1.1	Organization of the report	4
2	Background	6
2.1	B-ISDN Protocol Reference Model	6
2.2	Synchronous Optical NETwork (SONET)	7
2.3	Asynchronous Transfer Mode (ATM)	10
2.4	The AN2 ATM Switch	12
2.5	The ATM/SONET Gateway Card	14
2.5.1	Modes of Operation of Gateway Card	15
2.5.2	Transmit Section	15
2.5.3	Receive Section	17
2.5.4	ATM Support Hardware	19
2.5.5	The Line Card Processor	22
2.6	DECelx : Real Time Kernel	23
3	Switch Software Architecture	25
3.1	Boot Time Processing	25
3.2	ATM/SONET Gateway Boot Process	28
3.2.1	Control Register	28
3.2.2	Transmit SONET Overhead SRAM	30

3.2.3	FIFO Controller	30
3.2.4	i out of m	31
3.2.5	Batch Credit Processor	32
3.2.6	Descrambler	32
3.2.7	Traffic Measurements	33
3.2.8	Batch Credit Generation	33
3.3	ATM/SONET Gateway Run Time Processes	34
3.3.1	Interrupt Handling and Processing	34
3.3.2	Port Monitor	37
3.3.3	Management and Configuration Server	38
4	Dynamic Bandwidth Allocation	42
4.1	Motivation	42
4.2	The i-out-of-m Adaptive Shaping Algorithm	45
4.2.1	Measurement Parameters	45
4.2.2	Control Mechanism	47
4.3	Simulation and Benchmarking Results	50
4.4	Hardware-Software Implementation	65
4.4.1	i-out-of-m Controller	65
4.4.2	Pacing Measurement System	71
4.4.3	Hardware - LCP Interaction	73
4.5	Limitation of Implementation	80
5	Conclusion	83
5.1	Summary and Conclusions	83
5.2	Future Work	85
A	SONET Overhead	92
B	GUI	97

List of Tables

4.1	Timing and Performance of Adaptive Shaper	52
4.2	Delay Comparison for Adaptive and Non-Adaptive i-out-of-m Shapers	54

List of Figures

2-1	B-ISDN Protocol Reference Model	6
2-2	SONET STS-1 Frame Format	9
2-3	STS-3c Frame Structure	10
2-4	UNI-based ATM Cell Format	11
2-5	ATM Cells in a SONET Frame	12
2-6	The AN2 ATM Switch Architecture	13
2-7	Block Diagram of the Transmit Section of the Gateway Card. . .	16
2-8	Block Diagram of the Receive Section of the Gateway Card. . . .	18
2-9	Block Diagram of ATM Support Hardware on the Gateway Card.	20
3-1	ATM/SONET Gateway Card Boot Processing	29
3-2	Run Time Processing	35
3-3	Server - Management Station Communication	39
4-1	Dynamic Bandwidth Allocation System at UNI	45
4-2	Measurement Points on the ATM/SONET Gateway Card	46
4-3	Adaptive i-out-of-m shaping algorithm	49
4-4	Rate Adaptation of Shaper for 1,000 cell slot measurement interval	54
4-5	Rate Adaptation of Shaper for 5,000 cell slot measurement interval	55
4-6	Rate Adaptation of Shaper for 12,500 cell slot measurement interval	55
4-7	Rate Adaptation of Shaper for 20,000 cell slot measurement interval	56
4-8	Rate Adaptation of Shaper for 25,000 cell slot measurement interval	56

4-9	Rate Adaptation of Shaper for 50,000 cell slot measurement interval	57
4-10	Rate Adaptation of Shaper for 100,000 cell slot measurement interval	57
4-11	Performance of Algorithm with respect to Rate Adaptation Constant	59
4-12	Trace data showing amount simulated	60
4-13	Rate Adaptation of Shaper to Ethernet trace data	62
4-14	Static Rate Assignment on Ethernet trace data	63
4-15	System Block Diagram	66
4-16	i-out-of-m Block Diagram	67
4-17	Schedule Pointers Table	68
4-18	Schedule List	69
4-19	Example of Schedule of Credit Return	70
4-20	Example of Actual Hardware Credit Return	70
4-21	Scenario 1: System Gaining Credits	75
4-22	Credit Divergence - VC gaining credits	76
4-23	Scenario 2: System Loses Credits	77
4-24	Algorithm used to modify the i and m values in pacing table . . .	79
4-25	Time Usage of the i-out-of-m controller	81
A-1	SONET OC-3 Frame With Values	93
A-2	SONET OC-12 Frame With Values	94
B-1	SRAM Debug GUI	97
B-2	SONET Overhead GUI	98
B-3	PVC Setup GUI	99
B-4	Gateway Configuration GUI	100

Chapter 1

Introduction

Telecommunications and computer networking have attained vital importance in many areas including education, business, defense, space, and industry. The current generation of telecommunication networks have had a profound effect on many walks of life. They have decentralized decision making in large organizations and have improved industrial productivity.

An increasing number of applications utilize sets of powerful computing resources distributed over two main kinds of networks.

- Local Area Networks (LAN) : which interconnect computers in close proximity to each other.
- Wide Area Networks (WAN): which interconnect computers across wide geographical separation of distances.

In such a distributed environment, the full potential of computing resources can be realized only if the network speeds are sufficient to support the processing demands. Currently available WANs operate at megabits-per-second speed and therefore usually limit the performance of these distributed systems. Gigabit-per-second WANs promise a major advance in combining computing and communications thereby helping to produce powerful, geographically distributed com-

puting resources with high-speed access to remote and time-critical data sources. Such networks will allow researchers to develop distributed, interactive applications with massive data throughputs and real time requirements. Furthermore, they will allow data from multiple sources to be integrated by these applications, where neither the data sources nor the users will be co-located with the computing resources.

Many challenges must be addressed before the benefits of gigabit WANs can be achieved. Some of these include:

- Interoperability of heterogeneous computing and networking devices
- Coordination of multiple data streams destined for a single location
- Methods of accommodating bursty as well as steady traffic
- The ability to compensate for the effects of network delays and errors on data transmission

Considering the importance of distributed environments, the High Performance Computing and Communications (HPCC) initiative has as one of its key elements a high-speed communications network that spans the United States. This initiative will lay the foundation for changes in telecommunication at all levels, thus yielding significant benefits to research and education communities. Research on very high speed communications networks is expected to increase dramatically during this decade in countries moving towards prominent positions in tomorrow's telecommunications and computer industries.

The architecture of this infrastructure has been undergoing drastic changes over the last few years. The trend has been to move away from dedicated transmission facilities toward reconfigurable high-bandwidth virtual channel facilities. These changes have been brought about by the introduction of Broadband Integrated Services Digital Networks (B-ISDN) [21]. B-ISDN networks enable a

large variety of high bandwidth requirements and delay sensitive services, such as video-teleconferencing, distributed and networked computing and multimedia communication systems, that cannot be supported by existing networks. The emerging demand for these services and the desire to integrate them under a single interface have lead to a focused effort to define the standards for B-ISDN. The international body for telecommunications standards (CCITT - now known as ITU-T) has recommended a standardized multiplexing and switching technology called Asynchronous Transfer Mode (ATM) as the basis for B-ISDN [21]. ATM uses small fixed size cells, allowing efficient statistical multiplexing of continuous and bursty traffic, yet ATM is also flexible enough to provide bandwidth on demand. The potential of ATM based networks has been the focus of several R&D testbeds in the United States and other countries; favorable results from these testbeds have prompted most of the major public carriers in United States and other developed nations of the world to move toward ATM cell based networks [21, 31]

Fiber optic networks using Synchronous Optical NETWORK (SONET) protocols as the standard for transmission of digital information over an optical fiber have emerged as the transport technology for B-ISDN, since they can provide the bandwidth required by the applications of the future. This standard defines a new hierarchy, that is, a new set of transmission rates and formats that capitalize on the capabilities of optical fibers. These new formats are far more robust than existing transmission protocols. The potential for new services supported by this technology has lead exchange carriers and computer network providers to deploy and test these standards in both local and wide area networks.

Interfacing LANs with WANs will play a key role in the realization of high performance distributed computing. Gateways are devices that provide the interface between different networks. The work presented in this thesis is a portion of the system software for the DEC AN2 ATM LAN switch supporting a 622 Mb/s

gateway card which interfaces an ATM based LAN with an experimental ATM based WAN backbone. The AN2 developed by Digital Equipment Corporation's (DEC), Systems Research Center is a part of the research being conducted on the Multidimensional Applications Gigabit Internetwork Consortium (MAGIC) project. The MAGIC project is one of six gigabit wide area testbeds created for research in high speed networks that addresses the challenges of internetworking and demonstrates real-time, interactive exchanges of data at gigabit-per-second rates among multiple distributed servers and clients.

1.1 Organization of the report


This work presents the design and implementation issues associated with the system software for the gateway card created for the DEC AN2 ATM Switch, with emphasis on the hardware-software implementation of an adaptive traffic shaper developed to perform dynamic bandwidth management.

Chapter 2 provides background on the B-ISDN protocol reference model, SONET, ATM, and describes the DEC AN2 ATM switch and the 622 Mb/s gateway card developed at the University of Kansas.

Chapter 3 describes the design and architecture of the system software and the associated graphical user interfaces (GUIs). It describes the software needed to boot the gateway card as a part of the AN2 startup procedure. This chapter also details the system software that is needed during the normal operation of the gateway card in the AN2 environment.

Chapter 4 describes the simulation and performance results of an adaptive shaper that can be used for dynamic bandwidth management. This chapter also provides a hardware/software implementation of such a shaper as part of the system software described in the previous chapters.

Finally, Chapter 5 presents conclusions of this work and proposes future work



and extensions.

Chapter 2

Background

2.1 B-ISDN Protocol Reference Model

The protocol architecture for B-ISDN introduces some elements not found in integrated services digital network (ISDN) protocols. B-ISDN, which began as an evolution from circuit-switched ISDN networks, is being transformed into a packet-switched network as it encompasses high bandwidth multimedia and other broadband services [29, 22]. The B-ISDN protocol reference model shown in Figure 2-1 assumes that the host to host transfer of information occurs via fixed length packets called cells over a slotted transmission medium [29, 11].

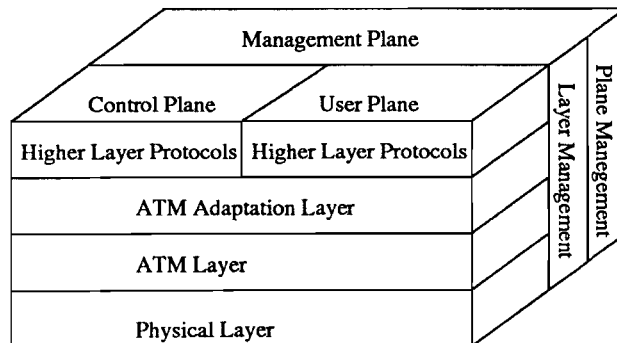


Figure 2-1: B-ISDN Protocol Reference Model

The model uses the concept of separate planes for user, control, and manage-

ment functions [29]. This provides for the segregation of functions among layers as recommended by the Open Systems Interconnection (OSI) model [29].

The protocol model describes three separate planes :

- User Plane : provides user information transfer, along with associated controls.
- Control Plane : performs signaling and call-management functions.
- Management Plane : performs management functions related to the system as a whole.

The physical layer of this model is based on SONET which is a slotted transmission medium. Detailed description of the standard is provided in Section 2.2. The ATM layer transports the ATM cells and is common to all services. The ATM layer performs multiplexing and demultiplexing of cells associated with different connections, routing translation, passing of cell information to and from the ATM Adaptation Layer (AAL), and flow control and policing functions. A detailed description of ATM can be found in Section 2.3.

2.2 Synchronous Optical NETWORK (SONET)

The most widely accepted standard for fiber optic transmission in North America is SONET. This standard has been agreed to internationally by the CCITT (also known as the ITU-T), such that transmission equipment from one manufacturer can communicate with receiving equipment from another manufacturer [3, 29]. Outside of North America, a largely compatible system the Synchronous Digital Hierarchy (SDH) is used.

The highest traditional circuit-switched rate, digital synchronous hierarchy signal, DS3, has approximately the same data rates as the basic building block

of the SONET format, a Synchronous Transport Signal (STS-1) [30]. An STS-1 frame contains two parts:

- *SONET Overhead*;
- *SONET Payload*.

Figure 2-2 shows an STS-1 SONET frame as a set of bytes arranged in rows. Transmission of this frame is done row by row, from left to right. Thus, the overhead is distributed throughout the frame rather than coming at the start as in the DS formats. Such distribution of overhead information across a frame increases the survivability and fault tolerance of a medium to bit errors. The SONET overhead is further broken into three parts:

- *Section* : individual fiber transmission links between repeaters and links between repeaters and other equipment , such as multiplexers;
- *Line* : fiber transmission lines connecting SONET equipment at a given rate in the SONET hierarchy, STS-n. A line may consist of several sections;
- *Path* : the path over which traffic is sent end-to-end between originating and terminating equipment using the SONET format. A path may traverse several lines.

The entire STS-1 frame consists of 810 bytes and repeats at 125 μ s intervals thus yielding a nominal bit rate of 51.84 Mb/s. The Synchronous Payload Envelope (SPE) is the area reserved for transporting the data - also called the *payload* - and contains 783 bytes per frame. Nine bytes in the SPE are overhead information that the system uses to verify the integrity of the payload and to supply end-to-end signaling. These nine bytes of overhead are referred to as *Path overhead* and they stay with the payload until it reaches the final destination. The remainder of the STS-1 frame consists of twenty-seven bytes, used for inter-network signaling

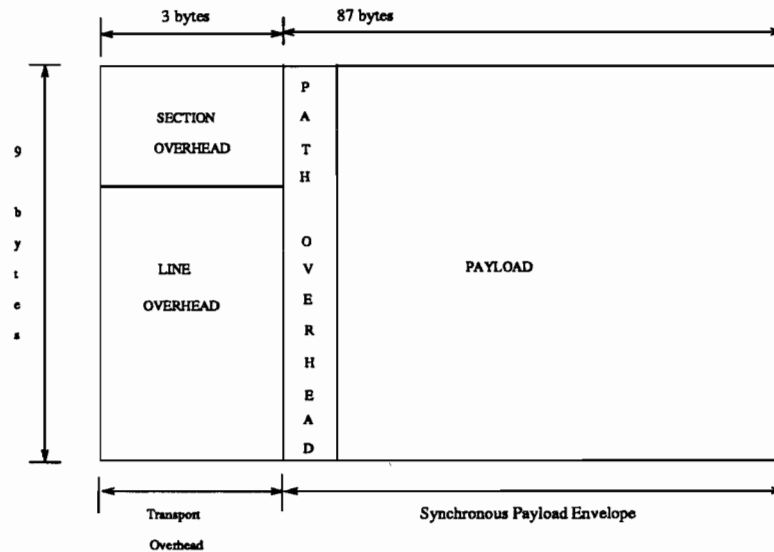


Figure 2-2: SONET STS-1 Frame Format

and error checking. Nine bytes of the twenty-seven are used for the section portion of the network, that is, between optical repeaters. The remaining eighteen bytes are used for the line portion of the network, that is, between terminals. The SONET format use a pointer to do frequency and phase aligning of the payload. The SONET pointer contains the location of the beginning of the data payload which allows forward or backward movement of data within the frame [30].

STS-1 can be multiplexed into higher rate signals denoted by the terminology STS-N, where N denotes the number of STS-1s that have been byte multiplexed together. The optical signal equivalent to STS-N is given the designation Optical Carrier (OC)-N. Higher rate signals can be achieved by concatenation of individual STS-1 frames into a single STS-Nc frame. A SONET STS-3c frame is made of 3 concatenated STS-1 frames and is carried as a single stream. Figure 2-3 shows a SONET STS-3c frame [29]. It is a 2430 byte pattern that repeats every $125\mu\text{s}$ thus yielding a nominal bit rate of 155.52 Mb/s. Its two main components are the 81 byte Transport Overhead (TOH) and the 2349-byte Synchronous Payload Envelope (SPE). STS multiplexing is a process in which the transport overhead of the multiplexed signals are aligned by the payload pointer, and in which the

signals are sequentially byte-interleaved. This results in the appearance of the byte triplets in a STS-3c frame. Some of the overhead bytes in an STS-3c, such as the path overhead byte, are defined only for the first STS-1 position in a concatenated structure [30].

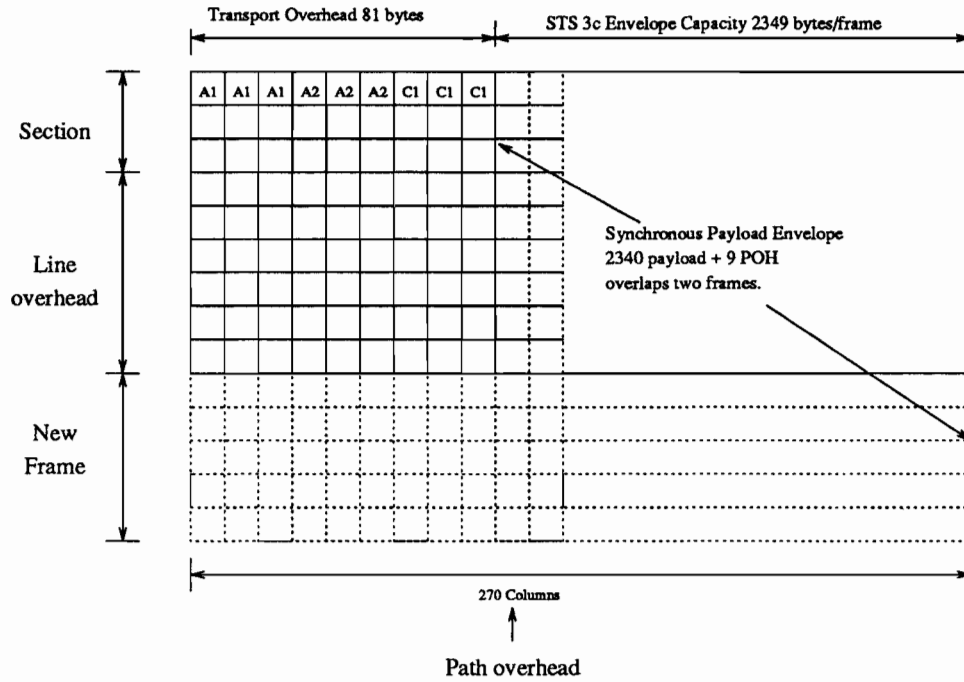


Figure 2-3: STS-3c Frame Structure

An STS-12c stream is at twelve times the basic SONET rate, that is 622 Mb/s. 12 STS-1 or 4 STS-3c streams can be multiplexed together to form a single STS-12c stream. The STS-12c frame is a scaled version of an STS-3c frame [3, 30].

2.3 Asynchronous Transfer Mode (ATM)

As the standards for B-ISDN began to evolve in the mid-1980's, it was assumed by most participants that some form of time-division multiplexing (TDM) would be used. However, it was found that the ISDN based TDM techniques failed to support the high rates needed to provide B-ISDN services [29]. Due to the non scalability of TDM techniques, an asynchronous, statistically multiplexed,

packet based scheme called asynchronous transfer mode (ATM) became the target technology for the implementation of the services offered by B-ISDN [22, 20].

ATM is a packet-oriented transfer method. Like frame relay and X.25 technologies, it allows multiple logical connections to be multiplexed over a single physical channel. The information flow on each ATM logical connection is organized into fixed-size packets called *cells* [22, 32]. As shown in Figure 2-4, ATM cells are 53 byte data units consisting of two parts:

- *Header* (5 bytes)
- *Body* (48 bytes)

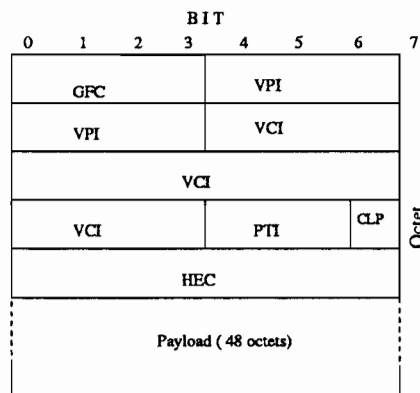


Figure 2-4: UNI-based ATM Cell Format

The header contains a *Virtual Circuit Identifier* (VCI) and a *Virtual Path Identifier* (VPI) which are used for routing by switches in the network. The PTI field contains the *Payload Type Indicator* used by AALs to indicate to the network the kind of traffic being sent. The CLP field indicates the *Cell Loss Priority* used by the network to decide which cells can be dropped when congestion occurs. A cell with a lower priority is the first candidate for dropping. The HEC field contains the *Header Error Code* which is used by the network equipment to determine if any bits in the header have been corrupted during transmission. Corruption of header bits may lead to delivery of cells to an incorrect end user.

The header also has a 4 bit GFC field that will be used for *Generic Flow Control*. The standards for use of these bits are presently being developed [12].

ATM cells are transported from end to end via SONET frames. The transmitting end assembles ATM cells into a the payload portion of a frame. The receiving end of the SONET frame extracts cell from the payload and passes them up the protocol stack. Figure 2-5 shows the transport of ATM cells in a SONET frame.

