EVALUATING REAL-TIME SCHEDULING TECHNIQUES FOR MULTIMEDIA SERVICES OVER ATM

by

RAYMOND SIU-LUN LAM

B.A.Sc., The University of British Columbia, 1996

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

October, 2000

© Raymond Siu-Lun Lam, 2000
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical and Computer Engineering
The University of British Columbia
Vancouver, Canada

Date: Oct 3rd, 2000
Abstract

In an ATM network, a single link may carry traffic for thousands of connections with different parameters and quality-of-service requirements. High-speed links, coupled with small cell size, require efficient schedulers that can handle cell arrivals and departures quickly. The design goals for bandwidth schedulers in ATM switches are presented, one crucial goal is the ability to guarantee service bandwidth to individual queues. Basically, scheduling algorithms can be divided into rate-based and schedule-based. Rate-based methods use a predefined set of allowable rates to allocate bandwidth to a connection, while schedule-based methods analyze the potential interactions between packets of different connections, and determine if there is any possibility of a deadline being missed. This thesis investigates the performances of various types of scheduling algorithms. In addition, we implement and simulate a rate-based algorithm called Self-Clocked Fair Queuing. We also investigate the feasibility of implementing such an algorithm. To determine the performance, we mainly look at two parameters: delay bound and jitter-delay bound. Several cases of traffic patterns are used in the simulations. Through the simulations, a bandwidth-coupling problem of SCFQ is identified. This bandwidth-coupling problem in SCFQ occurs because low bandwidth connections suffer a long delay and jitter-delay. To solve this, we propose a SCFQ hybrid algorithm called JEDD-SCFQ. Not only does this algorithm handles low-bandwidth connection fairly, it also provides nominal delay guarantees to all other connections.
# Table of Contents

ABSTRACT ii

TABLE OF CONTENTS iii

LIST OF FIGURES vi

LIST OF TABLES viii

CHAPTER 1 INTRODUCTION .................................................................................................................. 1

1.1 CONCEPTS OF ATM TECHNOLOGY ....................................................................................... 2
1.2 ATM REFERENCE MODEL ......................................................................................................... 4
1.3 ATM TRAFFIC CATEGORIES .................................................................................................... 6
1.4 SCHEDULING IN ATM NETWORK .......................................................................................... 8
1.5 THESIS CONTRIBUTIONS ....................................................................................................... 10
1.5 THESIS OUTLINE .................................................................................................................... 11

CHAPTER 2 SCHEDULING REAL-TIME TRAFFIC IN ATM NETWORKS ............................... 12

2.1 REAL-TIME COMMUNICATIONS OVER ATM NETWORK ...................................................... 12
2.2 QUALITY OF SERVICE PARAMETERS .................................................................................... 14
2.3 MAPPING OF REAL-TIME APPLICATIONS TO ATM TRAFFIC CATEGORIES ......................... 14
   2.3.1 Closed Loop Control ......................................................................................................... 15
   2.3.2 Open Loop Control ......................................................................................................... 16
   2.3.3 Traffic Model of CBR and rt-VBR .................................................................................. 16

CHAPTER 3 SCHEDULING ALGORITHMS FOR ATM TRAFFIC ........................................... 18

3.1 ATM QUEUEING SCHEMES .................................................................................................... 18
3.1.1 Input Queuing ................................................................. 19
3.1.2 Central Queuing ......................................................... 20
3.1.3 Output Queuing .......................................................... 21

3.2 Design Issues of Scheduling Algorithms ................................ 23
3.2.1 Priority Levels ............................................................ 24
3.2.2 Work-Conserving vs Non-Work-conserving .................... 26
3.2.3 Rate-Based vs Schedule-Based ....................................... 28
3.2.3 Frame-Based vs Sorted-Priority ....................................... 28
3.2.5 Summary ................................................................. 29

3.3 Rate-Based Scheduling Algorithms for ATM ......................... 30
3.3.1 Generalized Processor Sharing (GPS) .............................. 30
3.3.2 Weighted Fair Queuing (WFQ)/Packetized Generalized Processor Sharing (PGPS) ................................................. 34
3.3.2.1 Theory of WFQ and GPGS ........................................... 34
3.3.2 Self-Clocked Fair Queuing (SCFQ) .................................. 39

3.4 Schedule-Based Scheduling Algorithms for ATM .................... 42
3.4.1 Delay Earilness Due Date (D-EDD) ................................. 43
3.4.2 Jitter Earliest Due Date (J-EDD) ..................................... 44

3.5 Performance and Implementation Comparison ....................... 45
3.5.1 Quality-of-Service Performance Bound ............................ 46
3.5.2 Implementation Complexity .......................................... 50
3.5.4 Timestamp Calculation .................................................. 51
3.5.5 Memory Storage ....................................................... 52
3.5.6 Implementation of Jitter-Delay Regulator ......................... 52

CHAPTER 4 SIMULATION MODEL ................................................. 54

4.1 Input Traffic Model .......................................................... 54
4.1.1 Applications of rt-VBR Traffic ....................................... 54
4.1.2 Traffic Generators ..................................................... 55

4.2 Topologies of Output Queue Management Model ................... 58
List of Figures

Figure 1.1 ATM 3-D reference model................................................................. 4
Figure 3.1 Input Queuing .................................................................................. 19
Figure 3.2 Central queuing ............................................................................... 21
Figure 3.3 Output Queuing ............................................................................... 22
Figure 3.4 Knockout switch ............................................................................ 22
Figure 3.5 Arrival pattern and GPS service order .......................................... 33
Figure 3.6 Arrival pattern and WFQ/PGPS service order ................................. 36
Figure 3.7 An example of virtual time ν(t) in PGPS ....................................... 39
Figure 3.8 Functional Diagram of SCFQ: ....................................................... 40
Figure 3.9 Arrival pattern and WFQ/PGPS service order ................................. 41
Figure 4.1 Traffic shaper in ATM networks ..................................................... 56
Figure 4.2 Simple Block Diagram of ATM Switch ........................................... 59
Figure 4.3 Block Diagram of Dual-port Memory ............................................. 61
Figure 4.4 Timing Diagram of memory block ................................................ 61
Figure 4.4 Worst Case for Routing ................................................................. 62
Figure 4.5 Architecture of Output Queue Manager ....................................... 64
Figure 4.6 Block Diagram of Scheduler .......................................................... 66
Figure 5.1 Network Topology ........................................................................ 74
Figure 5.2 Service Pattern of JEDD-SCFQ Connections ............................... 79
Figure 5.3 Delay vs Maximum Burst Size for VC1 ........................................ 81
Figure 5.4 Maximum and Average Delay vs Bandwidth in SCFQ for VC1 .................. 82
Figure 5.5 Maximum Delay Comparison between SCFQ and DEDD-SCFQ for VC1 .......... 84
Figure 5.6 Maximum Delay vs Bandwidth of Delay-Bounded VC1 .......................... 87
Figure 5.7 Maximum Delay vs Bandwidth - delay bounded VC and non-delay bounded VC .......................... 89
Figure 5.8 Maximum Delay vs Bandwidth – non-delay bounded VC .......................... 91
List of Tables

Table 1.1 SONET Hierarchy ................................................................. 2
Table 1.2 Summary of the characteristics of ATM service category ........................................... 8
Table 2.1 Service Classes and Applicable Parameters ............................................................... 15
Table 3.1 Characteristics of Various Scheduling Algorithms ...................................................... 30
Table 5.1 Traffic and Scheduler Parameters ........................................................................... 75
Table 5.2 Bandwidth Allocation in Scenario 1 ........................................................................... 80
Table 5.3 Bandwidth Allocation for Scenario 2 ......................................................................... 82
Table 5.4 Bandwidth Allocation for Scenario 3 ......................................................................... 83
Table 5.5 Bandwidth allocation for Scenario 4 .......................................................................... 85
Table 5.6 Bandwidth Allocation for Scenario 5 ......................................................................... 88
Table 5.7 Bandwidth Allocation for the Two Trails in Scenario 6 ............................................... 90
Chapter 1 Introduction

Asynchronous Transfer Mode (ATM) has been proposed by the International Telecommunication Union-Telecommunication Sector (ITU-T) as the solution for the future broadband telecommunication network [1]. The most valuable advantage of ATM is its ability to carry different types of information, such as voice, video, and data over the same infrastructure while providing the required Quality of Service (QoS) for each type of media. Many future applications of computer networks will rely on the ability of the network to provide QoS. These QoS guarantees can be provided by bounding one or more of the QoS parameters, such as the end-to-end delay, bandwidth, delay jitter and cell loss rate. For example, real-time traffic requires small end-to-end delay and jitter, while data transfer requires small amount of cell loss. Thus, it is essential for an ATM switching node to use an efficient scheduling algorithm, so that these QoS parameters can be bounded.

Basically, an ATM network is formed by a number of ATM nodes, which are interconnected by high-speed physical links. A digital carrier is used to transfer information across these links. Although ATM is associated closely with Synchronous Optical Network (SONET) as the carrier, there is not any restriction to use other carrier systems, such as T1, E1, T3, E3 or even over a serial interface. However, due to SONET’s very high speed rate (up to 4.8 gigabit per second) and its’ vast capabilities of service rate flexibility, SONET becomes the logical choice for ATM [2]. Table 1.1 shows the different bit rate supported by SONET. Today’s SONET standards contain definitions for fiber-to-fiber interfaces at the physical level. They determine the optical line rate, wavelength, power levels, pulse shapes, and coding. Current
standards also fully define the frame structure, overhead, and payload mappings. An ATM network is bandwidth-transparent, which allows handling of a dynamically variable mixture of services at different bandwidths. ATM also easily accommodates traffic of variable speeds. As a result, SONET becomes very suitable for ATM.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Bit Rate</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>STS-1, OC-1</td>
<td>51.840 Mb/s</td>
<td>28 DS1s or 1 DS3</td>
</tr>
<tr>
<td>STS-3, OC-3</td>
<td>155.520 Mb/s</td>
<td>84 DS1s or 3 DS3s</td>
</tr>
<tr>
<td>STS-12, OC-12</td>
<td>622.080 Mb/s</td>
<td>336 DS1s or 12 DS3s</td>
</tr>
<tr>
<td>STS-48, OC-48</td>
<td>2488.820 Mb/s</td>
<td>1344 DS1s or 48 DS3s</td>
</tr>
<tr>
<td>STS-192, OC-192</td>
<td>9953.280 Mb/s</td>
<td>5376 DS1s or 192 DS3s</td>
</tr>
</tbody>
</table>

STS = Synchronous Transport Signal  
OC = Optical Carrier

Table 1.1 SONET Hierarchy [2]

In ATM networks, where data is transmitted in small fixed size cell, the scheduling algorithm is usually implemented in hardware to cope with the high speed requirements of ATM links. Usually, the scheduler is located at the output of an ATM switch. It receives multiple cells and transmits one cell in each time slot. Cells waiting for transmission are stored in queues. It is the scheduler’s responsibility to select which of these waiting queues is to be served in each time slot.

The following sections of this chapter will provide background information about ATM networks and the importance of scheduling algorithms.

### 1.1 Concepts of ATM Technology

Since the evolution of broadband integrated data networks (B-ISDN) as a universal network, a wide range of audio, video and data services can be supported in the same network.
The implementation of B-ISDN is based on a service-independent transfer technique called Asynchronous Transfer Mode (ATM). ATM has been proposed by the International Telecommunication Union-Telecommunication Sector (ITU-T) as the solution for the future broadband telecommunication network [3]. ATM is a virtual circuit oriented packet-switching technology that simplifies information transfer and exchanges by compartmentalize information into uniform segments called cells.

Each ATM cell has a size of 53 bytes, in which 5 bytes of a cell is the header and 48 bytes is the payload. The header field carries information that pertains to the ATM layer functionality itself, mainly the identification of cells. Inside the header, the virtual path identifier (VPI) and virtual circuit identifier (VCI) are responsible for providing the appropriate routing information.