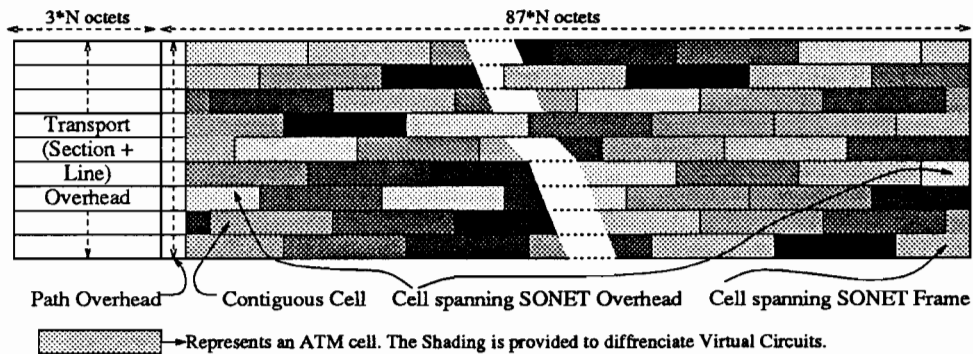


Figure 2-5: ATM Cells in a SONET Frame

2.4 The AN2 ATM Switch

The AN2 is an experimental LAN from Digital Equipment Corporation (DEC). Hosts are connected to switches across the User-Network Interface (UNI) by one or more 155 Mb/s host adapters (OTTO), and switches are interconnected across the Network-Network Interface (NII) by either 155 Mb/s or 622 Mb/s links. The switching fabric is crossbar based as shown in Figure 2-6.

The backplane operates at 12.8 Gb/s, while each of the 16 ports has a bandwidth of 800 Mb/s. With a non-blocking switching fabric, the switch itself will not drop a non-corrupted cell. However, multiple cells might be destined for the same port during the same cell cycle thus requiring buffering at the switch [2]. There are two different types of cards that plug into the switches crossbar :

1. the Quad line cards operating at 155 Mb/s, and

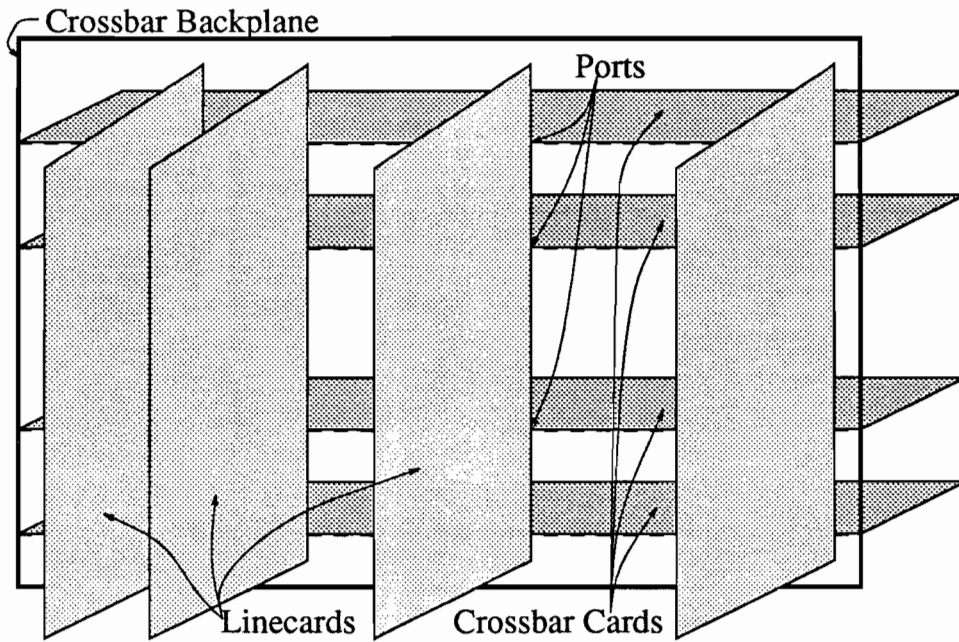


Figure 2-6: The AN2 ATM Switch Architecture

2. the ATM/SONET Gateway operating at 622 Mb/s.

The input side of the line card receives cells over the link, routes those cells, queues them, and sends them through the crossbar. The output side receives cells from the crossbar, buffers them, transmits the cells on the link and manages the flow control mechanism. In this switch, queues will never overflow because of the flow control mechanism which schedules their arrivals and departures [2].

Cells are synchronously switched through the crossbar with a slot period of 520 ns (13 clock cycles). During arbitration, lasting one slot period, each line card requests access to each output line card for which it has traffic. The arbitration mechanism will result in either no connection or a single connection to an output line card. Cells from the crossbar are immediately transmitted on the output link [2].

The AN2 implements a strict, window flow control mechanism on a link by link and virtual circuit-by-virtual circuit basis. During call setup, buffers are allocated on the input side of each line card along the route for the virtual circuit. A line card

will not transmit a cell on a link unless it is sure there is a buffer at the receiving end to store that cell. The transmit line card maintains an account balance of the number of buffers available at the receiver for each virtual circuit, which is decremented as each cell is transmitted. When the account balance reaches zero, the output notifies the input line card, through the crossbar, to stop the virtual circuit. The output side of a line card piggybacks the virtual circuit identifier of a cell forwarded through the crossbar on the cell stream going back to the far end. This serves as an acknowledgement that will be used by the output side of the far end line card to increment its account balance for that virtual circuit [19].

Each line card has a RISC based processor, called the Line Card Processor (LCP), to monitor, control, and manage the line card. The LCP is involved in call setup, call tear down, route finding, resource allocation, periodic bandwidth allocation, line card monitoring and performance measurement.

2.5 The ATM/SONET Gateway Card

The ATM/SONET Gateway (GW) is a hardware device that connects the DEC AN2 Local ATM network to the standards based ATM over SONET network [19]. The purpose of the gateway is to provide a means for data to move between the AN2 and the SONET based wide area network. The gateway card is unique since it has two distinct modes of operation - the SONET STS-12c and quadruple SONET STS-3c. At startup, one of the modes must be chosen.

The GW is a single card that plugs into an AN2 switch port. The GW supports the following features:

- operation at the SONET STS-12/STS-12c (622.08 Mb/s) data rates on the wide area side via fiber optic connection;
- operation within the DEC AN2 Local ATM switch by connecting to the AN2 switch backplane;

- experimental techniques for dynamic bandwidth allocation;
- experimental techniques for interoperability between connection-oriented and connection-less protocols;
- experimental signaling protocols for call setup and call parameter negotiations;
- measurement of network performance.

2.5.1 Modes of Operation of Gateway Card

The ATM/SONET Gateway card operates in two different modes. In the first mode of operation, a single stream of cells are packed into a SONET STS-12c frame with a bandwidth of 622 Mb/s. In the second mode, we have four streams of ATM cells packed into SONET STS-3c frames each with bandwidth of 155 Mb/s. Both these types of frames can be switched at the SONET layer by the standards based switching equipment. The gateway card uses Xilinx XC-3195 field programmable gate arrays (FPGA) to allow for both modes of operation on the same board without duplication of the hardware.

2.5.2 Transmit Section

The transmit section connects the AN2 LAN switch to the transmit SONET Network Termination Equipment (NTE) of the WAN provider. A block diagram of this section is given in Figure 2-7.

As cells come off the AN2, 4 FIFOs are used to perform rate adjustment needed to convert from the 800 Mb/s (40ns clock cycle) AN2 LAN to the 622 Mb/s (51.2ns clock cycle) SONET WAN. Four FIFO buffers are used, one for each stream in the quad STS-3c mode which are serviced by a round robin mechanism on a per FIFO basis. In the STS-12c mode, the 4 FIFOs are combined to form a single

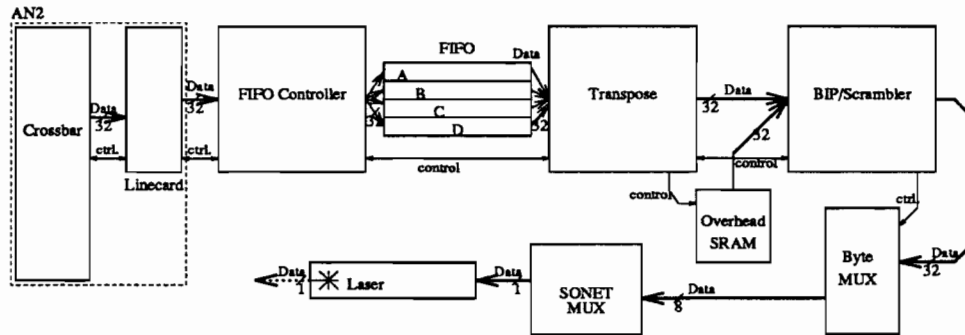


Figure 2-7: Block Diagram of the Transmit Section of the Gateway Card.

large FIFO. The writing of the FIFOs is controlled by a hardware device called the *FIFO Controller* which is the first Xilinx in the transmit path [33]. The *FIFO Controller* resets the FIFOs, initializes them under the command of the Line Card Processor, and then proceeds to fill the FIFOs with cells received from the AN2. It also contains a free running counter that is used to generate signals to synchronize the overhead and payload sections of the SONET frame. In general, these signals provide an indication of the start of the frame, the start of the row within a frame and the location of overhead within the SONET frame. After the FIFOs, the cells are sent to the *Transpose* the second chip in the transmit path. The *Transpose* performs the byte multiplexing of the four STS-3c streams into an STS-12c rate [26]. This operation is carried out by reading a single word (32 bit each) from each of the FIFOs and then performing a transpose operation on them to achieve a byte multiplexed stream. The *Transpose* also performs ATM level cell synchronous scrambling, computation and insertion of the header error correction byte and the insertion of the Section, Line and Path overhead into the SONET frame. Most of the SONET Section, Line and Path overhead is stored in dual ported SRAMs. This enables the LCP to write in the fields that do not require high speed processing. In order to simplify this entire process, it was chosen to start the SONET Synchronous Payload Envelope (SPE) right after the last C1 byte in the first row of the SONET frame. The LCP can write the transmit side

overhead to SRAMs and never change them. The fields in the overhead SRAM are sorted and arranged in order to simplify their insertion into the data path. With 12 bits of address, the least significant four bits represent the column within the SONET overhead, the middle four bits represented the row of the frame and the high four bits represented the frame number. This means that most of the overhead for 16 frames can be stored in the SRAM, with only a few fields requiring updates at relatively infrequent intervals [27]. The BIP/Scrambler performs the most intensive computation on the SONET frame structure [23]. This is the Bit Interleaved Parity (BIP) for the Section, Line and Path regions of a frame. Each of these fields is a parity on all the bits of that particular region taken in either a byte wide or integer number of bytes wide chunks. The Section and Path BIP are 1 byte per SONET frame, hence they are 1 byte in the STS-12c mode while they are 4 bytes wide in the quad STS-3c mode. The Line BIP is a total of 96 bits wide in both cases. The second and most important functionality of the BIP/Scrambler is the application of the SONET PN sequence to scramble the bits before they are transmitted over the fiber optic medium. Finally, the BIP Scrambler controls an external counter running at four times the word clock (51.2 ns). This counter runs off the SONET byte clock (12.86 ns) and controls a bank of multiplexers that convert the data path from 32 bits to 8 bits. The 8 bit data is fed into Vitesse VS8011LC 8 to 1 SONET Mux, then through an AT&T 1227 laser, and then into the fiber optic link.

2.5.3 Receive Section

The receive section, shown in Figure 2-8, connects to the received signal from the wide area SONET NTE.

On the receive side, the Sumitomo ES 9306-RD optical to electrical converter is used to convert the optical signal to an electrical signal. This signal is then used as input to an AT&T 600A 622 Mhz surface acoustic wave (SAW) filter that

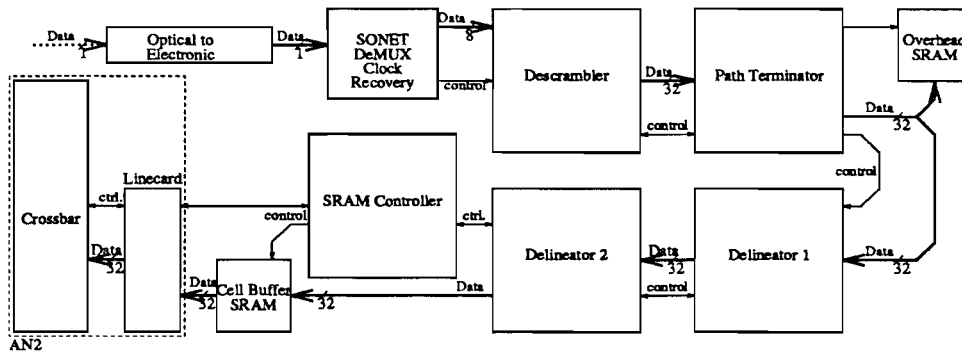


Figure 2-8: Block Diagram of the Receive Section of the Gateway Card.

recovers the bit clock (1.608 ns). A Vitesse VS 8012 SONET 1 to 8 SONET DeMUX derives the byte clock (12.86 ns). From this, the word clock is derived using a shift register with a preloaded pattern. The SONET byte stream is then passed into the *Descrambler* Xilinx [24]. This chip first converts the byte stream to a 32 bit word stream using the generated clock phases to trigger various quadrants of a register for byte to word conversion. Next, the stream is descrambled by applying the SONET PN sequence and the Section BIP is extracted. The BIP is also computed by a given algorithm and compared to the extracted value [30]. The Descrambler also provides an interface to the LCP for it to monitor the status of SONET framing information and other hardware interrupts that may be generated. The Descrambler generates the SONET synchronizing signals as well as the most significant 8 bits (out of 12) of address required for the receive SONET overhead SRAMs. The second Xilinx in the receive data path is the *Path Terminator* [25]. This component determines the location of the start of the Synchronous Payload Envelope (SPE) and the Path overhead in the incoming SONET frame. It also provides the 4 least significant bits of address for the receive overhead SRAMs and the required control signals to store all of the overhead including Path overhead. The third and fourth Xilinxes in the receive data path are the ATM delineation and partial cell buffer. The first component delineates the ATM cells out of the received stream. This is done by continuously checking

for the HEC byte in the header and verifying if it matches the CRC computed on the previous 4 HEC bytes. The partial cell buffer is required because complete cells cannot be buffered within the Xilinx. Partial cells, in the form of 32 bit words, are buffered before being written to a buffer SRAM where cells are reconstructed. The SRAM Controller Xilinx controls the cell reconstruction within the reconstruction SRAM. Partial cells from multiple streams can be multiplexed into the SRAM where complete cells are created. These cells are then forwarded to the AN2 for switching.

2.5.4 ATM Support Hardware

The ATM/SONET Gateway Card provides hardware support for several types of advanced control and monitoring features using the AN2 *qabus* as the primary interface. Some of the support hardware includes a real-time traffic measurement component, rate and burst based traffic measurement component, a credit based flow control implementation and a rate based flow control implementation. Figure 2-9 shows the ATM support hardware on the Gateway Card.

Traffic entering and leaving the ATM/SONET gateway card can be measured via the Traffic Measurement Xilinx. This Xilinx monitors the output and input ports of the AN2 line card interface to the ATM/SONET gateway. The VCI part of the ATM header of the cells that are transmitted and received on a line card are sampled at every cell slot and packaged into ATM cells to be recorded on a remote host. Up to three VCs that carry line traffic information can be configured. The first VC carries the header information of the cells on the output side of the line card. The second VC carries the header information of the cells going into the cell buffer VRAM on the common board and the third VC carries the headers of the cells emanating from the VRAM as they are being forwarded through the crossbar. Though the design allows for three VC's to be used, only two VCs are implemented due to spatial restrictions in the Xilinx.

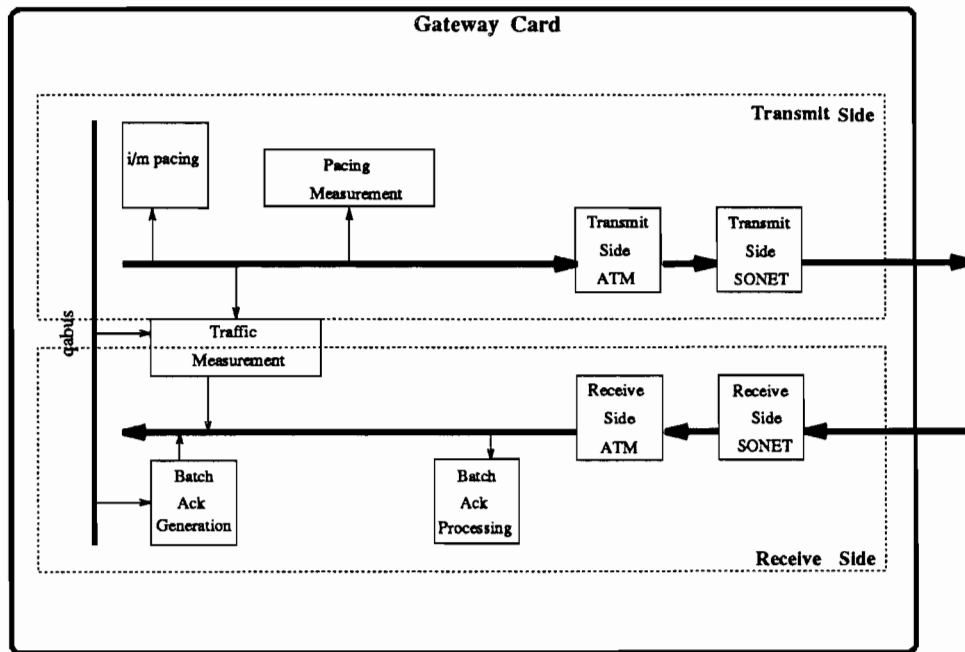


Figure 2-9: Block Diagram of ATM Support Hardware on the Gateway Card.

A slotted rate based flow control mechanism is implemented on the gateway card via the i-out-of-m Controller Xilinx. The i-out-of-m sliding-window algorithm provides a mechanism to control the average throughput and burst characteristics of a virtual circuit [12]. The control for this algorithm is based on the usage and return of credits which are required for a VC to be able to transmit. If a VC runs out of credits, its traffic is queued and it will not be allowed to transmit until a credit is returned. This algorithm can transmit a maximum of i ATM cells in any m cell slot interval, yielding a maximum sustained throughput of i/m normalized to the line rate. The value of i also indicates the maximum number of consecutive cells that can be transmitted. This algorithm is suitable for traffic that is transported in a slotted ATM over SONET environment. This algorithm is implemented on a per VC basis and any number of the 4096 VCs supported on the gateway can be simultaneously flow controlled independent of each other. Different VCs may have different values of i and m to provide more flexibility in control. The algorithm implemented is one of the strictest available rate based

control mechanisms.

Certain traffic streams can be rate and burst measured via the Pacing Measurement Xilinx. This hardware unit is also used to monitor the VCs that are being rate controlled by the *i-out-of-m controller*. The Pacing Measurement Xilinx maintains three fields on a per VC basis. One of the fields is the *rate measurement* that provides the number of cells transmitted on that VC. In addition, a single bit field called the *stopped flag* is used to indicate if the VC is currently stopped. This bit remains set until a cell is transmitted on that VC. The third and last field is the *burst count* which indicates the number of bursts in which the VC was active. These three fields can be read and cleared by the LCP.

Details of the i-out-of-m algorithm and an implementation of an adaptive shaper using the i-out-of-m controller and Pacing Measurement can be found in Chapter 4. This adaptive shaper can be used along with an adaptive network policer to perform dynamic bandwidth allocation across the UNI.

The other two Xilinx on the gateway card are used to implement a credit based flow control scheme which is an extension of the AN2 flow control mechanism. It is tailored for the WAN environment with ATM cells. One of the Xilinx implements the functionality for the upstream node while the other Xilinx implements the functionality for the downstream node. The downstream Xilinx monitors the buffer in the linecard. When cells are forwarded out of the cell buffer, thus freeing up space, credit entries are created and stored. Credit entries are needed because the ATM header cannot be used for cell credits in the WAN. These credit entries are then packaged into ATM cells and injected into the traffic stream going towards the upstream node. Each credit entry holds the VCI and a credit value. The WAN is also capable of remapping the VCI of the cells for purposes of switching. The VCI is a reverse mapping of the received VCI to the originating VCI on the upstream node. The credit value can credit from 1 to 16 buffer spaces. A maximum of 24 credits can be packaged into a single ATM cell. This method

allows for tunneling through a network that does not implement credit based flow control. Currently, the credit cell does not carry any error detection or error protection codes. Up to four VCs can be used to carry credits to the upstream nodes. These VCs may be switched in the network by non-flow controlled switches and routed to their destinations. The upstream Batch-Acks chip monitors the arriving cells as they are stored in the VRAM. When a VC is detected as one that carries the credits from any of the four downstream nodes, the payload of this cell is buffered. The credit entries are extracted from the cell and stored in a FIFO to prevent loss of credits if multiple cells arrive within a short period of time. The credit entry, one per cell cycle, is passed to the linecard by an interface that accepts the VCI and credit value. The common board updates the status of the VCIs under flow control.

The Gateway Card is populated with discrete components such as registers, memories, programmable array logic devices (for address decoding etc.), FIFOs, and buffers. These are used where small, quick storage or logic computation is required. The high speed section of the gateway consists of multiplexers, Transistor Transistor Logic (TTL) to Emitter Coupled Logic (ECL) and ECL to TTL level converters, terminating resistors and capacitors.

2.5.5 The Line Card Processor

Each line card in the DEC AN2 switch has a Reduced Instruction Set Computer (RISC) Architecture based processor, called the Line Card Processor (LCP). The LCP is a LSI Logic LR 33000-25 which is a 25 MHz high-performance integrated 32-bit Processor. The LR33000 contains an 8-Kbyte instruction cache and a 1Kbyte data cache. The processor can access both caches during a single clock cycle, which allows the processor to execute one instruction per cycle when executing instructions out of cache memory. Both caches employ direct address mapping and support snooping to maintain cache coherency. The data cache uses

a write-through technique to maintain coherency with main memory [18].

The LCP has the following memory and I/O devices:

- 8 Mbytes of DRAM. This is implemented with 8 1M * 8 DRAM chips. These chips are controlled by the LR33000 internal DRAM controller.
- 2 Mbytes of flash EEPROM. This is implemented with 4 512K * 8 AM29F040 chips.
- A 512K * 8 UV EPROM.

The LCP manages the resources of the receive section, transmit section, and communications paths using an embedded preemptive priority multitasking real-time operating system called DECelx [19]. The LCP is responsible for setting up circuits, releasing circuits, monitoring circuits, allocating bandwidth to circuits, and other network management operations both within the AN2 and with the WAN.