ATM is a connection-oriented network, a connection between two end points begins when one transmits a signaling request across the network. If the network has sufficient resources for the requested call, a virtual channel is setup across the ATM network. If it does not have enough resources, the call request is declined. Quality of service is negotiated at connection setup to make sure that the network can provide the QoS parameters requested by the source. The most important advantage of ATM is its ability to carry voice, video, and data over the same infrastructure while providing QoS guarantees in an efficient and simple way [1][4]. However, in order for the network to provide QoS, the source sending cells through the channel must conform to a traffic shaping. This contract requires the sender to obey some statistical parameters such as peak bandwidth, average sustained bandwidth, burst size and other information which are provided during the connection setup.
1.2 ATM Reference Model

The three dimensional B-ISDN protocol reference model is shown in Figure 1.1 [4]. It consists of three planes: the user plane which is responsible for the transfer of user information, the control plane which is in charge of connection control functions while the management plane performs network supervision functions. The user plane and the control plane are further divided into three main layers: the physical layer, the ATM layer and the ATM Adaptation Layer (AAL). The physical layer deals directly with the physical medium. It converts the ATM cell stream into bits to be transmitted over the physical medium and also converts bits into ATM cells to be processed by higher layers. It also deals with issues related to the physical medium, such as voltages and bit timing.
The ATM layer deals with cell and their transportation. There are four main functions of this layer, which includes generic flow control (GFC), cell header generation/extraction, cell VPI/VCI translation and cell multiplex/demultiplex. The GFC supports the control of ATM traffic flow in a network. It is used to alleviate congestion in the network. In the transmit direction, the cell header generation adds the appropriate ATM cell header to the cell except the HEC. In the opposite direction, the cell header extraction function removes the cell header. Only the payload field is passed to the AAL. At each switching node, the VPI/VCI translation has to be performed. At a VP (Virtual Path) switch, the VPI field of each incoming cell is translated into a new VPI value for the outgoing cell. At a VC (Virtual Circuit) switch, the values of the VPI as well as the VCI are translated into new values. For cell multiplexing at the transmit direction, cell from individual VPs and VCs are multiplexed into one merged cell stream. At the receiving side, the cell demultiplexing function splits the arriving cell stream into individual cell flows appropriate to the VP or VC.

The ATM Adaptation Layer (AAL) is divided into two sublayers, the Segmentation and Reassembly (SAR) Sublayer and the Convergence Sublayer (CS). The essential functions of the SAR sublayer are, at the transmitting side, the segmentation of higher layer packets into ATM cells and, at the receiving side, the reassembly of ATM cells into higher layer packets. The CS provides the interface between the ATM network and the upper layers by performing different application-dependent functions.

The main focus of this thesis is the design of hybrid schedulers in ATM switch buffers. Simply, a scheduler manages the order of cell transmission. A scheduler has to provide the
necessary quality-of-service requirements to the user. As a result, this thesis only concerns with the transmission of user data flowing through the User Plane. Specifically, the scheduler functions in the ATM layer, which does not involve the protocol issues of the ATM Adaptation Layers (AALs) and other higher layers.

1.3 ATM Traffic Categories

The enforcement of Quality-of-Service (QoS) through traffic management relies on the definition of service classes. The ATM Forum currently defines five service categories, which cover the spectrum of potential applications using ATM. Different categories have different QoS parameters. Parameters such as cell delay, cell loss, and cell delay variation [5] are measured to ensure that the traffic behavior conforms to the QoS characteristics of each class. In the following, we briefly introduce each category:

Constant Bit Rate (CBR) Service

The CBR service supports real-time applications such as voice and video conferencing and telephone connections. The traffic is deterministic and requires strict cell delay and cell delay variation requirements. It can be characterized by a constant cell arrival rate, which corresponds to a fixed peak emission rate.

Real-Time Variable Bit Rate (rt-VBR) Service

The rt-VBR service also supports real-time applications, such as interactive compressed video, which have known or predictable bursty traffic characteristics. rt-VBR requires a tightly
constrained delay and delay variation. The bandwidth requirements vary with time, which can be modeled by an on/off source.

Non-Real-Time Variable Bit Rate (nrt-VBR) Service

The nrt-VBR service supports non-real-time applications that also have bursty characteristics and require variable bandwidths. nrt-VBR traffic is not constrained by delay bound, but it does expect a low cell loss ratio. Application such as Multimedia email is an example of nrt-VBR service.

Unspecified Bit Rate (UBR) Service

The UBR service offers no QoS guarantee and does not require prior knowledge of traffic characteristics. UBR applications do not have any bandwidth guarantee. It is ideally suited to low-cost and low-priority services, such as email or file transfer. As opposed to CBR and VBR applications, UBR applications are not sensitive to cell loss.

Available Bit Rate (ABR) Service

Like UBR, ABR service also supports data applications such as file transfer and email, but with the difference that ABR applications respond to congestion feedback messages from the network [6]. ABR provides a guaranteed QoS concerning fair bandwidth allocation and cell loss. However, it does not guarantee delay nor delay variation.
<table>
<thead>
<tr>
<th>Service characteristics</th>
<th>CBR</th>
<th>rt-VBR</th>
<th>nrt-VBR</th>
<th>ABR</th>
<th>UBR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth guarantee</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>optional</td>
<td>N</td>
</tr>
<tr>
<td>Suitable for R/T traffic</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Bursty Traffic</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Feedback for congestion</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

Table 1.2 Summary of the characteristics of ATM service category [7]

Since this thesis concerns about scheduling real-time traffic, the focus will be put in rt-VBR services. Detailed QoS parameters for rt-VBR traffic will be discussed in later chapters.

### 1.4 Scheduling in ATM Network

The dramatically increased bandwidths and processing capabilities of high-speed ATM networks make possible many distributed real-time applications, such as multimedia conferencing, shared workspace and distributed interactive simulations, that are coexisted in an integrated network. Different types of traffic have significantly different quality-of-service (QoS) requirements. Real-time applications usually require a guaranteed portion of link bandwidth, as well as bounded end-to-end delay and low delay jitter. Traditional best of effort servicing schemes cannot provide the QoS which real-time applications required. To provide the guaranteed QoS, an efficient service discipline must be employed. In order to achieve this, a scheduler is used at each switch (or router) of the network. The function of a scheduler is to select for each outgoing link of the switch (or router) the packet to be transmitted in the next cycle from the available packets belonging to the flows sharing the output link. Different
scheduling techniques give various performances for the ATM switches. Implementation of these algorithms may be in hardware or software. Since ATM cell has a small cell size, the scheduling algorithm is usually implemented in hardware to achieve the fast switching characteristics of ATM switches.

Broadband packet networks based on ATM are currently enabling the integration of traffic with a wide range of characteristics, from real-time video traffic with stringent QoS requirements to “best-effort” data requiring no guarantees, within a single network. “Best-effort” algorithms, such as Round Robin and First-Come-First-Serve, are not suitable for our ATM network because they cannot provide the QoS guarantees that are required. As a result, different scheduling algorithms that can provide QoS guarantees have been proposed, for example, Generalized Processor Sharing, Weighted Fair Queuing, Self-Clocked Fair Queuing, Hierarchical Round Robin and Earliest Deadline Due. These algorithms differ in many ways, such as degree of QoS guarantees, implementation complexity, traffic versatility, etc.

In order to provide the required QoS, proper scheduling algorithms have to be used. In addition, buffer management is also important to the performance of a scheduling algorithm. When a scheduler queues up the cells, it needs to access these queues efficiently. There are different queuing techniques, such as input queuing, output queuing, shared queuing or hybrid of above.

Currently, the ATM Forum does not have a specification on scheduling algorithms. It is up to the ATM switch vendors to decide which algorithm to use. However, the range of choices
available in commercial switches is very limited [8]. Most schedulers are first-come-first-served, with a single level of priority. Some switches and routers allow multiple levels of priority, but even this is rarely used. The sale of switches and routers that provide more interesting alternatives, such as hierarchical round-robin and weighted fair queuing [8], is just beginning. Currently, there is no commercial switches providing non-work-conserving algorithm.

1.5 Thesis Contributions

The main contribution of this thesis is a hybrid scheduling technique that combines weighted fair queuing with deadline based scheduling. The purpose is to improve system response for lower bit rate flows. The thesis also reports the performance of a complete VHDL realization of the scheduling system. In order to achieve this, the thesis will:

1. Provide a characterization of queuing schemes and scheduling algorithms. Inside an ATM switch, cells waiting to be served are queued up. Different queuing techniques will affect the performance of the scheduler. Hence, queuing schemes have to be considered.

2. Analyze the implementation complexity of various scheduling algorithms. Since ATM networks have a very high speed. It is essential to have a simple and efficient scheduling algorithm in an ATM switch.

3. Identify the coupling problem between bandwidth and delay in fair-queue based algorithms. This coupling problem is especially important for real-time traffic, as it usually requires a
tight delay, but not a high bandwidth. It would waste the network bandwidth if a connection reserves a high bandwidth only to satisfy its delay bound.

4. Develop a variation of fair queuing scheme that tries to minimize the coupling problem between bandwidth and delay. In addition, the implementation complexity of this algorithm will also be investigated.

5. Identify new directions of scheduling algorithms that will be used by the next stage ATM switches.

1.5 Thesis Outline

The thesis is organized as follows. Chapter 2 provides the background of real-time communications in ATM networks. It identifies the goals of real-time scheduling, the requirements of QoS and various QoS parameters in different ATM traffic categories. Chapter 3 introduces some common scheduling algorithms and offers comparisons of these algorithms. Chapter 4 describes the simulation model of this thesis as well as the implementation of the schedulers. Chapter 5 presents the simulation results of various scheduling algorithms. Finally, Chapter 6 concludes the work presented in this thesis and provides some suggestions for future work.
Chapter 2 Scheduling Real-Time Traffic in ATM Networks

Traditional communication network applications such as file transfer and electronic mail are examples of non-real-time applications. These data-oriented applications have strict reliability requirements. The performance metrics of these applications are typically cell loss rate and throughput. In contrast, the performance of real-time communication is measured not only by the loss rate and throughput, but also delay, delay jitter, and/or bandwidth guarantees. As a result, it is necessary to determine the queuing and scheduling algorithms that support the specification and control of QoS parameters for real-time applications.

This chapter will discuss the requirements to schedule real-time traffic. The following section will introduce some fundamental notions in real-time communications. Then the QoS parameters for real-time traffic will be specified. Traffic in an ATM network must be mapped into one of the ATM service categories. Chapter 2.3 will address the issue on mapping real-time traffic onto ATM service classes.

2.1 Real-Time Communications over ATM Network

The value of real-time communication depends upon the times at which packets are successfully delivered. Real-time communication applications are commonly classified as either soft or hard real-time applications. Soft real-time applications can tolerate certain amount of packet loss, while hard real-time applications have zero packet loss tolerance. Thus, hard real-
time applications require a stringent service and deterministic predictability of network delays [9]. Both hard and soft real-time applications are bounded by a maximum delay time, called deadline. If a packet arrives at the destination after the deadline has expired, its value to the application is greatly reduced. As a result, having a low delay would seem to be desirable. Other than delay, another important metrics to real-time applications is the delay-jitter. It is defined as the maximum variation in delay experienced by packets in a single flow [10]. Some real-time applications do not care how much prior to a deadline a packet arrives. On the other hands, packets arriving early would require the receiver to buffer them up, so that a constant end-to-end delay can be achieved. Consequently, a small delay jitter would minimize the memory overhead of a scheduler.

Real-time applications, such as interactive video conferencing, require both the delay bound and the jitter bound. Furthermore, non-interactive applications, such as video playback, require only a jitter delay bound, but no delay bounds. The following are some of the desirable properties for real-time communications:

- ability to easily integrate non-real-time and real-time services
- low delay jitter
- low delay
- modest buffer requirement
- high effective bandwidth utilization
- low overhead in header
- low processing overhead per packet
One of the most valuable advantages of ATM is its ability to integrate real-time and non-real-time applications over the same infrastructure while providing acceptable QoS [11]. ATM is currently the main technology used to carry real-time packetized traffic. The relatively small cell payload in ATM is well suited for demanding real-time applications. For example, the 48-byte payload size of an ATM cell allows real-time packet voice transport without the need for echo cancellation equipment on continental connections.

### 2.2 Quality of Service Parameters

For each application running over ATM, there is a specific set of service and QoS parameters that need to be guaranteed by the ATM network. These parameters are negotiated during the connection setup and must be agreed upon by both the user and the network. The ATM Forum [4] specifies a list of traffic and QoS parameters that the traffic has to follow. However, not all the parameters are applicable to each traffic category. Table 2.1 below summarizes the applicable parameters for each class.