2.6 DECelx : Real Time Kernel

DECelx Real-time tools for Ultrix is a software product developed by Digital Equipment Corporation (DEC) for developing dedicated realtime and distributed applications that run on a supported set of LR33000 based processor platforms. The DECelx software includes a high performance runtime executive, powerful testing and debugging facilities, and an ULTRIX cross development package [8]. DECelx provides networking facilities that are UNIX-compatible.

Features of a DECelx real-time system are :

- High-performance real-time kernel facilities;
- POSIX synchronization facilities;
- Network facilities;

- Module loader and system symbol table;
- DECElx shell;
- Debugging facilities;
- I/O system;
- Local file system;
- Remote file system;
- Performance evaluation tools;
- Utility libraries;
- Board support packages;
- Boot ROMs;
- System configuration utilities.

Chapter 3

Switch Software Architecture

The software for the DEC AN2 switch is divide into two distinct parts. The first part is associated with booting the switch while the second part is associated with the normal operation of the switch such as setting up and managing the virtual circuits. Section 3.1 deals with the the procedures required to boot the AN2 switch. Section 3.2 deals with the the procedures required to boot the ATM/SONET gateway cards in the switch. Section 3.3 deals with the procedure and processing required for the ATM/SONET gateway cards during the normal operation of the switch.

3.1 Boot Time Processing

The boot time software is executed when the switch is turned on or restarted. This software involves bringing the switch into a know and stable state before the switch can operate in a normal mode. The first section of boot software deals with the processing and set up required by the motherboard and the crossbar while the second section of the boot time software is associated with the set up of the daughter cards. The software for the daughter card is further divided into two part associated with the two different daughter types of daughter cards that

are currently supported by the switch. The overall structure of the switch booting software is as shown below.

1. Boot DECElx from FLASH ROM
2. Load Motherboard Xilinx
3. Configure Switch
 - Determine Number of Cards
 - if 1 → *MASTER*
 - else Elect *MASTER*
4. Initialize Non-Volatile (FLASH) Data Area
5. Load Management Parameters from FLASH
6. Perform Barrier Synchronization
7. Load Motherboard Xilinx
8. Perform Barrier Synchronization
9. Load Crossbar Xilinx
10. Perform Barrier Synchronization
11. Initialize Motherboard RAMs
12. Perform Barrier Synchronization
13. Reset Motherboard
14. Perform Barrier Synchronization
15. Determine Daughter Card Type
 - if 155 → Initialize using 155 init procedure

- if 622 → Initialize using 622 init procedure

When the switch is powered on or rebooted, the DECElx operating system is loaded from the non-volatile memory (FLASH) and the processor is booted. This is followed by loading of the bit files into the motherboard Xilinxes. Once the motherboard Xilinx are loaded, the switch configuration begins.

The configuration software determines the number of cards present in the switch and elects a *master*. The election of a *master* involves selecting one of the cards to perform the switch management functions such as setting up routing table for the switch, setting up and maintaining the switch crossbar, etc.

The software for loading the motherboard hardware and the electing the *master* has been incorporate in the DECElx kernel stored in the FLASH memory of each motherboard. Each board thus boots independently and establishes a relationship to other cards in the switch.

The application code specific to the AN2 is used to boot the rest of the hardware in the switch. The application software first initializes the non-volatile FLASH allowing the processor to read from its data area. The application code proceeds to load the management parameters from the FLASH memory and then applies barrier synchronization to wait until all cards on the switch are in a similar state. On the successful completion of the barrier synchronization, the motherboard Xilinxes are reloaded and yet another barrier sync is performed to ensure that all the card on the switch have reached the same point in the boot sequence. A successful completion of this barrier synchronization is followed by the elected *master* loading the crossbar Xilinxes. A barrier synchronization is performed at this stage to ensure that all crossbar Xilinxes have been loaded correctly.

The application code then proceeds to load the motherboard RAMs with the values required to correctly set up the permutation and mapping tables for routing. A barrier synchronization is performed and a reset signal is applied to the motherboard in order to bring all the motherboard and crossbar hardware to a

known, stable and functional state.

At this point the application code moves on to boot the daughter cards connected to the motherboards. The application code reads a particular location in the FLASH memory and determines the daughter card type, set up its hardware and boots the daughter card . In case a 622 Mb/s ATM/SONET Gateway Card is present then the boot sequence described in Section 3.2 is used.

3.2 ATM/SONET Gateway Boot Process

Figure 3-1 shows the procedures that must be followed to boot the ATM/SONET Gateway card. The process begins by loading the Xilinxes with the appropriate code. Since the gateway card can operate in one of two modes, the OC-12 or 4 x OC-3, the mode of operation chosen determines the version of the Xilinx code loaded from the FLASH ROM and when the Xilinxes are programmed.

3.2.1 Control Register

The boot process sets up a control register that controls the various option that can be used on the SONET and ATM streams, sets up the window for the FLASH ROM, enables or disables loopback, and determines the start up state of the lasers.

The initial configuration for this control register is:

- Disable ATM cell descrambling on receive
- Disable SONET scrambling on transmit
- Disable ATM cell scrambling on transmit
- Disable Lasers
- Enable Sync Recovery on transmit
- Enable Frame Recovery on receive

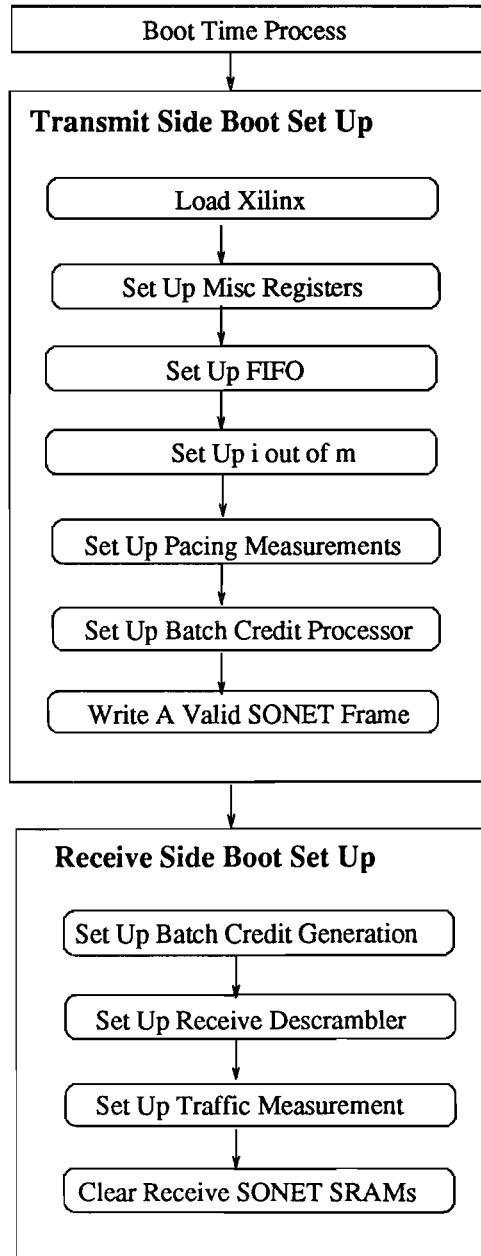


Figure 3-1: ATM/SONET Gateway Card Boot Processing

- Set Pacing Measurement Bank

To set up this control register with the preceding configuration, the LCP writes the value 0x00CF to the location 0x1EC0 0000. The run time process can change the value written to this location based on the options the user wants to enable or disable [9]

3.2.2 Transmit SONET Overhead SRAM

The SRAM for the transmit side begins at the base address of 0x1EC08000. Some field of the SONET frame are written by hardware, some fields are permanently setup and some fields are written by the LCP based on error processing described later in this chapter. SRAMs for the SONET overhead are broken up into two different halves or banks. Each bank is made up of SRAMs that are used to hold overhead information about 8 SONET frames. These banks are initially loaded with the permanently setup fields of the overhead [4, 30].

This will allow the resident hardware to receive cells from the AN2 crossbar and inject the cells and overhead written to the SRAMs into the transmit stream.

Some fields in the SONET overhead are unused. These fields will be set to all zeros in accordance to the specifications [4]. All the SONET overhead fields and their values are found in Appendix A.

3.2.3 FIFO Controller

The ATM/SONET application code must set up certain control registers on this Xilinx and clear all interrupts before the first SONET frame can be filled with ATM cells. In order to get the hardware to this state, the LCP disables all interrupts by writing the value 0x0000 to the address location 0x1EC3 8024.

Once interrupts have been disabled, the FIFOs need to be programmed according to the mode of operation. For the 4xOC3 case, the FIFO controller sets

up 4 small FIFOs that will be served in a round robin fashion. For the OC-12 case, a single large FIFO is set up. The programming of the FIFOs is done in hardware. To program the FIFOs, the value 0x0028 (decimal 40) is written to the last byte of the location 0x1EC3 802C. When the FIFOs have been programmed and set up, the boot process clears all interrupts that might have occurred during the programming of the FIFO's by writing the value 0x0001 to the location 0x1EC3 8020 [9].

At this point, the hardware is in a known state, and the boot process can set up the initial transmit frame interrupt interval. This specifies the frequency at which the FIFO controller will interrupt the LCP so that the LCP can write the SONET overhead information to the SRAMs. There are 5 possible interrupt intervals - 0, 1, 2, 4 or 8 frames. The value is determined by the system programmer based on the error reporting scheme that is chosen. At the present time the value for the interval is set to 8 frames. The interrupt interval is set up by writing the value 0x0008 to the location 0x1EC3 8024.

3.2.4 i out of m

The *i* and *m* SRAMs used by the i-out-of-m controller and the adaptive shaper process on the LCP need to be cleared by the application code. This is necessary since the SRAMs must be in a known state. The i-out-of-m controller uses bit eleven in the *i* SRAM to determine if a VC is being paced. A zero at this bit indicates to the i-out-of-m controller that a particular VC is not being paced while a 1 in that bit indicates that a VC is being paced [9]. At start up since it is known that there are no active VCs, clearing the SRAMs is required to indicate the proper state. The application code writes 0x0000 to a block of SRAMs (16384 bytes) beginning at the address location 0x1EC1 8000.

3.2.5 Batch Credit Processor

The boot process disables batch credit processing. This is the initial configuration for this piece of hardware since no VCs are active and there is no need for processing credits that may arrive from the far end. Batch credits can be received on four different cell streams, so this Xilinx has four header registers which can be individually enabled and disabled via control registers used for this purpose. The value 0xFFFF in these control registers indicates that a VC is enabled and 0x0000 to indicated that a VC is disabled. All other values can cause unpredictable results.

Therefore, it is necessary to explicitly disable batch credit processing by this Xilinx. This is achieved by writing the value 0x0000 to the control register and header located at 0x1EC4 8000, 0x1EC4 8004, 0x1EC4 8008, 0x1EC4 800C, 0x1EC4 8010, 0x1EC4 8014, 0x1EC4 8018, 0x1EC4 801C [9].

The run time process can alter the values in these locations when it determines that the user wishes to perform batch credit processing on credits being generated by the far end of the VC.

3.2.6 Descrambler

The application code for the ATM/SONET gateway boot process must set up certain control registers on this Xilinx and clear all interrupts before the first SONET frames containing ATM cells can be received. In order to disable interrupt generation by this hardware unit, the LCP is writes the value 0x0000 to the address location 0x1EC3 8008.

Once all interrupts generations have been disabled, the application code clears all interrupts that might have occurred during the initial start up. Clearing of interrupts is achieved by writing the value 0x00FF to the location 0x1EC3 8004.

Now that the hardware is in a known state, the boot process sets up the initial receive frame interrupt interval. This is the interval at which the Descrambler will interrupt the LCP so that the LCP can read the SONET overhead information in

the SRAMs. There are 5 different interrupt interval that can be specified - 0, 1, 2, 4 or 8 frames. The value is determined by the system programmer based on the error reporting scheme that is chosen. In the current system software the interval is set to 8 frames by writing the value 0x0008 to the location 0x1EC3 8008 [9].

3.2.7 Traffic Measurements

The application code used to boot the gateway card initially disables the ability to perform traffic measurements since no VCs are active and hence no measurement can be performed. When traffic measurement are performed, this chip can output VC header information on three different cell streams. Hence, this Xilinx has three header registers which can be individually enabled and disabled via control registers used for this purpose. The value 0xFFFF in these control registers indicates that the VC is enable and 0x0000 to indicate that a VC is disabled. All other values can cause unpredictable results.

Hence, it is necessary to explicitly disable the ability to perform traffic measurements by writing the header registers and control registers at 0x1EC5 0004, 0x1EC5 0008, 0x1EC5 000C, 0x1EC5 0010, 0x1EC5 0014, 0x1EC5 0018 on the Xilinx with the value 0x0000 [9].

The run time process can alter the values in these locations when the user wishes to perform traffic measurements on an active VC.

3.2.8 Batch Credit Generation

The gateway card boot process disables the generation of batch credits. This is the initial configuration for this hardware since no VCs are active and hence there is no need to generated credits. The batch credits can be transmitted on four different cell streams. Hence, this Xilinx has four header registers which can be individually enabled and disabled via control registers used for this purpose. The value 0xFFFF in these control registers indicates that a VC is enabled and 0x0000

to indicate that a VC is disabled. All other values can cause unpredictable results. Therefore, it is necessary to explicitly disable the generation of batch credits by writing the value 0x0000 to the address locations 0x1EC4 0000, 0x1EC4 0004, 0x1EC4 0008, 0x1EC4 000C, 0x1EC4 0010, 0x1EC4 0014, 0x1EC4 0018, 0x1EC4 001C [9].

The run time process alters the values in these locations when the user wishes to perform batch credit generation.

3.3 ATM/SONET Gateway Run Time Processes

The run time software is executed when the switch is normally operating. This portion of the system software handles setting up and managing the virtual circuits on the switch, processes that run on a periodic basis such as the task related to the i-out-of-m processing. It also is responsible for processing interrupts and other signals that are generated in response to certain events on the gateway card. In addition to these kinds of asynchronous and periodic event processing, the run time code is responsible for reporting switch performance and errors to host management stations. This section deals only with the processing required by the ATM/SONET gateway card during run time. Figure 3-2 shows all the processing that the LCP must provide during the normal operation of the switch.

3.3.1 Interrupt Handling and Processing

There are two interrupts to the LCP that can be generated by the ATM/SONET gateway. However, in the present implementation of the gateway, only one is used. The following conditions can generate an interrupt:

- Loss of Signal (low to high and high to low transition)
- Loss of Frame (low to high and high to low transition)

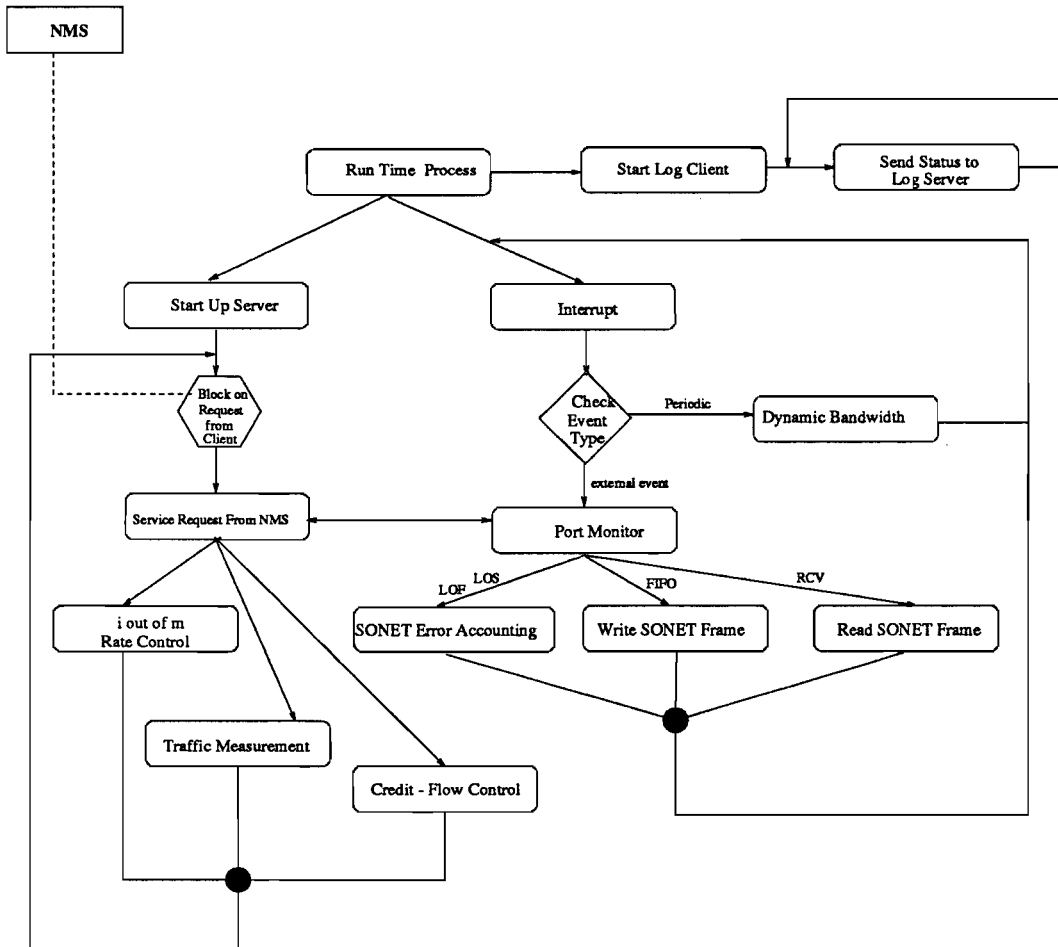


Figure 3-2: Run Time Processing

- Receive Frame Interrupt (indication to LCP to read receive SONET overhead)
- Transmit Frame Interrupt (indication to LCP to fill transmit SONET overhead)

If multiple events occur in the same time slot, the *descrambler* generates a single interrupt and sets the appropriate bits in the status register. For example, if a loss of frame (high to low transition) and a receive frame interrupt occur in the same time slot, then bits 4 and 1 of status register are set.

When the LCP receives an interrupt, the interrupt service routine (ISR) reads the Xilinx register at location 0X1EC3 8004 into a message queue and increments an global interrupt counter. The nature and causes of each of the interrupt type is described below.

Loss of Signal

The loss of signal interrupt occurs when an all zero pattern is detected in the SONET stream for $100\mu\text{s}$ or longer and will last until two consecutive valid frame patterns are detected. Hence there are two signals, a loss of signal high to low transition indicates that an all zero pattern has been detected and a low to high transition indicates that two consecutive valid patterns have been detected.

Loss of Frame

The loss of frame interrupt occurs when the out-of-frame signal on the incoming stream persists for $3\mu\text{s}$ or longer and will last until a continuous in-frame signal is detected for at least $1\mu\text{s}$. Hence there are two interrupts, a loss of frame high to low transition indicates that the out-of-frame has been detected on the incoming signal and a low to high transition indicates that in-frame has been detected.

Receive Frame Interrupt

The receive frame interrupt occurs when the frame interval number is reached. At boot time, this value was set to 8. Therefore, when 8 SONET frames have been received and their overhead information has been extracted and stuffed into the receive SONET overhead SRAMs, the descrambler Xilinx generates an interrupt and set bit 1 in the status register. This implies that the LCP may read the error count fields in the SONET frames.

Transmit Frame Interrupt

The transmit frame interrupt is very similar to the receive frame interrupt. It occurs when the frame interval number is reached. At boot time, this value was set to 8. Therefore, when 8 SONET frames have been transmitted and their overhead information has been used to fill the transmitted SONET frames, the *descrambler* Xilinx generates an interrupt and set bit 0 in the status register. This implies that the LCP may now set up the SONET frames that will be needed to send SONET frames in the future.

3.3.2 Port Monitor

The port monitor is the process that actually services an interrupt. The port monitor reads the message queue and takes the necessary action for each type of interrupt.

Loss of Signal

A count of the number of loss of signal is maintained and is used to fill in the Far End Error Reporting bytes of the transmit SONET frame.

Loss of Frame

A count of the number of loss of frames is maintained and is used to fill in the Far End Error Reporting bytes of the transmit SONET frame.

Receive Frame Interrupt

When this interrupt is being served, the LCP reads the receive frame index to determine which frame is being written. This helps the LCP decide which bank of the SONET overhead is not currently being written into and can thus be read. The LCP then uses an SRAM to SONET overhead mapping function to read the Z2-3 byte used to indicate Far End Block Errors, K2-1 byte used to indicate the Far End Receive Failures and the G1-1 byte used to indicate Path Receive Far End Block Errors. This error count information is then used by other management entities (eg. SONET management station).

Transmit Frame Interrupt

When this interrupt type is being served, the LCP reads the transmit frame index to determine which frame is being read. This helps the LCP decide which bank of the SONET overhead is not being read from and can thus be written to without creating a data inconsistency. The LCP then uses the mapping function to write the Z2-3 byte used to indicate Far End Block Errors, the K2-1 byte used to indicate the Far End Receive Failures and the G1-1 byte used to indicate Path Receive Far End Block Errors. This error count information is obtained from the data structures used to count the Loss of Signal and Loss of Frames.

3.3.3 Management and Configuration Server

The management and configuration server is a process that runs on the LCP allowing it to communicate with a network managements station (NMS). The

communication between the LCP and the management station is implemented using standard TCP/IP socket as shown in Figure 3-3.