### 2.3 Mapping of Real-Time Applications to ATM Traffic Categories

As was shown in Chapter 1.3, there are five ATM categories specified by the ATM Forum. A real-time system has to intelligently map the actual application requirements onto these services. To do this, of QoS requirements have to be identified. Traffic engineering is applied to meet QoS requirements of applications using these services. There are two types of traffic control - open loop and closed loop control.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>ATM Layer Service Categories</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Constant Bit Rate (CBR)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variable Bit Rate (VBR)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Real Time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variable Bit Rate (n-VBR)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Non-Real Time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Available Bit Rate (ABR)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unspecified Bit Rate (UBR)</td>
<td></td>
</tr>
<tr>
<td>Cell Loss Ratio (CLR)</td>
<td>Specified</td>
<td>QoS</td>
</tr>
<tr>
<td>Maximum cell transfer delay (CTD) and mean cell transfer delay (CDV)</td>
<td>CDV and max CTD</td>
<td>QoS</td>
</tr>
<tr>
<td></td>
<td>CDV and max CTD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean CTD only</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unspecified</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unspecified</td>
<td></td>
</tr>
<tr>
<td>Peak cell rate (PCR) and cell delay variation tolerance</td>
<td>Specified</td>
<td>Traffic</td>
</tr>
<tr>
<td></td>
<td>Specified</td>
<td></td>
</tr>
<tr>
<td>Sustainable cell rate (SCR) and burst tolerance (BT)</td>
<td>N/A</td>
<td>Traffic</td>
</tr>
<tr>
<td></td>
<td>Specified</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specified</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Minimum cell rate</td>
<td>N/A</td>
<td>Traffic</td>
</tr>
<tr>
<td></td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specified</td>
<td></td>
</tr>
<tr>
<td></td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Congestion control</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.1 Service Classes and Applicable Parameters

2.3.1 Closed Loop Control

Closed loop control is decided for the ABR service [12]. A closed loop feedback control mechanism is employed in which the network participates in informing the source how much traffic load is in the network. In this way, the traffic can avoid undesirable amount of cell losses, however, ABR service provides no guarantee on delay.
2.3.2 Open Loop Control

Open loop control is imposed for CBR, rt-VBR and nrt-VBR, where the QoS requirement with respect to delay bound is guaranteed only if the source does not exceed its specified sending rate. By mapping real-time traffic onto either CBR or rt-VBR services, performance guarantees can be obtained from the network with respect to delay bounds and loss ratio. Consequently, open-loop control is particularly suited for real-time communications.

2.3.3 Traffic Model of CBR and rt-VBR

As shown in table 2.1, PCR (peak cell rate), SCR (sustainable cell rate), BT (burst tolerance) are the traffic parameters used to characterize a traffic source, while CTD (maximum cell transfer delay) and CDV (mean cell delay variation) are end-to-end delay requirements. According to the ATM Forum’s Traffic Management Specification 4.0 [4], a CBR connection is characterized by its PCR. A rt-VBR connection is specified by SCR, MBS as well as the PCR. CBR is designed to support real-time communications that requires a constant amount of bandwidth and a tightly constrained CDV, while rt-VBR has a variable bandwidth (bursty traffic) and requires a less tightly constrained CDV. The QoS parameters for both CBR and rt-VBR include CTD and CDV.

The traffic model for CBR can be easily characterized by the PCR, that is 1/(peak bandwidth). For example, a MPEG video requires ~2Mbps bandwidth, then the PCR is equal to 212us/cell. On the other hand, the traffic model for rt-VBR can be interpreted as follow:
• $SCR$ is the long term average cell rate

• $\frac{1}{PCR}$ is the minimum inter-cell arrival time

• during any time interval of length $MBS \times \frac{1}{SCR}$, the maximum number of cells that can arrive

  with an inter-cell arrival time of $\frac{1}{PCR}$ is $MBS$

Traffic satisfying all the above conditions can be characterized as rt-VBR traffic.
Chapter 3 Scheduling Algorithms for ATM Traffic

ATM has been identified as the infrastructure technology for future B-ISDN which will carry different types of traffic [13]. One of the challenges of the researchers in this area is to design an efficient and implementable scheduling methodology which will also provide the network with the necessary QoS requirements. Recall that the three basic functions of an ATM switch are queuing, routing and header translation. Among these three functions, queuing plays an important role in the way cells are scheduled. Queuing is used in solving contention problems if two or more logical channels contend for the same output [14]. There are various queuing techniques for different switches. Scheduling algorithms providing QoS guarantees on a per-VC basis maintain separate queues for each VC. It is the responsibility of the scheduler and the queue manager to manage these VC queues. This can be quite challenging considering that the number of VCs on an ATM link. The combination of scheduling algorithm and queuing scheme greatly affect the performance of an ATM switch. The next two sections of this chapter will discuss different queuing techniques and the design issues pertaining to the realization of scheduling algorithms. In chapter 3.3 and 3.4, some common rate-based and schedule-based algorithms will be analyzed. Finally, a performance comparison of the above algorithms will be presented in chapter 3.5

3.1 ATM Queuing Schemes

There are mainly three queuing strategies that can be employed by a switch. These techniques differ from each other based on the physical location of the buffers inside the switch.
A switch can employ input buffering, output buffering or shared/central buffering. There are also some combinations and variations of the above schemes. [11]

3.1.1 Input Queuing

In an input-queued switch, cells are buffered at the input to solve the contention problem. Each input has a dedicated buffer to store the incoming cells until the arbitration logic (scheduler) decides to serve the buffer. Figure 3.1 shows an input queuing scheme.

The main advantage of an input-queued switch is that links in the switch fabric need to run only as fast as the input lines, as opposed to output-queued switch where the output queues need to run faster than the inputs. In addition, cells may arrive randomly at the switch, input queuing helps synchronize their headers before they enter the switch.

![Figure 3.1 Input Queuing](image)

Input Queuing suffers from low throughput because of head-of-line blocking. It occurs when a head-of-line cell is blocked, others in the same queue will also be blocked even though
they might have an open path to their output. If cells at each input arrive randomly distributed to all outputs, then it can be shown that an input-queued switch can achieve at most \(-58.6\%\) utilization. [15] The performance is even worst if some outputs are favored over others.

Several methods have been proposed to tackle the head-of-line blocking problem. One way to minimize the problem is by using random access scheduler [16]. In such algorithm, when a HOL cell is blocked, the next HOL cell in that queue can be routed to the destination. However, this kind of scheduler exhibits complex management and hardware design. As a result, pure input queuing is not attractive for ATM switches.

3.1.2 Central Queuing

In a central-queued or shared-memory switch, input and output ports share a common memory. Central buffers are used to resolve cell contention for internal links within the switch fabric. The performance of this kind of switch depends upon the topology of the fabric and the location of the buffers within the switching elements. Generally, cells are stored in the common memory as they arrive, and the packet header is extracted and routed to the output port. When the output port scheduler schedules a packet for transmission, it removes the packet from the shared memory. Figure 3.2 below shows a general configuration of central queuing.
The disadvantage of central queuing is that the internal buffers can destroy the cell order if fabric has multiple paths. Cells can be reordered at output ports but this requires a complex memory management system [17].