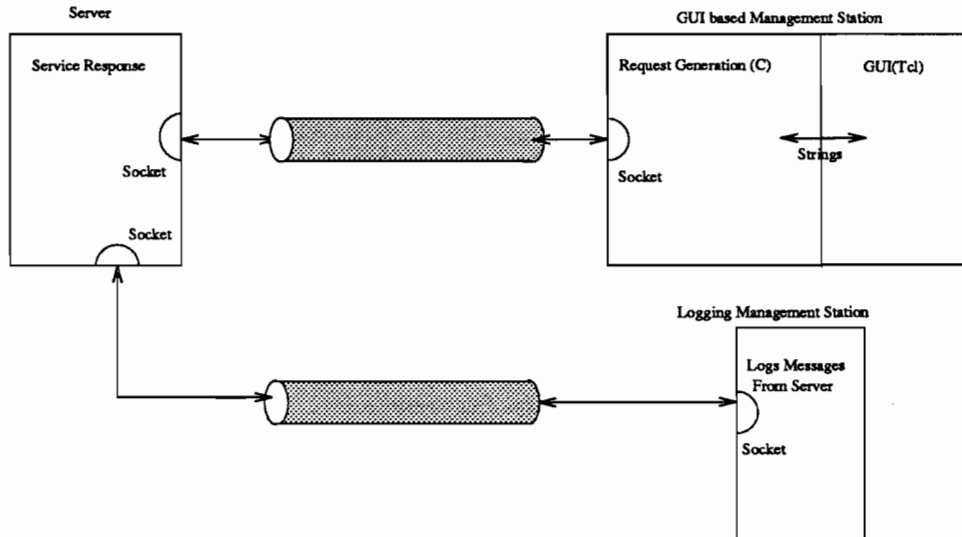


Figure 3-3: Server - Management Station Communication

The management and configuration server can be used to configure pieces of hardware on the switch using the GUIs on an NMS. The GUIs are implemented using a combination of Tcl/Tk and C. The switch configuration software on the NMS is designed to allow debugging of SRAMs and Xilinx registers during the development process and as an interface to network managers during normal operation. The interface software provides many services, including the ability to assist them in set up of VC (both SVCs and PVCs) and to set up the support hardware on the gateway card to measure, rate control or flow control some of the VC that are active.

In the debug mode, the server allows the design and development engineers to examine the contents of the SRAMs and to control some of the ATM and SONET options of the gateway card. In the configuration and management mode, the network manager can select some of the options available such as assign i and m values to a VC, set a PVC, set up a VC to be shaped according to the traffic parameters of the VC, etc. Examples of the debug and configuration GUIs used

on the NMS can be found in Appendix B.

The network management station and the server on the LCP communicate via a specially developed protocol. This protocol is used to pass variable length messages between the management station and server. The request message from the NMS has a format that includes a fixed length header message and a variable length service message.

The header message is further sub divided into two fields :

- Message Type - used to determine the type of service being requested
- Message Length - used to determine the length of the service message

The service message is any one of a large number of defined message types. Some of the message types available are the SONET read (fixed length), SONET write (variable length), i out of m tables (variable length), memory debugging information (fixed and variable lengths), etc. Based on the management station requests, the server assembles the required data into an appropriate structure and using a similar messaging scheme as described above communicates with the management station. These messages provide the data or acknowledgement to a request message sent by the NMS.

Both the management station software and server are implemented using non blocking sockets that allow the asynchronous events access to handlers for the file descriptor associated with the server socket. This means that interrupt service routines that need to notify the management station of an event do not have to wait for the file descriptor to become available due to a request from the NMS. Similarly, the NMS does not have to wait for a message to go out before it can read an incoming message from the server.

In addition to a GUI based NMS, a host connected to the source can be used to log status and information service on the LCP and switch. The LCP connects to this host computer after successfully booting up and sends messages on the status

of the LCP and the switch at regular intervals. This information is recorded to provide a log of operations that have occurred while the switch is active.

The system software on the LCP thus supports a wide range of operations associated with the development and operation of the gateway. The software designs most important feature is its extensibility since it is easy to add new features and modify existing ones.

Chapter 4

Dynamic Bandwidth Allocation

This chapter is organized as follows: Section 4.1 provides the motivation for performing bandwidth management and the need for adaptive shaping as a part of a dynamic bandwidth allocation mechanism; in Section 4.2, a detailed discussion of the proposed control algorithm is presented; in Section 4.3 a detailed discussion of the simulations set up and the results obtained are presented. Finally in Section 4.4 a hardware/software implementation of this algorithm is provided.

4.1 Motivation

Network resources can be over taxed by customers thereby congesting networks if no constraints exist. Therefore, there is a need for resource management schemes that allocate network resources such as buffers, ports, etc. One such resource that needs to be managed is the bandwidth to a particular customer. Bandwidth management can be divided into three categories: bandwidth *reservation*, bandwidth *limitation* and bandwidth *allocation*. While these three terms are often thought to be the same, there are important and significant differences.

Bandwidth *reservation* dedicates bandwidth to a user such that even if only a fraction of the reserved bandwidth is utilized, the remaining portion of the

reserved bandwidth is not available to any other users in the network. Bandwidth *limitation* constrains the maximum amount of traffic that can be sent by a single user. If the user attempts to send more traffic than the upper limit allows, the non-conforming traffic could be discarded, buffered, or penalized. The third area of bandwidth management, *allocation*, is similar to bandwidth reservation in that the bandwidth is “guaranteed” to be available to the customer; it differs in that if the customer does not use all of the allocated bandwidth, the unused portion is made available to other users.

Bandwidth management may be applied on the customer side or the network side of the ATM User-Network Interface (UNI) [12]. For example, customers are encouraged to produce traffic that does not violate the traffic contract with the network provider. The network side may *police* the customer traffic and penalize nonconforming traffic. To produce conforming traffic the customer may *shape* the traffic between the source and the network. Shaping or smoothing refers to queueing ATM cells and then releasing those cells so that the burstiness of the source is controlled.

There are two methods of performing bandwidth management:

- Static
- Dynamic.

Static bandwidth management approaches in [1, 7, 14] do not reflect the inherently dynamic nature of user requirements and the network state, which may change during the lifetime of the connection. However, dynamic bandwidth management approaches change the bandwidth allocated to a particular customer over time. This may be achieved by determining the behavior of customer traffic and allocating available network resources to accommodate the traffic. An important feature of dynamic bandwidth allocation is the ability to change the bandwidth assigned to a customer based on continuous monitoring of their traffic. Consider

the case where n virtual circuits share a transmission pipe, either a physical or a virtual path. The QoS provided by the dynamic adjustment of the network resources among the n VCs is superior to that obtained by static assignment schemes [5].

In a dynamic bandwidth allocation system, if a customer increases the amount of traffic submitted to the network, the adaptation algorithm should allocate more network resources, and if the customer reduces the traffic, the algorithm should reduce the amount of network resources allocated [5]. Since the allocation is based on the customer's bandwidth usage, traffic characteristics must be monitored. Dynamic bandwidth allocation can be supported by passing messages between the customer premise equipment (CPE) and the network provider equipment (NPE) (Figure 4-1). An adaptive i-out-of-m shaper at the CPE controls the traffic submitted to the network while the adaptive Leaky Bucket policer at the NPE estimates the network state and controls the network resources. In the proposed framework, the communication across the UNI between the CPE and NPE would allow a dynamic bandwidth allocation scheme in which the adaptive shaper monitors the customer's traffic and periodically notifies the adaptive policer of its traffic characteristics. In response to a shaper notification, the adaptive policer attempts to modify the allocated bandwidth in the network and notifies the shaper of success or failure. If the policer was successful in obtaining the required resources, adaptive shaper sets its parameters to conform to the new traffic contract.

Previous algorithms for dynamic traffic control [13, 10, 16, 6, 35, 15] are extremely complex and difficult to implement in hardware, thereby making real time control more difficult. The i-out-of-m sliding-window algorithm provides a mechanism to efficiently control the average throughput and burst characteristics of a virtual circuit [12]. This algorithm is simple and does not involve computations of mathematically complex functions do most other algorithms.

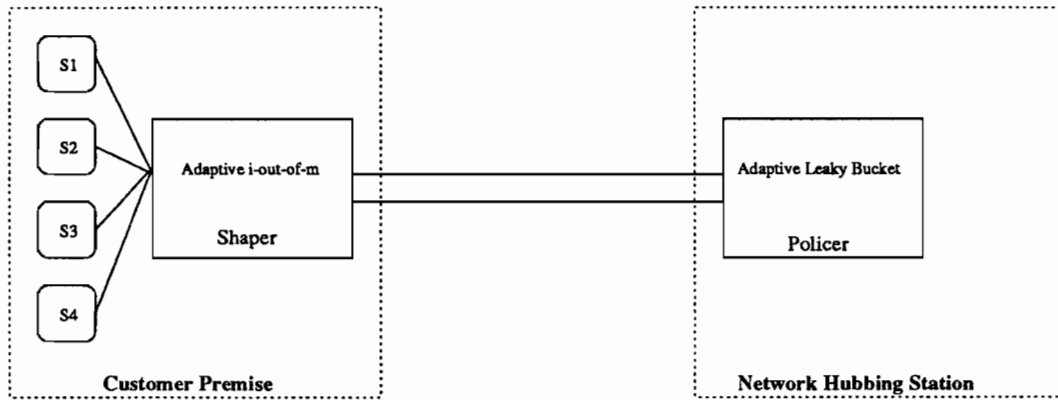


Figure 4-1: Dynamic Bandwidth Allocation System at UNI

This algorithm is suitable for traffic transported in a slotted ATM over SONET environment as provided by the ATM/SONET Gateway Card. Using this algorithm a user can transmit a maximum of i ATM cells in any m cell slot interval, yielding a maximum sustained throughput of i/m normalized to the line rate. The value of i also indicates the maximum number of consecutive cells that can be transmitted by that user. As a VC transmits a cell, the credit bank for that VC is decremented by one and that credit is returned to the VC after m cell slots. The adaptive i-out-of-m shaper will change the i and m values according to the measurements obtained in order to match the rate (i/m) and burst (i) characteristics of the source [5].

4.2 The i-out-of-m Adaptive Shaping Algorithm

4.2.1 Measurement Parameters

There are many approaches to providing adaptive traffic control. An obvious and easy approach to such control would be to monitor the output of the source and to adapt changes in the traffic submitted by the user to the network. However, it is not possible to directly monitor the source output at the ATM/SONET gateway card on the AN2 Switch. The only source measurement that can be obtained is at

the output of the round robin queue on the gateway card as seen in Figure 4-2. The traffic characteristics of the output of the round robin queue differ from the source due to queuing effects, the contention arbitration performed by the DEC AN2. These changes to the source stream make it difficult to develop measurements that accurately estimate the traffic submitted by a source. It was found from experimentation that the average rate and average burst parameters best capture the actual source outputs characteristics and are suitable for the adaptation of i-out-of-m shaper parameters [5].

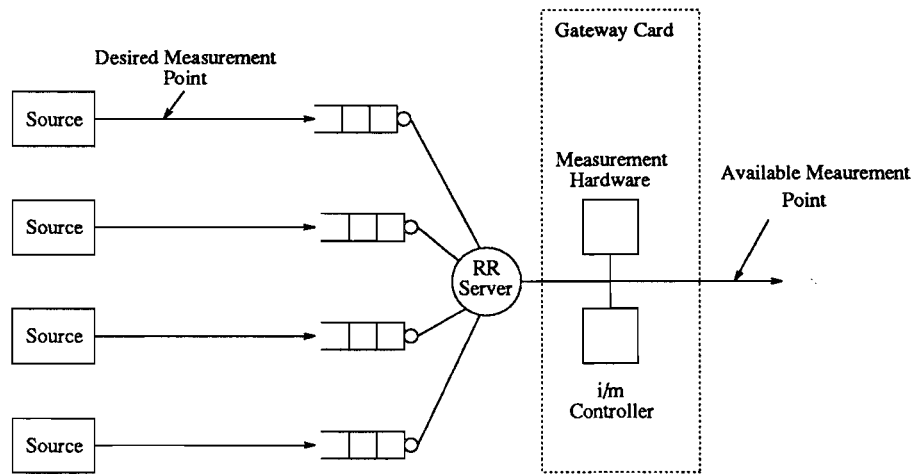


Figure 4-2: Measurement Points on the ATM/SONET Gateway Card

Average Rate Parameter

A running count of the number of cells transmitted on a particular VC in a known measurement interval can be used to compute the average rate for the VC [5]. The average rate is computed by dividing the number of cells transmitted by the time interval during which the measurements were gathered, that is,

$$\lambda = \frac{\text{total number of cells in a measurement period}}{\text{measurement interval}}$$

Average Burst Size Parameter

The measurement hardware also provides the number of bursts that have occurred on a given VC in the measurement interval. This can be used in conjunction with the cell count described above to yield the average number of cells in each burst or the average burst size:

$$\beta = \frac{\text{total number of cells in a measurement period}}{\text{number of bursts in a measurement interval}}$$

For a burst to occur the line must initially be idle (no cells from *any* VC are being transmitted) then become active (at least one cell transmitted on *any* VC), and again go idle. If a VC transmits at least 1 cell during a burst, then the burst count for that VC is incremented [5]. One potential problem with this measurement is that it is possible to concatenate every burst in a measurement interval together. While this result is not an accurate measure of the average burst size for the source it does not adversely affect the algorithm because the adaptation for i is not based on the actual value of the average burst parameter, but is based on the value of the average burst parameter relative to the previous value for i . Another exception to the burst parameter rule is the case of a single active source. Due to a hardware operation on the DEC AN2, a single source cannot transmit back-to-back cells. It can transmit, at maximum, during every other cell slot. According to the burst count mechanism described above, each single cell would be a burst. Clearly, this is an incorrect burst size if the source transmits more than one cell in a burst. To correct this problem, the VC number for the cell currently being served is compared to that of the last cell transmitted. If they are the same, the burst count for that VC is not incremented.

4.2.2 Control Mechanism

This adaptive mechanism can be defined as a two parameter control problem. The two parameters that need to be adapted are the average rate and the average

burst size for a VC. In this algorithm the burst size adaptation is performed first and then the rate adaptation.

Since the measurement variables have been described, the actual algorithm can be presented. Figure 4-3 shows a flow chart of the algorithm. The values for i , m , and the measurement interval need to be initially set by the user. Once a measurement interval is completed, the adaptive shaper process on the LCP receives a signal. At this point the LCP uses the raw measurements taken by the hardware to compute the required parameters (the rate and average burst size) as described above. From the measured rate and the present i and m values the rate utilization is calculated as:

$$r = \frac{\lambda}{i/m}.$$

This parameter measures how closely the shaper is estimating the actual rate of the source. Based on the estimation error the algorithm begins the adaptation of the burst parameter. This is done by comparing the value of i to the average burst size. As can be seen from the flow chart in Figure 4-3 if i is less than $1.1 * \beta$ then i must be increased since the number of credits assigned to the VC is too small. Similarly, if i is greater than $1.3 * \beta$ then i must decrease since the number of credits assigned to the VC is too large. If i is between 1.1 and 1.3 times the value of β no adaptation is performed on i [5]. Once the direction for the adaptation has been determined, the i adaptation is achieved using:

$$i = i + (i * \mu+)$$

for an increase in the number of credits, or

$$i = i - (i * \mu-)$$

for a decrease in the number of credits. The parameters $\mu+$ and $\mu-$ are adaptation parameters [5].

When the burst adaptation has been completed the rate adaptation is performed. Deciding the direction of the rate adaptation is simpler due to the lack of

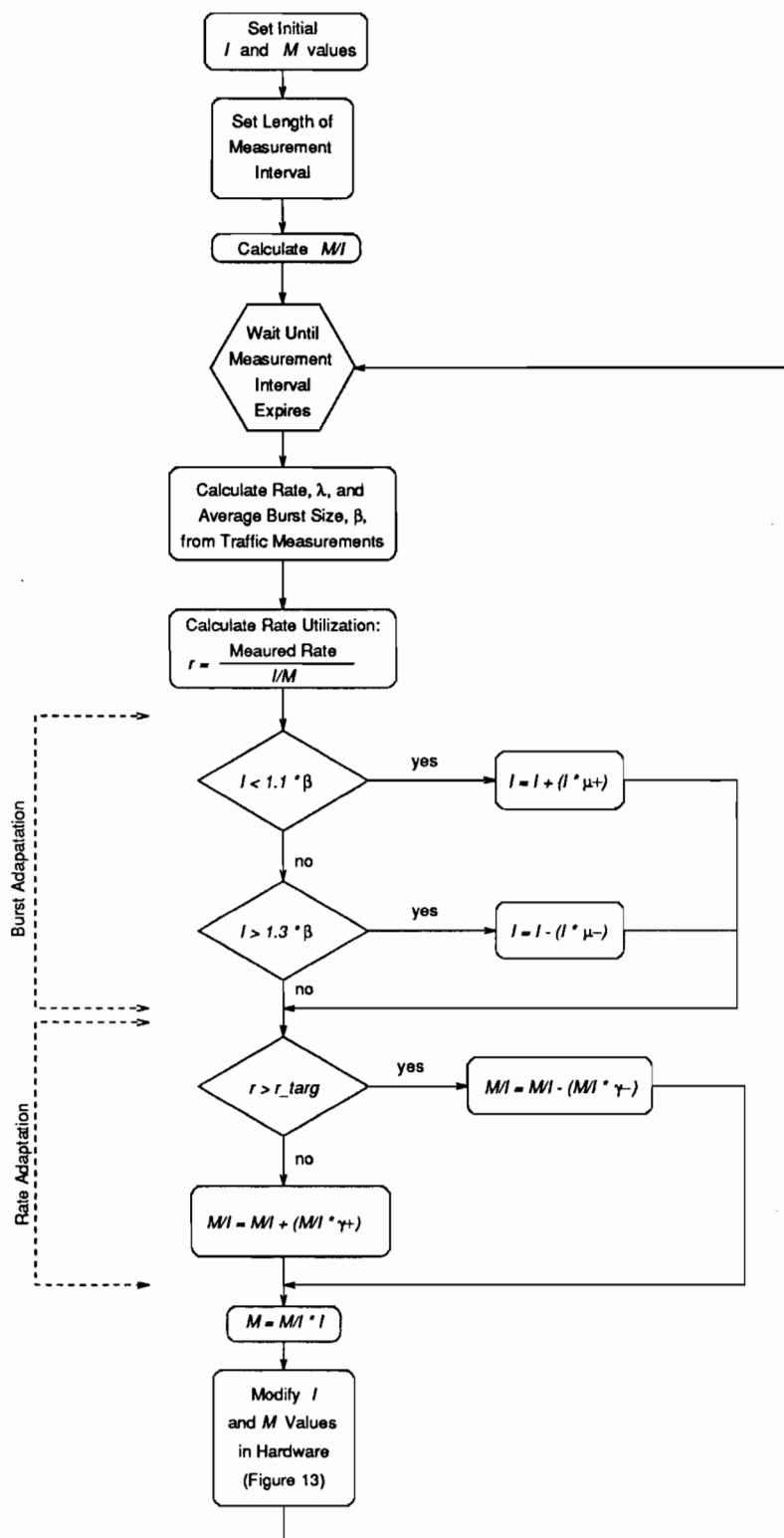


Figure 4-3: Adaptive i-out-of-m shaping algorithm

hysteresis in this adaptation factor. If the normalized rate utilization, r , is greater than a threshold (this is the target utilization), r_{targ} , the rate (i/m) is increased. However if the rate is less than or equal to the target utilization, then rate is decreased. For the simulations presented in Section 4.3, $r_{targ} = 0.85$, meaning that the target shaper rate will be 15% greater than the computed average rate, λ . The parameters $\gamma+$ and $\gamma-$ are the adaptive coefficients for *rate* [5].

$$m = m/i * i.$$

Hence, the rate adaptation is performed according to the equations

$$m/i = m/i - (m/i * \gamma-)$$

for an increase in the shaper rate, or

$$m/i = m/i + (m/i * \gamma+)$$

for a decrease in the shaper rate.

4.3 Simulation and Benchmarking Results

Pioneering work by Braun in the area of adaptive shaping using a sliding window shows that the i-out-of-m adaptive algorithm reacts to changes in the sources average burst size and average rate [5]. In this pioneering and theoretical work, the algorithm was tested for its adaptability to rate and burst changes using periodic and ON-OFF exponential source models. Modifications were made to the exponential ON-OFF source models to provide artificial rate increase for a 50,000 cell slot period. In addition to these source models, a portion of ethernet trace data obtained from Bellcore was used to test the adaptability of the algorithm [17, 5]. These tests were performed using a measurement interval of 1,000 cell slots and an arbitrary rate adaptation constant(γ) of 0.25. The results obtained

from these tests show that the adaptive algorithm provides much shorter average delays than the comparable static assignment algorithms [5].

However, there are some practical considerations which the previous theoretical work did not take into account. A 1,000 cell slot window represents a measurement interval of 0.52ms at 622 Mb/s and 2.08ms at 155 Mb/s and thus is too short a time to support in any reasonable real-time hardware-software implementation of this algorithm. The current real-time implementation of this algorithm uses the i-out-of-m controller and pacing measurement support chip on the ATM/SONET Gateway Card, the DECelx real-time operating system and the LCP (LR33000 processor). This represents very good support for a real-time implementation compared to other similar switches that use the UNIX operating system which is essentially incapable of supporting real time constraints. Even so, DECelx has several practical limitations such as creating a 10ms lower bound on the granularity at which the system can execute time driven periodic tasks. This implies that the 1,000 cell slot (0.52ms) measurement interval used in the previous work is unrealistic for this real-time hardware-software implementation of the algorithm. Further, the goal of the algorithm is to adapt to *substantive* changes in the source rate. However, a short measurement interval forces the algorithm to adapt to fine grain variance in the average source rate. This is an added disadvantage of the short measurement interval used in the previous work.

A final problem with the previous work is that its simulations were run for 100,000 cell slots which represents only 52 ms of data at 622 Mb/s and 208ms of data 155 Mb/s [5]. This is too short an interval to adequately test the algorithm since most sources requiring adaptation remain active in excess of seconds. For example sources such as video-on-demand, multimedia conferencing and other real-time sources stay active for several minutes at a time and sustained rate changes occur over hundreds of milliseconds. Therefore a 52ms or 208ms simulation period is not realistic. To perform more realistic tests, the sources should be active for at

least a second (500,000 cells slots) which would be more representative of sources that would exist in the network.