3.1.3 Output Queuing

In an output-queued switch, buffers are located at each output port to solve the contention problem. The queued cells are served using one of the scheduling disciplines. Output buffering does not require complex memory management schemes or arbitration algorithms like central and input queuing. This reduces the implementation complexity and the cost of the switch. As shown in other [15], output buffering can achieve a high throughput under random uniform traffic. The queuing delay is generally less than that of input queuing because HOL blocking does not occur.
The main disadvantage of output queuing is that the switch has to speed up its operation so that it can write up to N cells into a queue in the same time slot. The switch has to deliver N cells to a single output when all the input ports have cells destined to the same output port. This makes the switch and the queues more expensive to build than with input-buffered switch. One way to reduce this problem is to use the knockout principle or the knockout concentrator (figure 3.4). It states that if there are N inputs, the output queue needs only to run S times faster than the system, where $S < N$. The difficulty in designing the knockout circuit is to ensure that cell losses are fairly distributed among the incoming virtual circuits.
In brief, each of above methods has their advantages and drawbacks. For input queuing, scheduler is implemented at each input port as shown in literature [15]. However, input queuing has a low utilization due to the head-of-line effect. For central queuing and output queuing, scheduler is implemented at the output port. Central queuing requires a complex memory management system. Output queuing achieves the optimal throughput but requires switch operation to be much faster than the incoming traffic. Pure output queuing also exerts poor buffer utilization due to the fact that buffers are not shared among output ports. In general, the queuing technique a switch employed affects performances, such as the queuing delay and cell loss, of a scheduler. There are also variations of output queuing which combine shared buffering and output buffering [17]. In addition, microprocessors controlling the operations of a switch can now run much faster than the incoming traffic rate and can be implemented at a reasonable cost. As a result, output queuing techniques are preferred for many ATM switch vendors. In chapter 3.3, scheduling algorithms based on output queuing will be investigated.

3.2 Design Issues of Scheduling Algorithms

The theories of real-time scheduling have been developed and applied primarily to scheduling of jobs on a single processor [18]. For real-time communication, the link replaces the central processor as the central resource, while packets are the units of work requiring this resource, just as jobs compete for the usage of a processor. Basically, a scheduler allocates the usage of the link according to some predefined allocation discipline. A scheduling discipline actually has two orthogonal components. First, it decides the order in which requests are
serviced. Second, it manages the service queues of requests awaiting service. The discipline may be optimized for uniformity as in round-robin, simplicity as in FCFS (first-come-first-serve), or other criteria as in priority-based techniques. The following sections will discuss a few choices when designing scheduling algorithms.

3.2.1 Priority Levels

In a priority scheduling scheme, each connection is associated with a priority level. If there are \( n \) priority levels, and a higher-numbered priority-level corresponds to a connection with higher priority, the scheduler serves the packet from priority level \( k \) only if there are no packets awaiting service in higher priority levels \( k+1, k+2, \ldots, n \). A priority scheduler provides the high priority level flow with low queuing delay at the expense of packets at lower priority. The number of priority levels depends on the number of delay classes that the network wants to support. It may range from no priority levels to priority based on per-VC information. In B-ISDN, it is desirable to have at least three priority levels: a high priority level for urgent messages, a medium priority level for guaranteed service, and a low priority level for best-effort traffic [19].

Priorities may be assigned by an end-user or by a scheduler. It is usually assigned according to some properties of the connection, such as the deadline or arrival period. It may be assigned statically for all packets in a connection, or may be assigned dynamically at the time of arrival of a packet. The scheduler may be preemptive or non-preemptive, depending on when the
scheduler enforces priorities. Since the ATM cell size is small, a ATM scheduler will not preempt an active transmission until the current cell is serviced.

For a pure priority scheduler, starvation occurs when the high priority level traffic misbehaves, preempting all the lower priority level traffic. This happens when the high priority class always has something to send, then the bandwidth available to the lower level traffic will be decreased. As a result, in a priority scheduler, it is important to have appropriate admission control and policing strategies to restrict the service rates of each class.

**Degree of Aggregation**

When designing scheduling discipline, one of the issues to decide is the degree to which individual connections are aggregated in deciding their service order. If a scheduler uses a single state variable to describe all connections, then these connections must share the same QoS. In contrast, if a scheduler maintains per-connection state, it can give different connections different bandwidth and delay bounds. [19]

For instance, denote the largest burst size of each connection $i$ of a first-come-first-serve scheduler as $b_i$. If the scheduler aggregates all connections, it cannot distinguish between packets from individual connections. The worst case delay bound for connection $i$, $d_i$, is achieved when bursts from all sources arrive simultaneously, which leads to a queue length $\sum b_i$. As a result, if this connection requires a delay bound, the CAC can only ensure that:

$$\sum b_i \leq d_i$$
where it takes the sum over all connections at the switch.

When a scheduler aggregates connections in classes corresponding to their delay bounds, it uses multilevel priority where increasing priority levels correspond to increasing delay bounds. At the highest priority level, it ensures that

$$\sum_{b_i \in \text{highestpriorityclass}} b_i \leq \text{delay bound of highest priority class}$$

At lower priority levels, the delay bound takes the sum over its own level plus all the higher priority levels. The tradeoff between these two is that the latter requires more scheduler state space.

**Implementation**

Priorities are simple to implement. To make a scheduling decision, a scheduler needs to determine only the highest priority nonempty service queue. It requires only a small amount of scheduling state for buffer management. For scheduling real-time traffic, it is essential to have a scheduling discipline that has different priority levels, so that the QoS parameters, such as delay and jitter, can be bounded for different connections.

### 3.2.2 Work-Conserving vs Non-Work-conserving

A scheduling policy can be classified as either work-conserving or non-work-conserving [18]. A method is work-conserving if an output link will never be idle as long as there are packets waiting to use that link. In contrast, a non-work-conserving method may be idle even if there are packets to serve. Work-conserving schemes may seem attractive, as it promises lower
average end-to-end delay and it is easier to implement than that of non-work-conserving. However, with non-work-conserving methods, the traffic arriving at downstream switches is more predictable, thus reducing both the buffer size necessary at the output queues, and the delay jitter experienced by a connection. For real-time traffic, reducing the jitter and maximum packet delay is usually more important than reducing the average delay. Moreover, in order to provide a better link utilization, a non-work-conserving scheme can simply serve other best-of-effort traffic whenever there are no packets eligible to serve. It may be desirable if a scheduler can have a mix of non-work-conserving and work-conserving services, so that both the delay jitter and average delay can be bounded.