Before a real-time implementation of the adaptation algorithm could be attempted, some benchmarking of the system supporting the algorithm was required. The main benchmark performed, helped define the timing constraints on the *i*-out-of-*m* algorithm. For these tests, the priority of the adaptive shaper task was assigned various levels and run under DECelx. Giving the adaptive shaper task the highest priority implies that it will be the only process running on the LCP and provides the best case performance of the system. It also provides the firm lower bound on the period at which the task can be run. The results from this test provided a 5.1 ms lower bound for execution period of this adaptive shaper task.

Priority	1 highest	10 high	20 medium	50 medium	99 shell
Adaptive Shaper Timing (ms)	5.1	5.1	5.1	6.2	12.5

Table 4.1: Timing and Performance of Adaptive Shaper

This period is also not realistic since, from previous chapters, it is known that the shaper task is run in conjunction with other tasks needed for processing data in the switch. The next benchmark test used a representative system load and assigned various priority levels to the adaptive shaper task. The results for this set of experiments are given in Table 4.1. The shaper was given a wide range of task priority, 1 which represents the highest priority process, 10 which represent a high priority process, 20 and 50 which represent a medium priority process and 99 which is the priority given to shell tasks. Most of the processes in representative load had priorities in the range of 30 to 50. The current algorithm adapts the *i/m* value at the end of every measurement interval and hence all 127 VCs rate information are updated every time the shaper task is executed and

hence provides the worst case execution time with a difference of one addition that might be avoided due to the hysteresis in the adaptation of the i value. The values in Table 4.1 provides a reasonable estimate of the worst case execution time for the algorithm. From these results, it can be concluded that the adaptive shaper task can be run at a reasonable priority (50) and will run with other switch tasks. Based on the results of Table 4.1 and the conclusion drawn from it, simulations were performed using the modified ON-OFF exponential source model with the rate artificially increased for a period of 50,000 cell slots. The rate is artificially increased for this period so as to allow a source to have a substantial increase in rate for a reasonable amount of time. These simulations were run for a million cell slots (one order of magnitude higher time than in the previous work) and for a range of measurement intervals. One million cell slots represent approximately 2 seconds of data and is a reasonable amount of time for a source in the network to be active.

Simulations were performed for measurement intervals of 5,000, 12,500, 20,000, 25,000, 50,000 and 100,000 cell slots which represent 10ms, 26ms, 41ms, 52ms, 104ms and 208ms respectively at 155Mb/s. Measurement interval times at 622Mb/s can be obtained by dividing the time values at 155Mb/s by four. Average delay per cell was used as the performance metric as was done in the previous work [5]. Simulations were performed using a static assignment policy and the adaptive i -out-of- m . Simulations of the adaptive case were performed to evaluate the effect of increasing the measurement intervals.

Figures 4-4, 4-5, 4-6, 4-7, 4-8, 4-9 and 4-10 show the rate adaptation for the algorithm for the various measurement intervals mentioned above. These figures show the source rate (dotted line), the measured rate (dot-dash line) and the assigned rate (solid line). All rates are normalized to the output line rate. The source rate is captured at the output of the source while the measured and assigned rate are captured at the output of the gateway card.

ON-OFF Exponential Traffic with Rate Bump for 50000 cell slots								
	Static i-out-of-m	Dynamic i-out-of-m						
	i=10, m=100	1,000	5,000	12,500	20,000	25,000	50,000	10,0000
Delay (slots)	51,877	514	3,000	4,458	6,000	8,266	13,490	20,512
Max Delay (slots)	140,000	5,000	18,000	30,000	42,000	55,000	70,000	81,000
Max Buffers(MB)	7.42	0.265	0.954	1.59	2.22	2.92	3.71	4.29

Table 4.2: Delay Comparison for Adaptive and Non-Adaptive i-out-of-m Shapers

From Table 4.2, it can be observed that the average delay shrinks as the measurement interval gets smaller. This reduction in delay is due to the fact that the algorithm is able to detect and clear the queued back log more quickly.

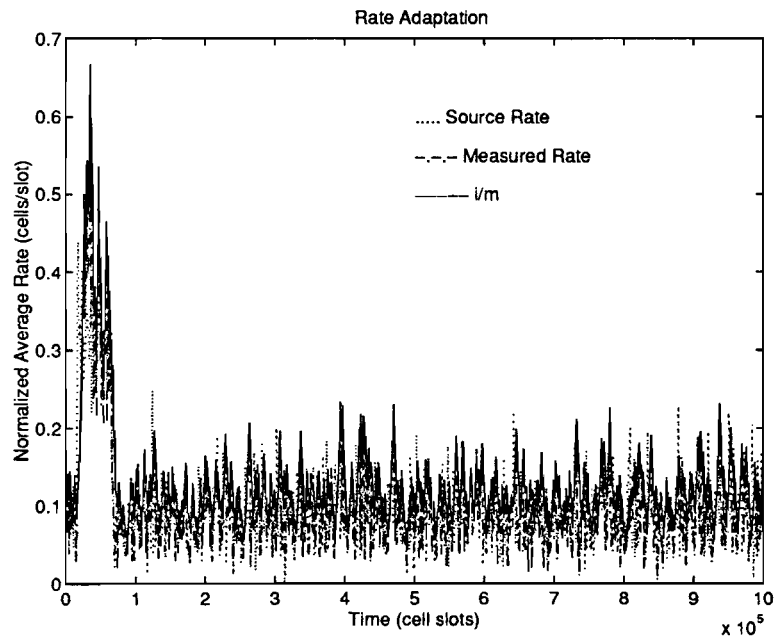


Figure 4-4: Rate Adaptation of Shaper for 1,000 cell slot measurement interval

These results indicate that it is best to run the algorithm at the fastest possible rate. However, Figures 4-5, 4-6, 4-7, 4-8 and 4-9 show that a shorter interval allows the adaptive shaper to adapt to small and insignificant rate changes. A comparison of Figure 4-4 and Figure 4-6 shows that the longer measurement interval has a smoothing effect on the traffic variance and that only significant rate

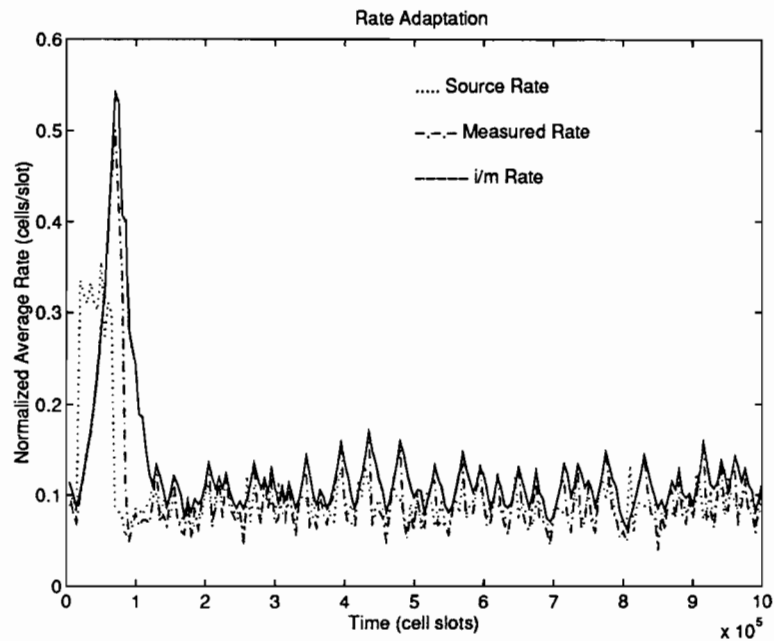


Figure 4-5: Rate Adaptation of Shaper for 5,000 cell slot measurement interval

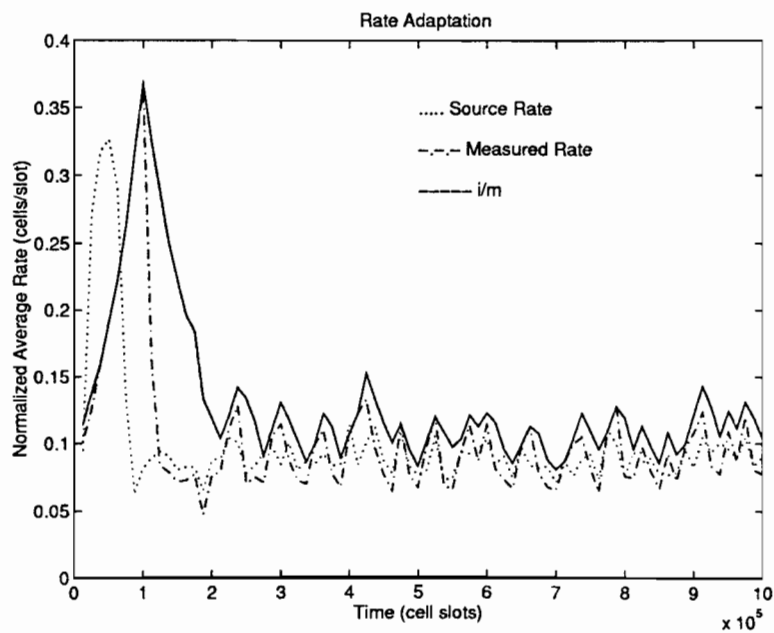


Figure 4-6: Rate Adaptation of Shaper for 12,500 cell slot measurement interval

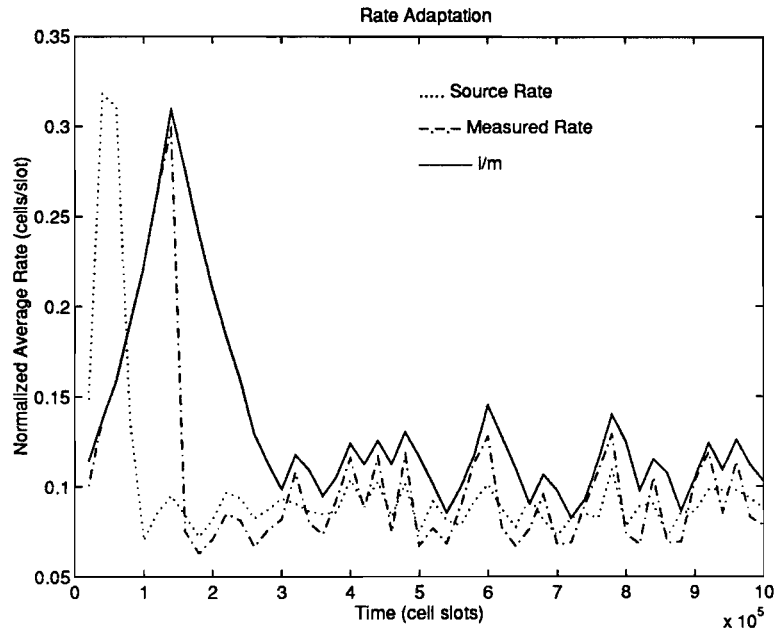


Figure 4-7: Rate Adaptation of Shaper for 20,000 cell slot measurement interval

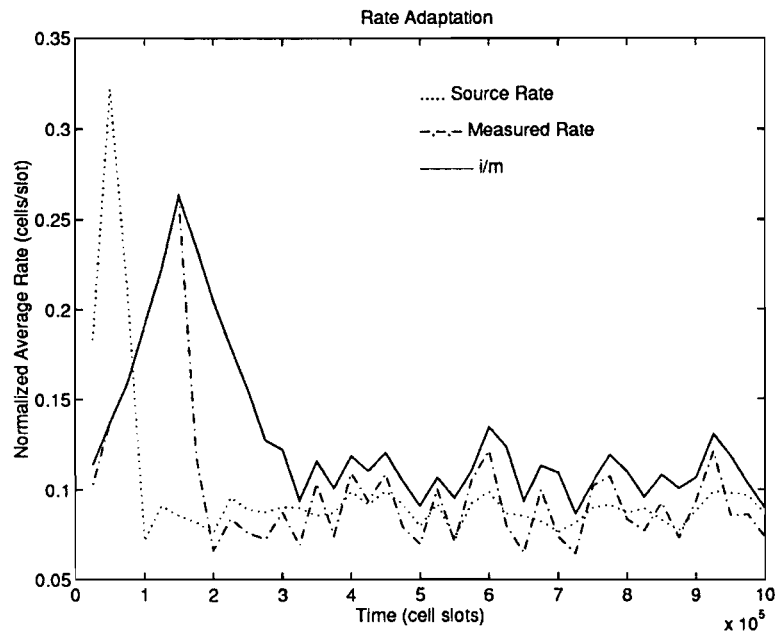


Figure 4-8: Rate Adaptation of Shaper for 25,000 cell slot measurement interval

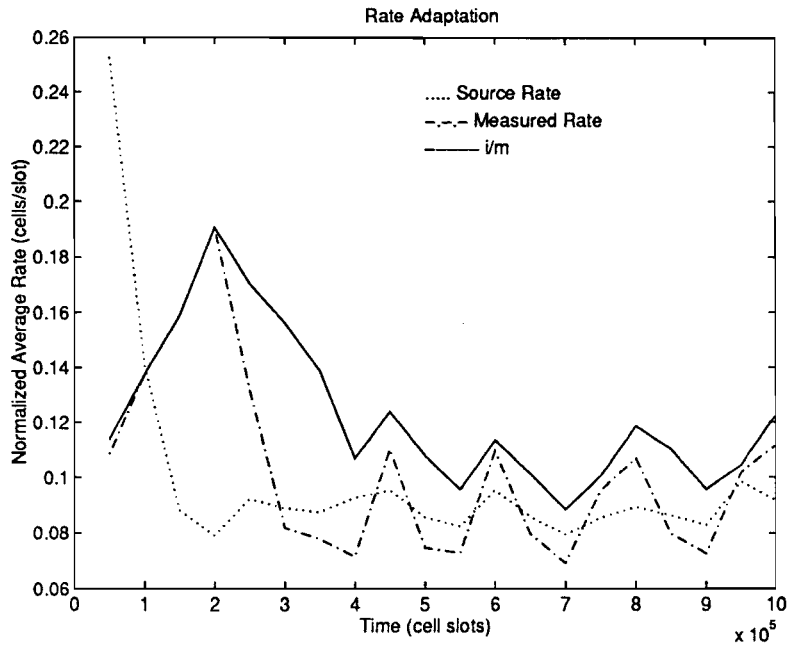


Figure 4-9: Rate Adaptation of Shaper for 50,000 cell slot measurement interval

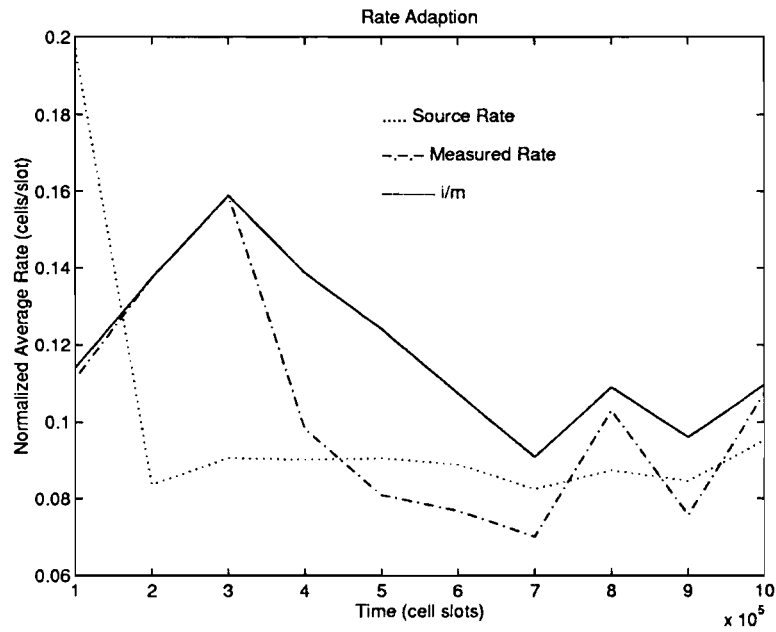


Figure 4-10: Rate Adaptation of Shaper for 100,000 cell slot measurement interval

changes are captured. Although this more realistic measurement interval is 12.5 times that used in the previous study, the observed delays have increased by only 8 fold. This increased delay problem can be improved by selecting adaptation constants, γ_+ and γ_- , that will result in a faster attack for increases in the source rate and slower decay for decrease.

Figure 4-11 shows the effect on the algorithm of increasing the adaptation constant, γ_+ for a 12,500 cell slot interval. As the adaptation constant is increased to allow faster attack for increases in average rate, the average delay was minimized using values in the range of 0.30 and 0.35. These results are for the case where the decay constant, γ_- , was kept at 0.15. The results from these tests, explore the recommendations made in the previous work about improvements to the rate adaptation factors [5]. In this experiment, we explore the cases of $\gamma_+ = \gamma_-$, $\gamma_+ < \gamma_-$ and $\gamma_+ > \gamma_-$. Based on these results, we chose to use 0.32 as the attack constant and 0.15 as the decay constant for further simulations. Using a larger attack than decay constant ensures a more conservative policy. Such a policy tends to favor the customer since it over allocates the bandwidth given to a VC.

In the case where the measurement interval was 100,000 cell slots (Figure 4-10), the source burst of 50,000 cell slots was not detected by the shaper. This is obvious since the event happened within the 100,000 cell slot measurement interval. Due to this, complete adaptation to the rate increase is significantly delayed, leading to queue build up and an average delay of 20,512 cell slots (0.01 sec). This is still significantly better than the 51,877 cell slot (0.02 sec) delay obtained for the static assignment policy. The larger average delay of the static policy is due to its inability to increase its rate and thus fails to clear queue backlog effectively. The graphs show that as the measurement interval is reduced, the algorithm is able to detect and adapt more effectively to rate changes over several measurement intervals.

The results of these simulations also help estimate the buffering requirements,

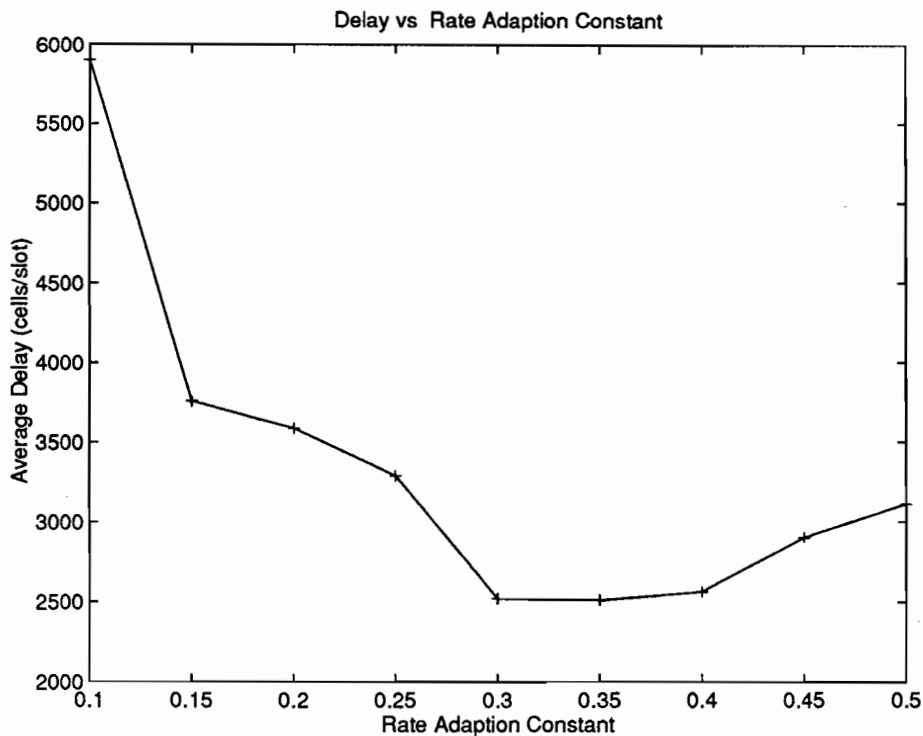


Figure 4-11: Performance of Algorithm with respect to Rate Adaptation Constant

needed at either the switch or the source, defined by the maximum delay that a cell experiences (Table 4.2). From the results, it can be seen that a large buffers would be needed for the traffic types used in this study. Buffering of cells at the switch is more expensive than buffering cells at a source and hence source buffering is preferred. A combination of the i-out-of-m (rate based flow control mechanism) and the DEC AN2 LAN Flowmaster (a credit based flow control mechanism) can be used to buffer cells at the source. In the Flowmaster mechanism, a cell is not forwarded to the switch until a buffer is available to receive the cell. This mechanism ensures that a source does not forward a cell to the switch when the i-out-of-m controller has run out of credits. This happens if a free buffer is not available at the switch and hence, the buffering of the cells would occur at the source. In contrast, switches that do not have a credit based flow control mechanism will require a large amount of buffering - on the order of

a few megabytes per VC.