*Implementation*

As of the time of writing this thesis, non-work-conserving disciplines have not been implemented in commercial switches. One reason for this is their implementation complexity. There are also strong arguments in the literature stating that elastic buffer can be used at the endpoints of a connection to absorb the delay jitter [19]. This literature claims that due the rapid dropping cost of memory, it may not be necessary to bound the delay jitter. The rebuttal to this is that non-work-conserving disciplines also reduce the number of buffers needed at each switch. These buffers are more expensive than those at the end-points, since they have to be accessed much faster inside switches.
For work-conserving disciplines, a scheduler has to make sure that the bandwidth can be allocated to lower priority levels if there are no higher priority level traffic requesting it. Nowadays, all of the commercial switches are work-conserving.

3.2.3 Rate-Based vs Schedule-Based

Methodologies of service disciplines can be based in aspect, namely, rate-based and schedule-based [18]. For rate-based methods, a predefined set of allowable rates is established. Priority of each class can be assigned statically or dynamically. The allocated bandwidth guarantees a fixed maximum delay for each packet in that rate class. For schedule-based methods, the scheduler analyses the potential interactions between packets of different connections, and determines if there is any possibility of a deadline being missed. Priorities are assigned dynamically based on deadlines. Details of implementing schedulers will be covered in later chapters.

3.2.3 Frame-Based vs Sorted-Priority

A scheduler can be based on framing or sorted-priority, depending on the internal architecture of it [20]. In a sorted-priority scheduler, a virtual time is associated with each outgoing link of the switch. Each time a packet arrives or departs, the virtual time is updated. A timestamp, which is computed as a function of virtual time, is associated with each packet in the switch as it arrives. The service order of the packets is then based on the timestamps of the packets.
In a frame-based scheduler, time is split into frames of fixed or variable length. Reservations of sessions are made in terms of the maximum amount of traffic the session is allowed to transmit during a frame period [21]. Thus, the server may remain idle if sessions transmit less traffic than their reservations over the duration of a frame. There are also frame-based disciplines that use variable frame size, so that whenever the session is overbooked, a new frame can be started early.

**Implementation**

Detailed implementation issues will be discussed in a later chapter for sorted-priority algorithms. For frame-based discipline, the implementation complexity is high in general. There is a synchronization requirement so that the framing structure is the same at both the sending and the receiving ends of the link [22]. Moreover, algorithms using variable frame size suffer from complex implementations. This is why framing schemes are not popular.

### 3.2.5 Summary

Table 3.1 below shows some of the common scheduling algorithms with their characteristics.

<table>
<thead>
<tr>
<th>Scheduling Disciplines</th>
<th>Priority Queues</th>
<th>Work-Conserving Or Non-Work-Conserving</th>
<th>Rate-Based or Schedule-Based</th>
<th>Framing or Sorted-Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>First-Come-First-Serve</td>
<td>N</td>
<td>WC</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>Round-Robin</td>
<td>N</td>
<td>WC</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>Hierarchical Round Robin</td>
<td>Y</td>
<td>NWC</td>
<td>R</td>
<td>F</td>
</tr>
<tr>
<td>Stop-and-Go</td>
<td>N</td>
<td>NWC</td>
<td>R</td>
<td>F</td>
</tr>
<tr>
<td>Weighted Fair Queuing</td>
<td>Y</td>
<td>WC</td>
<td>R</td>
<td>SP</td>
</tr>
</tbody>
</table>
In the next two sections, some of the above scheduling disciplines will be investigated, although we will focus on sorted-priority techniques. Priority based methods are easier to implement and are more suited to the multi-service nature of B-ISDN.

### 3.3 Rate-Based Scheduling Algorithms for ATM

As discussed in the last section, scheduling algorithms can be classified as rate-based or schedule-based. A rate-based algorithm is parameterized by a set of statistics such as average rate, maximum rate, and burstiness, and is assigned a vector of values corresponding to these values [23]. According to these values, the service order of different classes is determined. Thus, the allocated bandwidth guarantees a fixed maximum delay for each packet in that rate class. In the next section, an ideal scheduling discipline, called Generalized Processor Sharing (GPS) will be introduced. Based on GPS, two other algorithms - WFQ and SCFQ, which try to approximate GPS, will also be analyzed.

#### 3.3.1 Generalized Processor Sharing (GPS)

In order to provide effective flow control in a network, an appropriate scheduling discipline at each node or switch is essential. It has to be efficient, flexible and analyzable. GPS

<table>
<thead>
<tr>
<th>Scheduling Discipline</th>
<th>Y</th>
<th>WC</th>
<th>R</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Clocked Fair Queuing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet Generalized Processor Sharing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delay Earliest Due Day</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jitter Earliest Due Day</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1 Characteristics of Various Scheduling Algorithms
is a work-conserving, rate-based and sorted-priority type scheduling discipline, which seems very appropriate for integrated services networks. Although it has the drawback of being unimplementable in practice, it can serve as an ideal scheduler that other disciplines can compare against. GPS provides the best delay guarantees and the best absolute fairness. Other techniques can only approximate its performance.

3.3.1.1 Theory of GPS

Intuitively, a GPS scheduler serves packets as if they are in separate logical queues, visiting each nonempty queue in turn and serving an infinitesimally small amount of data from each queue so that in any finite time interval, it can visit every logical queue at least once. Connections can be associated with service weights, and they receive service in proportion to this weight whenever they have data in the queue. If a connection does not have data, the scheduler skips to the next nonempty queue [24].

A GPS server serving \( N \) sessions is characterized by \( N \) positive real numbers, \( \phi_1, \phi_2, \ldots, \phi_N \), denoting service weights. Let \( S_i(t_1, t_2) \) be the amount of session \( i \) traffic served in the interval \([t_1, t_2] \). A session is backlogged at time \( t \) if a positive amount of that session's traffic is queued at time \( t \). Then a GPS server is defined as follow [24]:

\[
\frac{S_j(t_1, t_2)}{S_j(t_1, t_2)} \geq \frac{\phi_j}{\phi_j}, \quad j = 1, 2, \ldots, N
\]  

(1)

From equation (1), it is shown that the service time that GPS provides to a session is proportional to its assigned service weight (\( \phi \)). The larger the weight, the more the services time
a connection gets. This holds true for any session \( i \) that is backlogged throughout the interval \([t1, t2]\). Summing over all sessions \( j \):

\[
S_i(t1, t2) \cdot \sum_j \phi_j \geq (t2 - t1)r\phi_i
\]  

(2)

where \( r \) is the link speed and \( \sum_j \phi \) includes all the nonempty queues (or backlogged sessions).