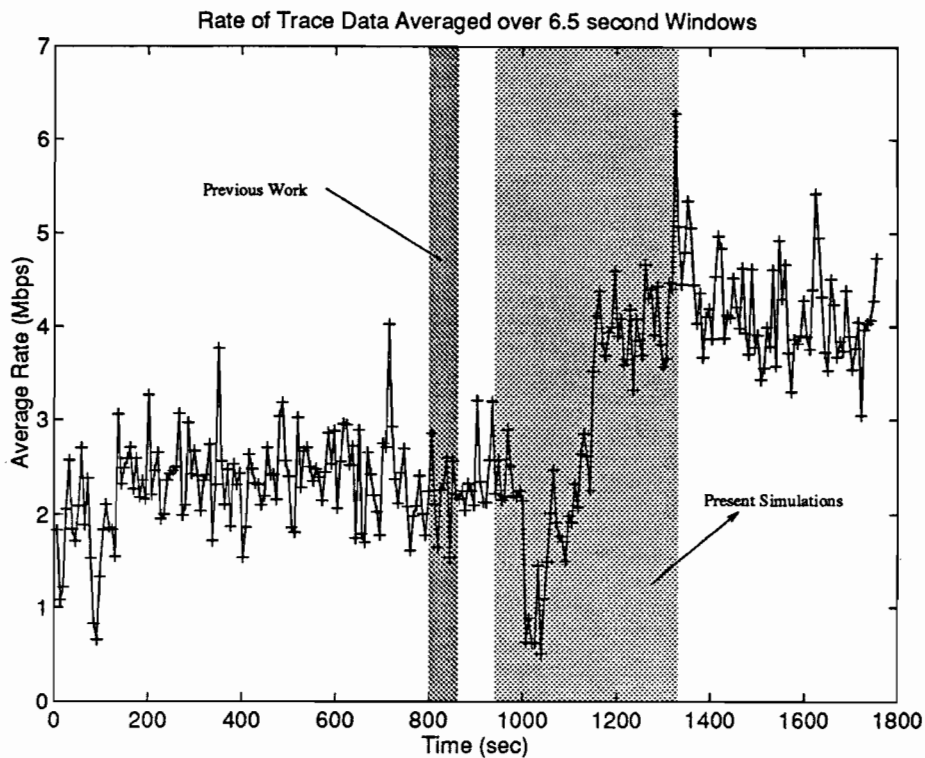


Figure 4-12: Trace data showing amount simulated

All of the preceding simulations were performed using source models. A much better procedure to validate this adaptive algorithm would involve performing simulation tests using traffic from a real source. The traffic trace used in this study is inter LAN ethernet traffic obtained from Bellcore, and represents a reasonable source that might submit traffic to a WAN. Inter LAN traffic is data being exchanged between different LANs. Simulations in the previous work were performed for a very small and random slice (68 seconds) of the available ethernet traces. In addition, the simulations in the previous work were performed on the section of data that did not represent any significant increase in the average source rate. Figure 4-12 shows the trace data averaged using a 6.5 second moving window. The data obtained from Bellcore contains traces for approximately 1,800 seconds. The simulations in the previous work used the data between 800

and 868 seconds in the original data and is shown in the dark grey box of Figure 4-12. However, from this figure, we find that the long term change occurs during the interval starting at approximately 1,000 seconds and continues for a period of 100 seconds. Therefore, a simulation was set up to capture the long term change and to observe the effect of larger measurement intervals on this larger data set than in the previous work. The simulation used a 50,000 cell slot (104ms at 155 Mb/s) measurement interval and used ethernet traces spanning 341 seconds. This simulation provided approximately 3,200 data points. A simulation using a static assignment policy that allocated a rate that was 20 percent over the average source rate was also performed on the same data. Such an assignment policy is considered to be more conservative than most other average rate based policies.

These large simulations were possible since the new version of the i-out-of-m simulator was optimized and intermediate results are stored to maintain state in case of a machine failure.

Figure 4-13 and Figure 4-14 show the results obtained from the simulations using the ethernet trace data. The data used to obtain these figures have been averaged over a one second interval. As mentioned above, these simulations produced about 3,200 data points. Averaging of results need to be performed to make the data more presentable with respect to rate changes that we expect to capture. The average delay using the adaptive shaper with a measurement interval of 50,000 cell slots (104ms) was 16,700 cell slots (0.03 sec) which is much shorter than the 310,930 cell slot (0.64 sec) delay obtained from the conservative static assignment policy. From Figure 4-13 it can be observed that the dynamic shaper adapts to the significant rate change that occurs at the 1,000 second mark in Figure 4-12. The shaper detects the large increase in the source rate and in turn it increases the assigned rate of that VC.

This result is substantially different from those obtained in previous work, which showed that the static assignment policy was better than the adaptive

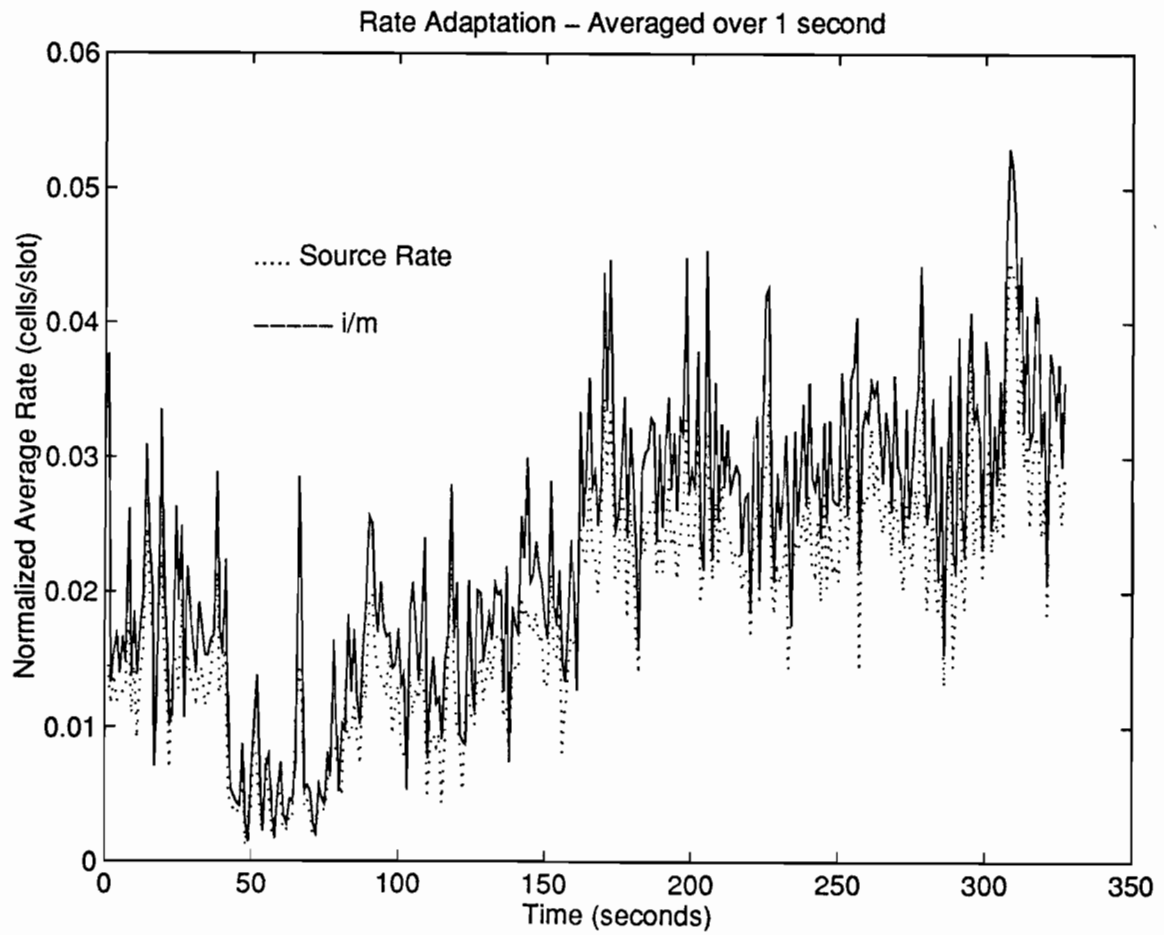


Figure 4-13: Rate Adaptation of Shaper to Ethernet trace data

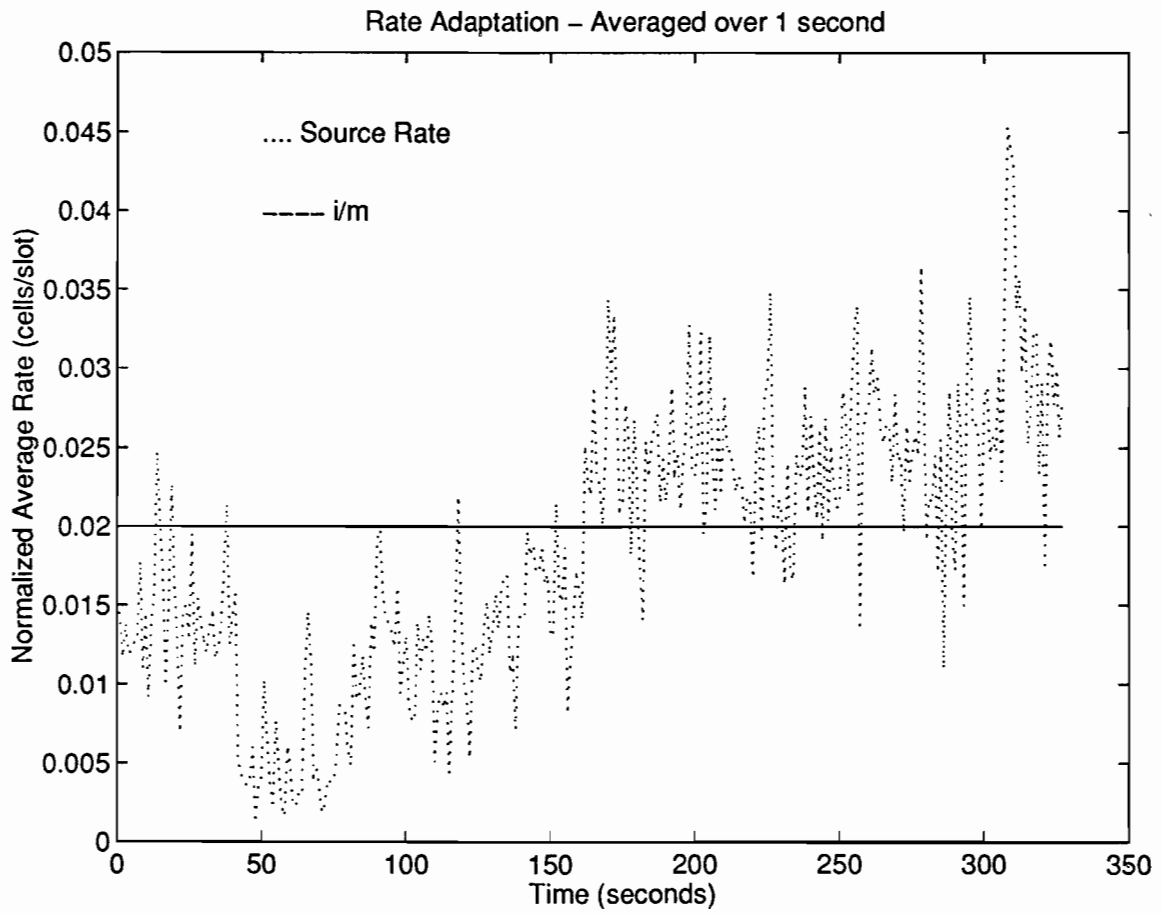


Figure 4-14: Static Rate Assignment on Ethernet trace data

algorithm. In that work, the portion of the trace data used represents an interval during which the source is operating below its average rate. In such a scenario, the static shaper would provide better delay performance since it over assigns bandwidth to the VC, while the adaptive shaper with a small measurement interval (1,000 cell slots) tries to follow the short term variation in the source rate and thereby over or under shoots the actual rate. If a different section of the data had been used in the previous work, the results might have proved that the adaptive algorithm out performed static allocation policy.

Based on the results of these experiments, it can be concluded that a larger measurement interval smoothes the source variance from the source rate and only more substantive rate changes are captured. Therefore, a more reasonable window size larger than the 1,000 cells slots used by the previous work needs to be chosen. A fair estimate of this parameter can be made since the adaptation constant chosen (γ) is known and the percent change we want to detect (Δ) from the average source rate (λ) is known. An estimate of the measurement window can be calculated using the following equation:

$$interval = \frac{\Delta * \lambda}{\gamma}$$

For example, assume a connection that has an average rate of 1Mb/s and the algorithm must adapt to a 2 percent change in rate. Then a reasonable window size based on the above equation is approximately 50,000 cell slots.

The data used for experiments in this work is representative of traffic that may be submitted to a network. Results from these experiments help conclude that the dynamic algorithm performs well in cases where the sources are active for large periods of time and have sustained rate changes that occur over a period of seconds. The results from the ethernet trace data simulations confirm that the adaptive algorithm is effective in monitoring a real data sources. In general, it can be concluded that the i-out-of-m adaptive control scheme is effective in tracking

and following a customer's traffic requirement thus being far more effective than a static shaper.

The benchmarking results of the algorithm also prove that a real-time hardware-software implementation of such an adaptive shaper is possible given the system constraints on the ATM/SONET gateway card. It also proves that it is possible for the adaptive shaper task to co-exist with other switch control software mentioned in the previous chapter.

4.4 Hardware-Software Implementation

This section discusses the hardware - software implementation of the adaptive shaper described in Section 4.2. This implementation of the algorithm uses two Field Programmable Gate Array (FPGA) components, the *i*-out-of-*m* controller and the pacing measurement hardware, and the general purpose LCP. One of the tasks performed by the LCP is the periodic calculations that are involved in computing the new *i* and *m* values as described in Section 4.2.2.

Figure 4-15 provides a view of all the components of the system that are involved in the implementation of the *i*-out-of-*m* control algorithm. The functionality and design of the *i*-out-of-*m* and pacing measurements hardware units are discussed in sections 4.4.1 and 4.4.2 respectively. The hardware can adaptively control a maximum of 127 VC's out of the available 4096 VC's.

4.4.1 *i*-out-of-*m* Controller

The *i*-out-of-*m* controller can be divided into two functional blocks, the flow control hardware mechanism and the credit return mechanism. The flow control hardware unit is responsible for rate controlling the VCs based on credits used by each VC. The credit return mechanism is the open loop credit control mechanism for the flow control hardware unit.

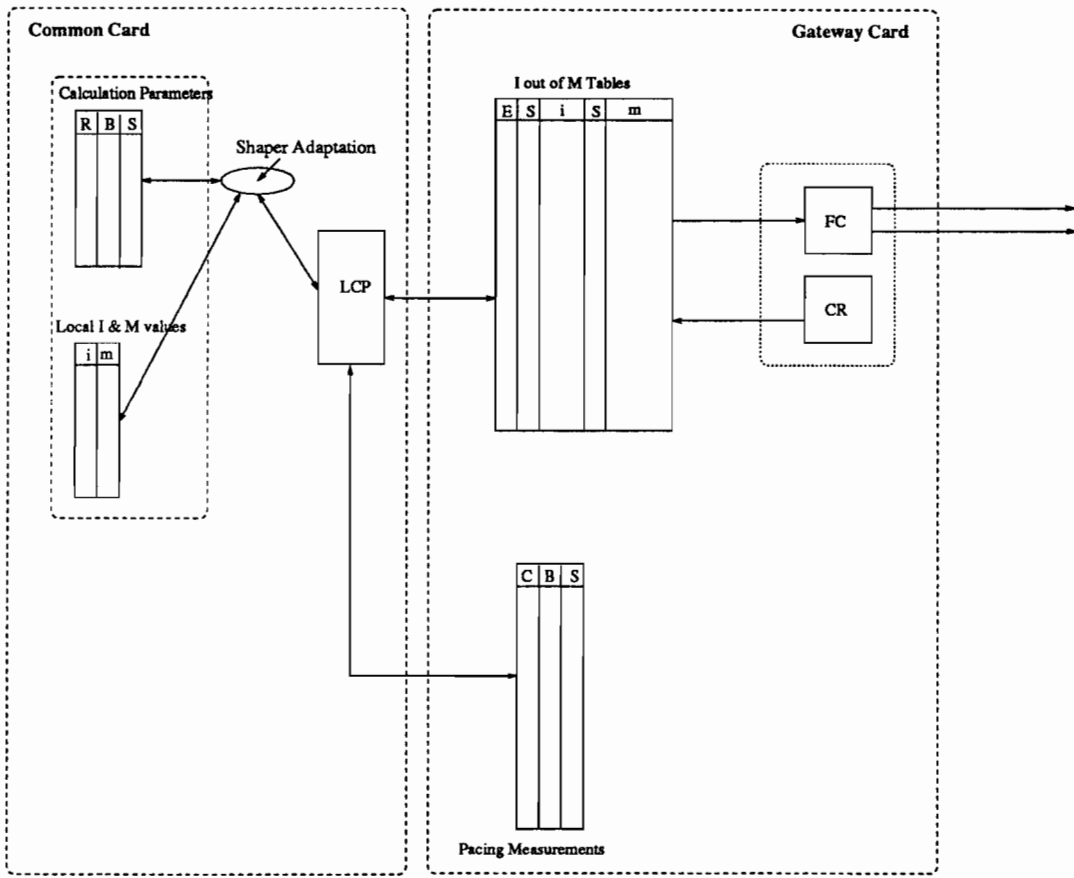


Figure 4-15: System Block Diagram

The *i*-out-of-*m* controller manages several memory structures which contain information on current bandwidth usage relative to allocated bandwidth for each VC. Figure 4-16 shows the *i*-out-of-*m* controller and its interaction with the memory structures.

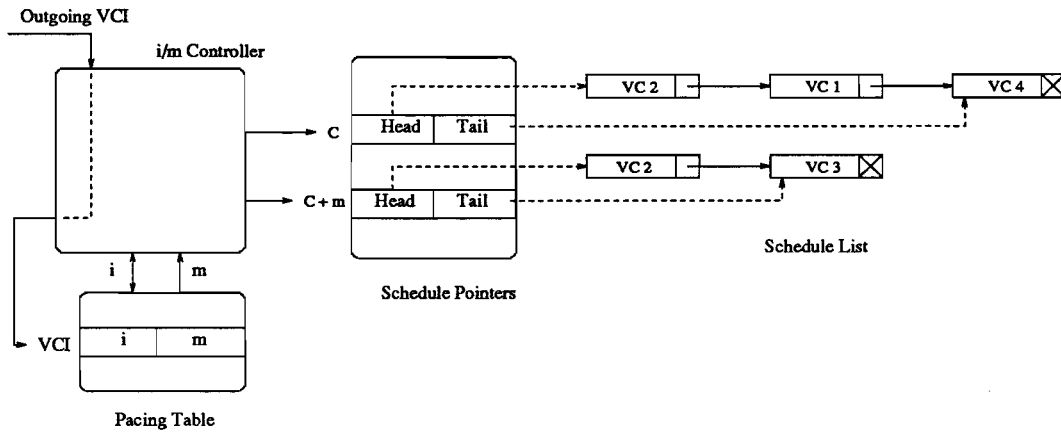


Figure 4-16: *i*-out-of-*m* Block Diagram

Pacing Table

The Pacing Table contains the information needed to control the bandwidth of the VC. This table is indexed by Virtual Circuit Identifier (VCI). For each VCI the table contains a 10 bit signed *i* value, which is the number of credits in the VC account, an 11 bit unsigned *m* value, which is the window size in cell slots, a flag to indicate if the VC is bandwidth controlled or not, and a semaphore bit that indicates to the credit return hardware the modification of a VC's *i* and *m* values by the LCP (see Section 4.4.3). The current size of the Pacing Table is 4096 entries [34].

Schedule Pointers Table

The Schedule Pointers Table keeps track of the pointers for the queues of VCs that are scheduled for credit update at different cell slots. Each entry in this table consists of a 16 bit Head pointer and a 16-bit Tail pointer, which are used as indices

to the Schedule List. The most significant bit of each pointer is a nil indicator. The current implementation supports 2048 entries which is the maximum allowed window size (in cell slots) [34]. Figure 4-17 shows the structure of the schedule pointer table.

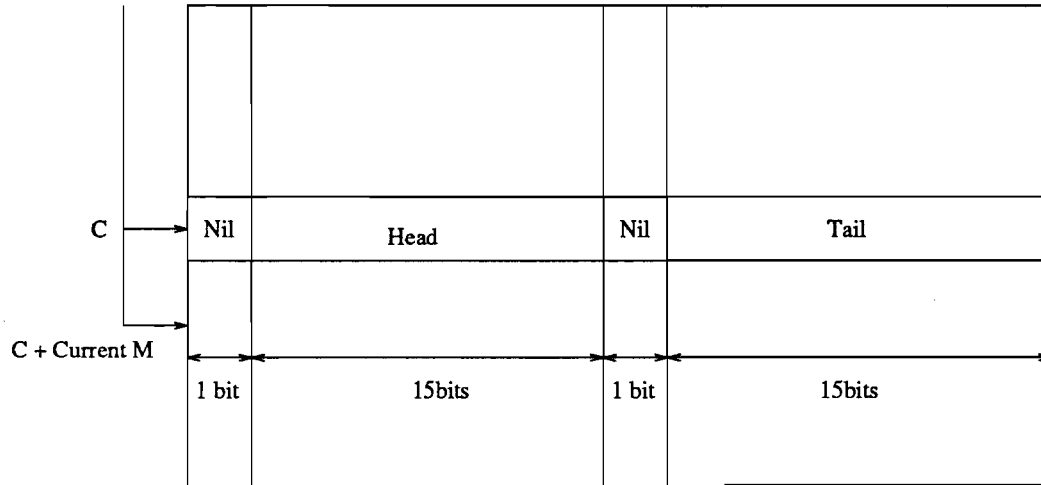


Figure 4-17: Schedule Pointers Table

Schedule List

The VC queues are organized in the Schedule List as linked lists. Each element in the list contains the following: a 12 bit VCI value, whose credit needs to be updated; a 16 bit pointer to the next element in the queue; and a nil flag to indicate the end of the list. The current size of the Schedule List is 16384 entries. Figure 4-18 shows the schedule list memory structure.

Flow Control Hardware

The flow control hardware is a FPGA that decrements the value of i in the pacing table of a VC after a cell is transmitted on that VC and schedules a return of these corresponding credit m slots later. When a cell is transmitted on a particular VC, the flow control hardware uses the VCI to index the Pacing Table and read the i and m values. The i value is then decremented by one.

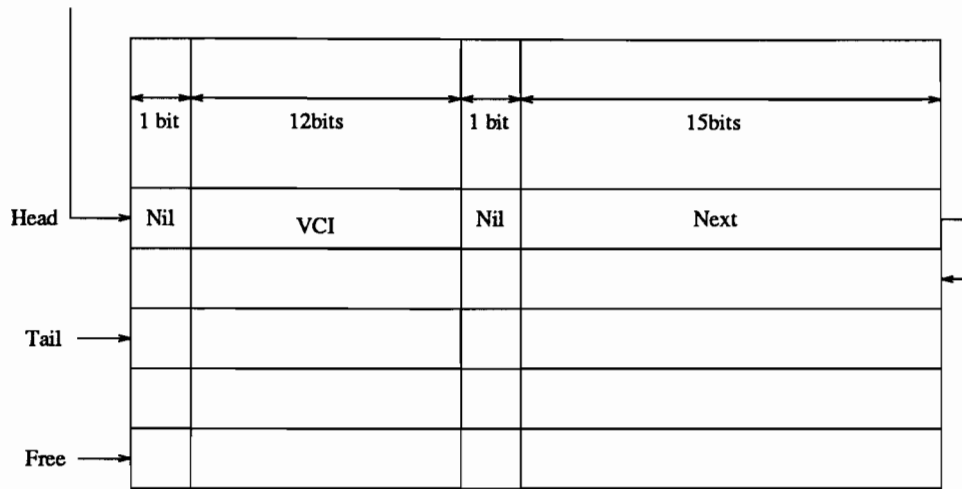


Figure 4-18: Schedule List

If the new i value is less than a threshold value, meaning that the VC has run out of credits, the flow control hardware sends a *Stop* signal back to the source to inform it not to forward any more cells on that VC. The flow control hardware then writes the new i value to the Pacing Table.

The credit for the cell which was transmitted must be added to the Schedule List described in Section 4.4.1. The m value read from the pacing table is added to the value of an internal circular counter to obtain an address into the Schedule Pointer Table described in Section 4.4.1. The circular counter is used as an index to indicate which cell slot is being serviced. The *Tail Pointer* is read from the Schedule Pointer table and is used to add the VC to the end of the queue in the Schedule List [34].

Credit Return Hardware

The credit return hardware is a FPGA that updates the Schedule List and increments the credits of the VCs whose credit return is scheduled for the cell slot indicated by the circular counter. The increment in the i value in the pacing table signifies the return of a credit that had been previously scheduled by the flow control hardware.

The value of the circular counter is the address of the Head pointer of the VC queue that needs to be serviced. The credit return hardware may have multiple VCs that need to have their credits returned at a given time. Since only one VC can be serviced per cell slot (a hardware limitation), the credit return hardware increments by one the credits of the VC at the head of the queue and then removes the VC from the queue. The second VC in the queue is serviced in the next cell slot, and so on, until the queue is empty. When one queue is emptied, the Controller starts updating the credits of the VCs in the next queue [34]. This kind of scheduling can lead to a skew in the credit return process. Figure 4-19 shows an example of a possible credit return schedule and Figure 4-20 shows the actual return of credits as performed by the hardware. The time lost due to the skew in credit returns can be reduced if the aggregate average rate of all VCs on a link is less than the total capacity of the link [28].

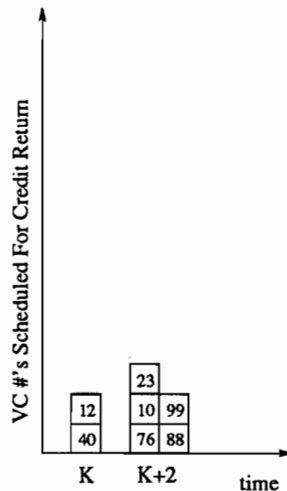


Figure 4-19: Example of Schedule of Credit Return

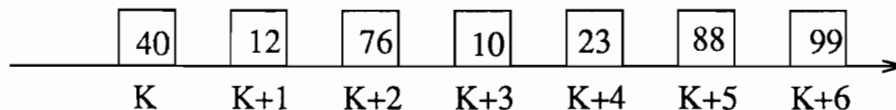


Figure 4-20: Example of Actual Hardware Credit Return

If the new i value becomes larger than a threshold value, meaning that the

VC has credits to transmit, the credit return hardware sends a *Start* signal back to the source informing it to forward cells on that VC.

4.4.2 Pacing Measurement System

The pacing measurement system can be divided into two functional blocks, the cell count sub-system and the burst count sub-system. The cell count sub-system is a hardware unit that is responsible for counting the number of cells that have been transmitted by a VC. The burst count sub-system determines the number of bursts that have occurred in the measurement interval [36]. The cell count and burst count system use two memory structures to store each of their values. There are two identical banks of these structures that allow the LCP to operate on data in one of the memory structures while the measurement sub-systems write data to the other.

Measurement Table

The Measurement Table contains the information needed to calculate the bandwidth utilization of a given VC. Each entry in the table contains a 24 bit unsigned value, which is the number of cells that have gone by on a VC (*cell count*) and a 24 bit unsigned value, which is the number of bursts in which a VC has participated (*burst count*). These are the values that are used to calculate the average rate and average burst described in Section 4.2.1. The current size of the measurement table is 128 entries [36].

Translation Table

The Translation Table provides the mapping of VC's to entries in the measurement table. If a VC is not being measured, the mapping entry for that VC is zero. This is to avoid multiple VC entries pointing to the same location in the measurement table. When a VC is removed from the measured list, the entry for that VC is set

to zero. The translation table is a 4096 entry table indexed by the VCI [36].

Cell Count Sub-System

The cell count mechanism reads the bus to determine the value of the VC being transmitted. It then uses the translation table to determine if the VC is being measured. If the VC is being measured, the hardware increments the *cell count* field for that VC index in the measurement table [36].

Burst Count Sub-System

The burst count subsystem reads the bus to identify which VC is transmitting. It then uses a time stamp based state machine for that VC to determine the time stamp for the last cell transmitted on that VC (*previous cell time stamp*). It also maintains a time stamp of the last time the line was idle (*idle cell time stamp*). The hardware detects an idle line when two consecutive idle cells are seen. These time stamps are required to determine if the current cell is part of a new burst or a continuing burst. The hardware subtracts the value of *present cell time stamp* from the *previous cell time stamp* for that VC and also calculates the time difference from the *present cell time stamp* and the *idle cell time stamp*. If the first value is greater than the second value then this is the first time the VC is participating in a burst and this means that the *burst count* field for the VC needs to be incremented; otherwise, the VC has already transmitted during this burst so the burst count is not incremented. A special condition might occur when a VC is stopped due to lack of credits. In this case, the *stopped flag* is used in combination with the burst count as a solution [36]. The *stopped flag* is set when a VC has run out of credits. If the *stopped flag* was set for a particular VC when a burst ended, the number of bursts for that VC is not incremented when a new burst begins. The reasoning for this is, if the VC had credits when the burst ended it would have been able to continue sending the cells in its queue and the second

burst would not have occurred. By not incrementing the number of bursts for that VC the same result has occurred. This may result in concatenation of every burst in a measurement interval. While this result is not an accurate measure of the average burst size for the source it does not adversely affect the algorithm because the adaptation for i is not based on the actual value of the average burst parameter, but is based on the value of the average burst parameter relative to the previous value for i .

4.4.3 Hardware - LCP Interaction

The LCP and i-out-of-m controller need to interact at the end of each measurement interval. Once the end of a measurement interval is reached, the measurement banks in the pacing measurement system are switched and the pacing measurement table is read to obtain the number of bursts and the total number of cells sent per VC [28]. The adaptive shaper process on the LCP uses this information to calculate the average rate and average burst size for each VC. Based on these measurements and the algorithm provided in Section 4.2, the LCP determines new i and m values.

The difference between the old and new i and m values specifies the number of credits that need to be added or taken away from the current credit balance maintained in the i field of the pacing table. The LCP must thus engage in a Read-Modify-Write cycle on the data in the i field in the pacing table.

One of the very important aspect of an implementation is the difference in the time scale at which the hardware and software computations access the i value in the i and m tables. This implies that there needs to be some mechanism to maintain data consistency among the various pieces of the computations.

Concurrent access to the i and m tables by the LCP and i-out-of-m controller can lead to two different scenarios where the:

1. System may gain

or

2. System may lose credits.

Scenario 1 : System Gains Credits

The system may gain credits in two different cases

1. If the new- i is greater than previous- i (need to add credits to the existing credit count) and cells have gone by on that VC [28].
2. If the new- i value is less than the previous- i value (we need to subtract credits from the existing count) and cells have gone by on that VC [28].

For both these cases, the LCP reads the i tables to determine number of unused credits and then computes the new number to be written [28]. In a case where no concurrent access resolution mechanism is not being used, the number of credits in the i table could change in the time required by the LCP to perform its read, add/subtract and write to the i table. The value in the i table can change between a read and write if a cell is transmitted or a credit is returned on that VC after the read portion but before the write portion of the LCP's READ-MODIFY-WRITE cycle.

In both cases (addition and subtraction), an incorrect value would be written, since a credit decrement would have been ignored leading to a VC gaining credits. Figure 4-21 shows an example of the credit gaining problem. In this case, the measurement interval ends at time $t+4$ and a read of the i value occurs at time $t+5$. The credit remaining at this time is 6. Now the shaper process on the LCP calculates and new i and m values at time $t+8$. The new i value is computed to be 20 giving the VC 10 extra credits to use. At this time the credits remaining are 4 (i.e. two credits have been used up in the time taken for the calculation). Based on the value read at $t+6$ the new available credit entry (i) for the VC is

computed to be 16 and the value 16 is written at $t+9$. However, the right value that should have been written to i is 14. This difference give the VC two extra credit [28].

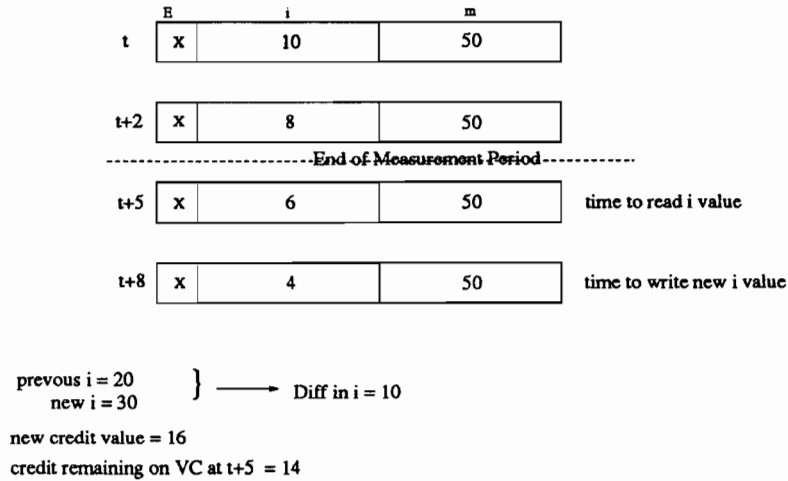


Figure 4-21: Scenario 1: System Gaining Credits

No correction of these values is possible since there is no mechanism in the system to determine what the correct outstanding credit balance. If no data consistency mechanisms is used, the assigned and actual values of i can drift without bound. The situation is worse than it might first appear since a VC's rate can increase over several measurement periods and could gain credits during each. Such a gain of credits on successive cycles could lead to a VC consuming all of the available bandwidth in the system. The maximum number of credits that a VC can gain *per measurement interval* is given by:

$$\text{max credits gained} = (\text{Time taken for RMW})(0.5 * \text{CellRate})$$

Let $I(a)$ be the actual value of the credit that a VC must have and $I(w)$ be the value of the credits that are given to a VC in that measurement interval. Given that we have a RMW time of 3100ns using the above equation, a maximum of 3 cells can go by at 622Mb/s. Figure 4-22 shows the rate of divergence between the actual assigned value ($I(a)$) and value written to the i location ($I(w)$) at 622Mb/s. From the figure it can be seen that the values can diverge without bound.

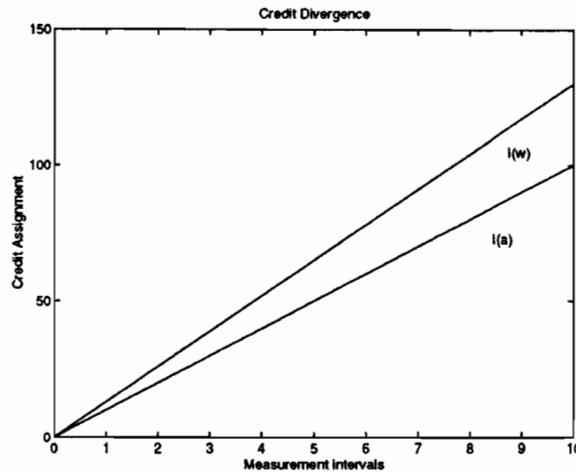


Figure 4-22: Credit Divergence - VC gaining credits

Scenario 2 : System Looses Credits

The system may loose credits in two different case

1. If the new- i is greater than previous- i (we add credits to the existing credit count) and credit returns occur [28].
2. If the new- i value is less than the previous- i value (we need to subtract credits from the existing count) and credit returns occur [28].

In case where no concurrent access resolution mechanism is not being used, the number of credits in the i table could change in the time required by the LCP to performs its read and write to the i table. The value in the i table can changes between a read and write if a cell is transmitted or a credit is returned on that VC after the read portion but before the write portion of the LCP's READ-MODIFY-WRITE cycle.

In both these cases, an incorrect value would be written, since a credit return would have been ignored leading to a loss credits. Figure 4-23 shows an example of the credit loosing problem. In this case, the measurement interval ends at time $t+4$ and a read of the i value occurs at time $t+5$. The credit remaining at this

time is 6. Now the shaper process on the LCP calculates and new i and m values at time $t+8$. The new i value is computed to be 10 giving the VC 10 less credits to be use. At this time the credits remaining are 8 (i.e. two credits have been returned in the time taken for the calculation). Based on the value read at $t+6$ the new credit value for the VC is computed to be -4 and the value -4 is written at $t+9$. However, the right value that should have been written to i is -2. This difference give the VC two less credit.

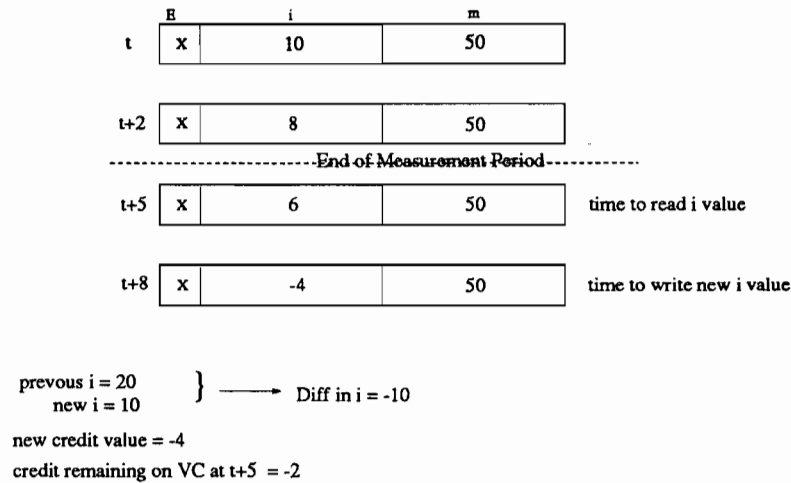


Figure 4-23: Scenario 2: System Loses Credits

No correction of these values is possible since there is no mechanism in the system to determine the correct outstanding credit balance. If no data consistency mechanisms is used, the values in the i register can diverge without bounds from the actual value since a VC can have a rate decreases over several measurement periods and a VC could loose credits on each of these measurement intervals. The rate of divergence between the actual assigned value ($I(a)$) and value written to the i location ($I(w)$) can be obtained using an analysis similar to that used for a VC gaining credits. Such a loss of credits on successive cycles could lead to a VC being starved since the actual value goes to zero.

The data consistency can be maintained via two different mechanism. The first mechanism by which the LCP can maintain data consistency with all other

pieces of hardware is via the addition of two software routines that *start* and *stop* a VC and a semaphore bit in the *m* field of the Pacing table. The credit decrements performed by the flow control hardware as ATM cells go by is prevented during the READ-MODIFY-WRITE cycle by *stopping* the VC. This ensures that no cells come across the crossbar for that VC and therefore no decrement will occur since the flow control hardware will never see a cell on the VC it is modifying the *i* value [28]. The credit increment is also be disabled during the READ-MODIFY-WRITE cycle to avoid losing a returned credit. This is achieved by setting the semaphore bit in the *m* field of the pacing tables. This notifies the credit return hardware that the VC is being updated by the LCP and that its internal counter needs to be frozen until the semaphore bit is unset [28].

The algorithm used by the adaptive shaper process to implement a data consistent READ-MODIFY-WRITE cycle on a shared data structure is given below (Figure 4-24).

1. spl_high (raise shaper process priority so that no other ELX process can interrupt)
2. Stop VC using the an_ec.t lcs_stop_invci (an_port.t inport, an_invci.t invci, an_outvci.t outvci);
3. Set the semaphore bit in the *m* tables for the VC.
4. Busy wait for 2 major Gateway time cycle to allow the values in the *i* settle down
5. DO THE READ-MODIFY-WRITE
6. Upon completion of the RMW (4) cycle, the semaphore bit is unset in the *m* table entry for the VC
7. According to the number that we entered into the *i* tables, the LCP needs to decide if the VC needs to be restarted or should continue to be stopped

(a) If the VC need to started up again using the `an_ec.t lcs_start_invci`
 (`an_port_t inport, an_invci_t invci, an_outvci_t outvci`);

8. `spl_x` (return the DBA process to the original priority level)

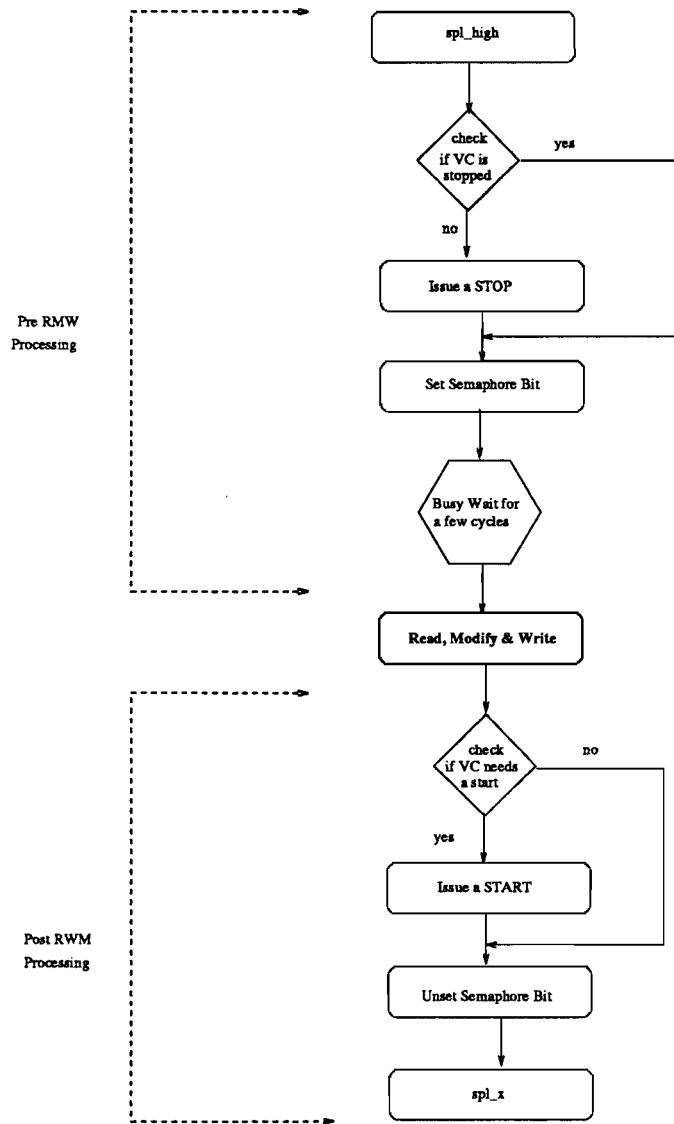


Figure 4-24: Algorithm used to modify the *i* and *m* values in pacing table

The data consistency problem can also be solved by some modifications to the hardware in the i-out-of-m controller in order to provide synchronization of the reads of the LCP and the read on the support hardware. To be able to

understand this solution, a detailed description of the time slot allocation of the i-out-of-m controller must be provided. Figure 4-25 provides the time at which events occur in the i-out-of-m controller. The i-out-of-m controller performs a credit decrement at time T2 and a credit increment at T4 of the 13 sub slots that make up a cell slot time. In this solution, the LCP raise shaper process priority so that no other DECelx process can interrupt and asserts a read to the location that it want to perform the READ-MODIFY-WRITE on. The LCP read request is complete when the *LBDone* signal goes high. In this solution, the issue of this signal is delayed by the i-out-of-m controller until the end of time slot T4. On the completion of T4, the processor receives the *LBDone* signal and the processor latches in the value and continues the MODIFY and WRITE portion of the cycle. If the rest of the cycle can be completed by the time the hardware can decrement a credit for a VC at the next T2, then the data consistency can be maintained.

This implies that the MODIFY and WRITE portion of the cycle needs to be completed in 11 slot times or 440 ns. To examine if this solution was viable, test were performed to find the average time take to perform a READ-MODIFY-WRITE cycle. The average time was 1.8 μ s and is more than the 440ns specified. Since this test could not validate the 440ns limit, we decided to measure the exact time taken in hardware using a logic analyzer to determine the timing in hardware. From this test, it was found that the time taken in to perform the READ-MODIFY-WRITE in hardware was 3100ns. This value is about one order of magnitude higher than the 440ns that exist between the T4 and the next T2 and hence this solution is not viable.

4.5 Limitation of Implementation

This implementation of the algorithm does have certain limitations. Some of the limitations of this implementation occur due to the nature of the system being

Timing for the i/m BandwidthControl Algorithm

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
Latch VCI	Read ENB/CREDIT from PaceRAM	IF ENB = 0 wait for next cell ELSE DEC(CREDIT) Endif	No OP	No OP	Write back New CREDIT	IF CREDIT = 1 Then issue Stop VCI end if	Read M-value for the VC into variable CurM	CurM <- C + CurM MOD 4096	No OP	Buffer CurM for use by next Process		
Read TAIL pointer using CurM as address into Schedule Pointers	No OP	IF TAIL = NIL then Write CurM/HEAD <- FREE end if	Write CurM/TAIL <- FREE in SchPtr	No OP	No OP	FREE <- FREE.NEXT	Read CurM.TAIL from SchPtr	Write TAIL.Next <- NIL Write TAIL.VCI <- VCI in SchList				
Read C	Read C.HEAD from SchPtrs	Read HEAD.VCI from SchList	Read VCI/CREDIT from PaceRAM	INC(CREDIT)	IF CREDIT > 03hex then issue START_VC	Write VCI/CREDIT <- CREDIT in PaceRAM	Read HEAD.Next from SchPtrs Variable OFree <- FREE	IF HEAD.NEXT = NIL Then INC(C) end IF FREE <- HEAD	HEAD.Next <- OFree	HEAD <- HEAD.NEXT	Write HEAD in SchPtrs	

Figure 4-25: Time Usage of the i-out-of-m controller

used. One such limitation, is the lack of a floating point co-processor or a floating point emulator. Due to this, there was a need to convert all floating point values to integers and perform integer operation on them hence reducing the resolution of operation.

The 11 bit value of i and 12 bit value for m also limit the maximum burst size that can be assigned to a VC. The maximum number of burst that a VC can send in this implementation is 1024 cell (50kbps). This limitation can be eliminated by using all 16 bits in the i and m tables and using a more advanced FPGA that can has resources to process and use all 16 bits.

The lack of a minimum rate for this algorithm can lead to a VC being starved forever. This may occur if that VC has a large silence interval and the assigned rate goes to zero. This VC will never be able to transmit since it would have zero credits assigned to it. To avoid starving a VC, the customer may be required to specify a minimum cell rate (MCR) that needs to be maintained during the lifetime of the connections. Such a parameter will need to be negotiated at call set up time. The ATM forum is discussing such a parameter as an optional parameter in the traffic contract that the customer must provide. If the customer fails to provide such a parameter in the traffic contract, the network provider might uses the average rate or some percentage of the average rate to be the MCR. The MCR assignment will eliminate the starvation problem that might otherwise occur.

Chapter 5

Conclusion

5.1 Summary and Conclusions

This thesis has presented an implementation of an adaptive shaper that can be controlled in real-time based on a customer's traffic behavior. The practical constraints placed by the real-time system along with the theoretical results of the previous work were used to provide more realistic estimates of possible measurement intervals for the adaptive shaper. The algorithm's average delay performance was monitored to observe the effect of larger measurement intervals. The results from these simulations showed that when sources have a high variance from the average rate, the larger measurement intervals smooth out the variance of the source from the average rate and capture only significant changes to the rate. This smoothing effect is desirable for the rate adaptation since it helps capture the long-term changes that might not be detected by the small measurement interval since the shaper does not perform any long-term aggregation of rate over several measurement intervals.

In addition, this work provided results that verified that the adaptive shaper is effective in monitoring real data sources that are likely to exist in the network. In this work a large simulation was performed on data obtained from intra LAN

traffic at Bellcore.

By testing the *i*-out-of-*m* adaptive control scheme with different sources, it became apparent that the adaptive shaping algorithm can follow a customer's traffic and therefore can be more effective than a static shaper. The adaptive shapers provide lower average delays and better utilization than static shapers. The adaptation factors and measurement interval for the adaptive shaper, have been tuned to support different traffic types.

This work proved that a real-time implementation of such a shaper is possible and given the simplicity of the algorithm can co-exist with other control software. The implementation of such an adaptive shaper on the ATM/SONET gateway card is significant because of the sheer number of computations, hardware and software that interact with the shared memory (*i* and *m* tables). Maintaining data consistency was a challenge. The various solutions used to avoid some of the race conditions that could lead to data inconsistency are important and significant to a real-time implementation of such an adaptive shaper.

Finally, this work brings out the fact that there are significant divisions in the telecommunication community with respect to several significant issues. Switch vendors and network providers have yet to agree on the buffer capacities that are required and where they should reside. Switch vendors tend to keep the amount of buffers in the switch small while it is clear from this work that too small a buffer at the switch could lead to large cell losses if there is no source based buffering and hence poor performance to the end user. These sections of the telecommunication community need to cooperate in their efforts to provide the end user with quality ATM services that are attractive and useful to a customer.

5.2 Future Work

The present implementation of the shaper uses *constants* for the adaptation parameters. The adaptation parameters can also be adaptively varied as a factor of some of the negotiated parameters. Such adaptive parameters might allow the algorithm to provide finer grain control on the assigned rate based on the previously assigned rate, the average rate and the maximum rate.

Further, the present implementation does not specify a minimum rate that such a shaper must assign to any VC. To avoid starving a VC, the customer may be required to specify a minimum cell rate (MCR) that needs to be maintained during the lifetime of the connections. Such a parameter will need to be negotiated at call set up time. The MCR assignment will eliminate the starvation problem that might otherwise occur. An interesting study would be to adapt the parameters of the algorithm when a source reaches its MCR. This would involve setting up the parameters in such a way as that the algorithm allow the user to adapt back to their average rate within a few measurement intervals.

Other useful performance measurements of the adaptive shaper would be to use limited buffers or queues to estimate the cell loss ratio for various loss tolerant traffic types such as video and voice. This would provide another performance metric that can be used to validate this algorithm. Investigation is required into the interactions between the number of credits for free buffers in the LAN environment and the i and m values which need to be established in order to evaluate the buffer requirements at a switch. Such an interaction is likely to give benefit to this scheme.

The framework of communication (how and when) between the CPE and NPE entities needs to be further developed to determine the frequency at which the adaptive shaper can notify the adaptive policer of changes. The frequency at which this should happen is based on the associated signaling overheads and call set up times, and may well bound the measurement interval of the i -out-of- m

shaper. Such a framework could also be linked to a pricing model that makes ATM services cost effective and hence very attractive to a customer while it still profitable to a network provider.

Bibliography

- [1] D. Anderson, R. Herrtwich, and C. Schaefer. SRP: A resource reservation protocol for guaranteed performance communication. Technical Report TR-90-006, International Computer Sciences Institute, University of California - Berkeley, February 1990.
- [2] T. Anderson, C. Owicki, J. Sax, and C. Thacker. High speed switch scheduling for local area networks. In *Proc. Int. Conf. Arch. Supp. Prog. Lang. and Op. Sys.*, October 1992.
- [3] R. Ballart and Y. Ching. SONET: Now it's the standard optical network. *IEEE Communications Magazine*, 27(3):8-15, March 1989.
- [4] M. Bog. Credit statistics measurement for dynamic bandwidth management. TISL Technical Memo, TISL-9770-08, 1993.
- [5] C. Braun. Performance evaluation of dynamic and static bandwidth management methods for ATM networks. Master's thesis, University of Kansas, September 1994.
- [6] S. Chong, S. Li, and J. Ghosh. Dynamic bandwidth allocation for efficient transport of real time VBR video over ATM. In *Proc. IEEE INFOCOM*, pages 1c.2.1-1c.2.10, 1994.
- [7] I. Cidon, I. Gopal, and R. Guerin. Bandwidth management and congestion control in plaNET. *IEEE Communications Magazine*, 1991.

- [8] Digital Equipment Corporation. *DECelz: Programming Guide*, December 1992.
- [9] B. Ewy, V.Sirkay, and B.Buchanan. SRAM memory map. TISL Technical Memo, October 1994.
- [10] T. Faber, L. Landweber, and A. Mukherjee. Dynamic time windows: Packet admission control with feedback. In *Proc. ACM SIGCOMM*, pages 124–135, 1992.
- [11] ATM Forum. ATM User-Network Interface Specification, Version 2.0, June 1992.
- [12] ATM Forum. ATM User-Network Interface Specification, Version 3.0, June 1993.
- [13] Y. Gong and I. Akyildiz. Dynamic traffic control using feedback and traffic prediction in ATM networks. In *Proc. IEEE INFOCOM*, pages 1c.3.1–1c.3.7, 1994.
- [14] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent capacity-Its applications to bandwidth allocations in high speed networks. *IEEE J. Select. Areas Commun.*, 9(7):968–981, September 1991.
- [15] L. Gun and T. Tedijanto. Dynamic bandwidth estimation and allocation in high speed networks. IBM technical report, IBM-RTP, 1993.
- [16] H. Lee and J. Mark. Capacity allocation in statistical multiplexing of ATM sources. In *Proc. IEEE INFOCOM*, pages 6a.2.1–6a.2.8, 1994.
- [17] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic. In *Proc. ACM SIGCOMM*, pages 183–193, 1993.
- [18] LSI Logic. *LR33000 User's Manual*, 1991.

- [19] G. Minden, J. Evans, D. Petr, and V. Frost. An ATM WAN/LAN gateway architecture. In *2nd IEEE Symp. High Perf. Dist. Comp.*, pages 136–143, July 1993.
- [20] S. Minzer. Broadband ISDN and asynchronous transfer mode (ATM). *IEEE Communications Magazine*, 27(9), September 1989.
- [21] C. Partridge. *Gigabit Networking*. Addison-Wesley, 1993.
- [22] M. Prycker. *Asynchronous Transfer Mode: Solution for B-ISDN*. Ellis Horwood, 1991.
- [23] S. Seetharam. Description of SONET OC-12c and quad OC-3c mode of the SONET BIP/ Scrambler implemented in a Xilinx 3195 FPGA. TISL Technical Report TISL-9770-20, Telecommunication and Information Science Laboratory, University of Kansas, 1994.
- [24] S. Seetharam. Description of SONET OC-12c and quad OC-3c mode of the SONET Descrambler implemented in a Xilinx 3195 FPGA. TISL Technical Report TISL-9770-21, Telecommunication and Information Science Laboratory, University of Kansas, 1994.
- [25] S. Seetharam. Description of SONET OC-12c and quad OC-3c mode of the SONET Path Termination implemented in a Xilinx 3195 FPGA. TISL Technical Report TISL-9770-22, Telecommunication and Information Science Laboratory, University of Kansas, 1994.
- [26] S. Seetharam. Description of SONET OC-12c mode of the Transpose chip implemented in a Xilinx 3195 FPGA. TISL Technical Report TISL-9770-18, Telecommunication and Information Science Laboratory, University of Kansas, 1994.

- [27] S. Seetharam. Description of transmit and receive SONET overhead SRAMS in the KU MAGIC ATM/SONET Gateway. TISL Technical Report TISL-9770-19, Telecommunication and Information Science Laboratory, University of Kansas, 1994.
- [28] V. Sirkay, D. Niehaus, and B. Buchanan. Solution to concurrent access of data in i and m tables by the line card processor and i-out-of-m controller. TISL Technical Report TISL-9770-25, Telecommunication and Information Science Laboratory, University of Kansas, 1994.
- [29] W. Stallings. *ISDN and Broadband ISDN*. Macmillan Publishing Company, 1992.
- [30] Bellcore Technical Reference TR-NWT-000253. Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria, Dec 1991.
- [31] J. Turner. New directions in communications (or which the information age?). *IEEE Communications Magazine*, 24(10):8–15, October 1986.
- [32] J. Turner. Design of a broadcast packet switching network. *IEEE Trans. Comm.*, COM-36(6):734–743, June 1988.
- [33] H. Uriona. Description of FIFO controller for the AN2/SONET gateway card. TISL Technical Report TISL-9770-23, Telecommunication and Information Science Laboratory, University of Kansas, 1994.
- [34] H. Uriona, S. Seetharam, and V. Sirkay. Description of the i-out-of-m controller - Design 1 implemented in Xilinx 3195 FPGA. TISL Technical Report TISL-9770-26, Telecommunication and Information Science Laboratory, University of Kansas, 1994.
- [35] S. Wang. A novel rate-based flow control and buffer management architecture for ATM communication networks. In *Proc. IC3N*, pages 54–62, 1993.

- [36] E. Yousefi. Description of the Pacing Measurement - Design 1 implemented in Xilinx 3195 FPGA. TISL Technical Report TISL-9770-17, Telecommunication and Information Science Laboratory, University of Kansas, 1994.

Appendix A

SONET Overhead

- A1 - Framing
- A2 - Framing
- C1 - STS-1 ID
- J1 - STS Path Trace
- B1 - BIP - 8
- E1 - OrderWire
- F1 - Section User Channel
- B3 - BIP-8
- D1 - Section Data Com Channel
- D2 - Section Data Com Channel
- D3 - Section Data Com Channel
- C2 - STS Path Signal Label
- H1 - Pointers

A1 A1 A1 A1 A1 A1 A1 A1	A2 A2 A2 A2 A2 A2 A2 A2	C1 C1 C1 C1 C1 C1 C1 C1	J1
F6 F6 F6 F6 F6 F6 F6 F6	28 28 28 28 28 28 28 28	01 01 01 01 01 01 01 01	01 01 01 01 01 01 01 01
B1	E1	F1	B3
H	00	00	H
D1	D2	D3	C2
00	00	00	00
H1 H1 H1 H1 H1 H1 H1 H1	H2 H2 H2 H2 H2 H2 H2 H2	H3 H3 H3 H3 H3 H3 H3 H3	G1
62 93 93 93 93 93 93 93	0A FF FF FF FF FF FF FF	00 00 00 00 00 00 00 00	L
B2 B2 B2 B2 B2 B2 B2 B2	K1	K2	F2
H H H H H H H H	00	L	00
D4	D5	D6	H4
00	00	00	00
D7	D8	D9	Z3
00	00	00	00
D10	D11	D12	Z4
00	00	00	00
Z1 Z1 Z1 Z1 Z1 Z1 Z1 Z1	Z2 Z2 Z2 Z2 Z2 Z2 Z2 Z2	E2	Z5
00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00

Figure A-2: SONET OC-12 Frame With Values

- H2 - Pointers
- H3 - Pointer Action Byte
- G1 - Path Status
- B2 - BIP -8
- K1 - APS Channel
- K2 - APS Channel
- F2 - Path User Channel
- D4 - Line Data Com Channel
- D5 - Line Data Com Channel
- D6 - Line Data Com Channel
- H4 - Indicator Byte
- D7 - Line Data Com Channel
- D8 - Line Data Com Channel
- D9 - Line Data Com Channel
- Z3 - Growth
- D10 - Line Data Com Channel
- D11 - Line Data Com Channel
- D12 - Line Data Com Channel
- Z4 - Growth
- Z1 - Growth

- Z2 - Growth
- E2 - OrderWire
- Z5 - Growth

Appendix B

GUI

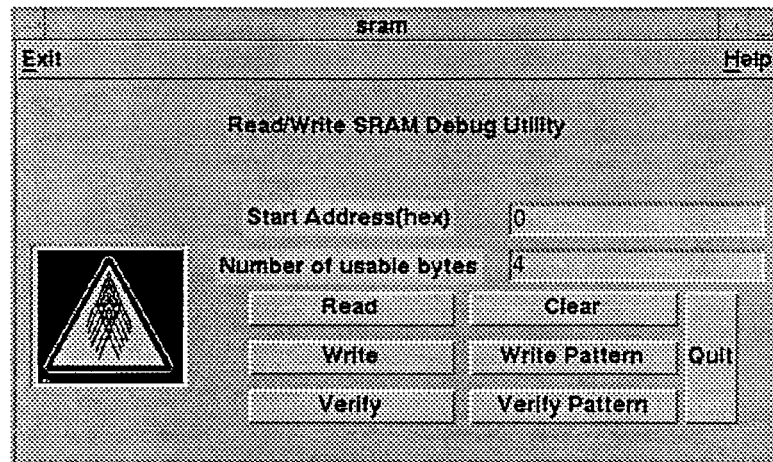


Figure B-1: SRAM Debug GUI

File Help

The frame below is 9 rows by 36 bytes of SONET overhead.
 Line and Section overhead is displayed in columns 1 to 36.
 Path information is displayed in columns 37, 38, 39, and 40.

Q12C 0 1 2 3 4 5 6 7 xmit set:
 OC12 6 9 10 11 12 13 14 15 frame

The GUI displays a 9x36 grid of overhead bytes. The first 36 columns are organized into 9 rows and 4 columns of 9 bytes each. The first two columns of each row contain hexadecimal values (e.g., 01, 02, 03, 04, 05, 06, 07, 08, 09). The next two columns contain alphanumeric labels (e.g., F1, F2, F3, F4, F5, F6, F7, F8, F9). The final two columns contain alphanumeric labels (e.g., B1, B2, B3, B4, B5, B6, B7, B8, B9). The last four columns (37-40) show path information with alphanumeric labels (e.g., M1, M2, M3, M4, M5, M6, M7, M8, M9) and binary values (0, 1).

Read Write Clear

Figure B-2: SONET Overhead GUI

File Host

Source Destination List

<input type="checkbox"/>	nauchly	punx
<input type="checkbox"/>	hopper	punx
<input type="checkbox"/>	stibitz	punx
<input type="checkbox"/>	nauchly	iss-1.edc
<input type="checkbox"/>	hopper	iss-1.edc
<input type="checkbox"/>	stibitz	iss-1.edc
<input type="checkbox"/>	nauchly	nerlin-atn
<input checked="" type="checkbox"/>	hopper	nerlin-atn

Parameters from input to switch		Parameters from switch to output	
Port	<input type="text" value="6"/>	Port	<input type="text" value="6"/>
Head	<input type="text" value="1"/>	Head	<input type="text" value="2"/>
VCI	<input type="text" value="130"/>	VCI	<input type="text" value="130"/>
Flow Control		Flow Control	
<input type="checkbox"/> Input FlowMaster		<input type="checkbox"/> Output FlowMaster	

Forward Rate

CBR Forward Rate (Kb/s)

Duplex Connection

Reverse Rate

CBR Reverse Rate (Kb/s)

Figure B-3: PVC Setup GUI

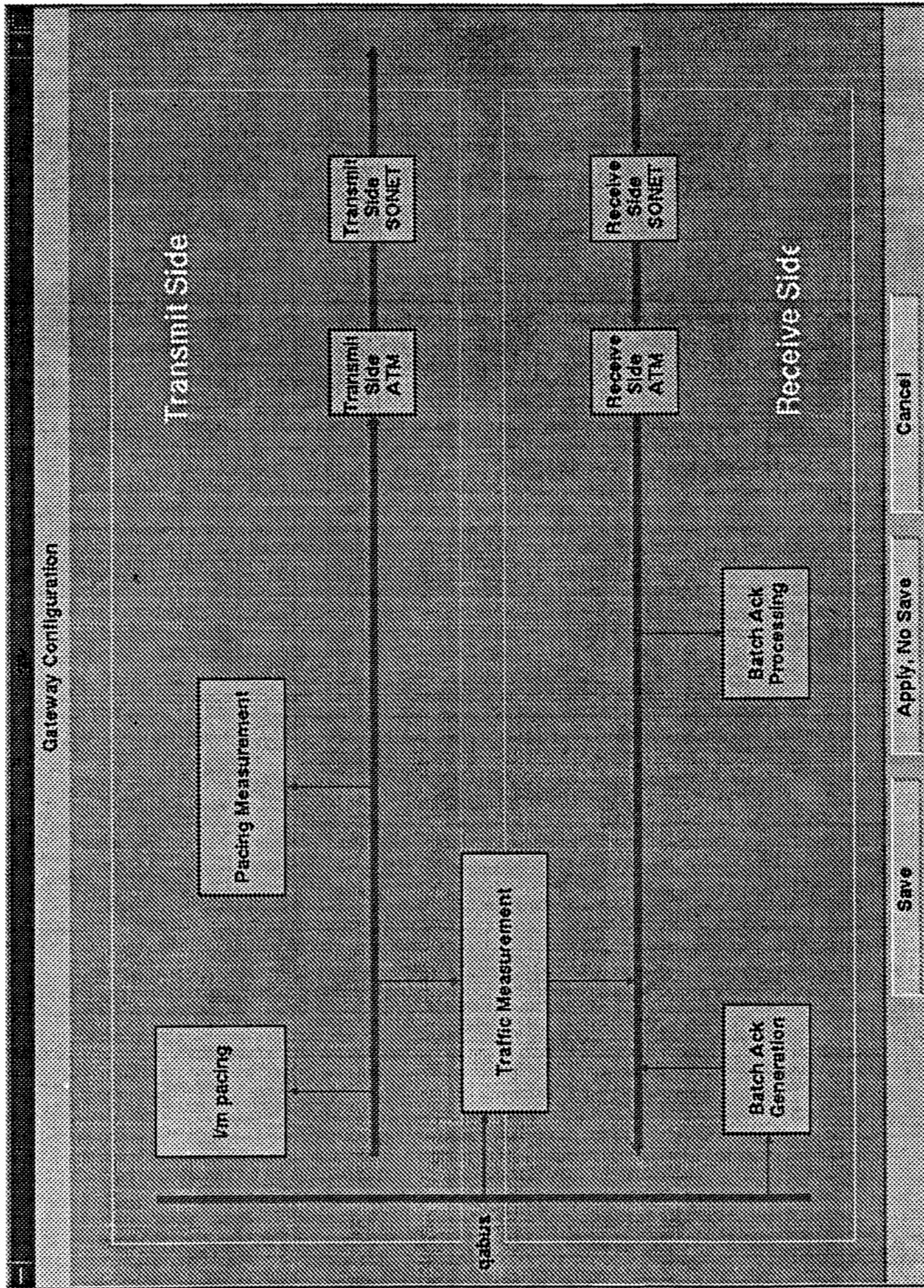


Figure B-4: Gateway Configuration GUI