Working around equation (2), then session \( i \) is guaranteed a rate \( (g_i) \) of

\[
g_i = \frac{\phi_i}{\sum_j \phi_j} r
\]

(3)

As a result, session \( i \) is guaranteed a minimum service rate of \( r_i \) during any interval when it is backlogged. GPS is a very useful algorithm, it allows other disciplines to evaluate their performances. Here are some features of GPS [24]:

1. Define \( r_i \) as the average rate for session \( i \). Then as long as \( r_i \leq g_i \), the session can be guaranteed a throughput of \( \rho_i \) independent of the demands of other sessions. Moreover, a session \( i \) backlog will always be cleared at a rate \( \geq g_i \).

2. The bandwidth allocated to each session can be modified by varying \( \phi \), as long as the combined average rate of the sessions is less than \( r \).

3. GPS can guarantee a worst-case network queuing delay when the sources are constrained by leaky buckets. As a result, GPS is particularly attractive for session sending real-time traffic.

The delay bounds of GPS and other GPS approximate disciplines will be given in Section 3.4.
3.3.1.2 A GPS Example

Figure 3.5 below shows an example of GPS service order. In this example, there are 11 sessions sharing the link. Assume all packets have the same size of 1 and the link speed is 1. The guaranteed rate for session 1 is 0.5, while the guaranteed rate for session 2 to 11 is 0.05.

As shown in figure 3.5. Session 1 sends 11 back-to-back packets at time 0 while sessions 2 to 11 sends only one packet each at time 0. According to GPS, the server will serve in turn a small amount of packets from each connection depending on their rates. With a guaranteed rate of 0.5, session 1 requires 2 time units to service a packet while other sessions require 20 time units to service a packet. While GPS is an ideal discipline which bounds the queuing delay and provides ideal fairness, it is unimplementable due to its infinitesimal packet size assumption. The next section will introduce a scheduling discipline that tries to approximate GPS and is implementable.
3.3.2 Weighted Fair Queuing (WFQ)/ Packetized Generalized Processor Sharing (PGPS)

Although GPS is an idealized servicing discipline, it does not transmit packets as entities. It assumes that the server can serve all backlogged sessions simultaneously and that a packet is infinitely divisible. In a realistic packet network system, only one session can receive service at a time and an entire packet must be served before another packet can be served. There are different ways of emulating GPS service in a packet system. The most popular one is Weighted Fair Queuing (WFQ), also called Packetized Generalized Processor Sharing (PGPS). WFQ approximates GPS scheduling, but does not make GPS's infinitesimal packet size assumption. ATM switch vendors, like CISCO and LUCENT, have built some of their switches with WFQ scheduling disciplines [25].

3.2.2.1 Theory of WFQ and GPGS

The intuition behind WFQ is to compute the time in which a packet would complete service had it been served by a GPS server. The packets are then served in order of these finish times, also called finish numbers. A finish number is merely a service tag that indicates the relative order in which the packet is to be served, and has nothing to do with the actual time at which the packet is served.
From equation (1) in the last section, it gives the amount of a session served in the interval \([t_1, t_2]\). In literature [26], Parekh establishes the following relationships between the GPS system and its corresponding packet WFQ system:

\[
d_{i,WFQ}^k - d_{i,GPS}^k \leq \frac{L_{\text{max}}}{r} \quad \forall i, k
\]

\[
W_{i,GPS}(0, \tau) - W_{i,WFQ}(0, \tau) \leq L_{\text{max}} \quad \forall i, r
\]

where \(d_{i,WFQ}^k\) and \(d_{i,GPS}^k\) are the times at which the \(k\)th packet in session \(i\) departs under WFQ and GPS, respectively. \(W_{i,WFQ}(0, \tau)\) and \(W_{i,GPS}(0, \tau)\) are the total amounts of service given to session \(i\) up to time \(\tau\) under WFQ and GPS, respectively, and \(L_{\text{max}}\) is the maximum packet size, which is constant for ATM packets.

Equations (4) and (5) shows that WFQ can lag GPS by at most \(\frac{L_{\text{max}}}{r}\) [28]. In other words, Parekh proved that WFQ cannot lag fall behind GPS with respect to servicing a given session by more than one maximum size packet. However, packets can leave much earlier in a WFQ system than in a GPS system, which means that WFQ can be far ahead of GPS. This can result in bursty service over short time scales. A variant of WFQ, called Worst-Case-Fair-WFQ was proposed to avoid this problem by serving the packet with the smallest finish number among all the packets that have started service in the corresponding GPS system. Nonetheless, WFQ and \(W^2\)FQ provide the same end-to-end delay bounds.

### 3.3.2.2 Examples of WFQ/GPS

As section 3.3.1.2 shows an example of how GPS works, this section shows the service order of WFQ corresponding to the same arrival pattern. Figure 3.6 below shows an example of
WFQ/PGPS service order. Again, there are 11 sessions sharing the link. Assume all packets have the same size of 1 and the link speed is 1. The guaranteed rate for session 1 is 0.5, while the guaranteed rate for session 2 to 11 is 0.05.

As shown in figure 3.6, session 1 sends 11 back-to-back packets at time 0 while sessions 2 to 11 send only one packet each at time 0. According to WFQ/PGPS, at time 0, all 11 sessions have packets backlogged. Since packets \( p_j^i \) finish at time \( 2k \), for \( k = 1, 2, \ldots, 10 \), and 21 for packet \( p_{21} \). Each of the first packets in sessions 2 to 11 has a finish number of 20. As a result, 10 back-to-back packets from session 1 will be served first, then the packets from 2 to 11 and finally, packet 11 from session 1 will be served. Notice how the packet service order is different from that of GPS. This has important implications on how WFQ or PGPS is related to GPS.

3.3.2.2 Finish number computation

The service order of WFQ or PGPS is based on a Finish Number. The finish number uses a virtual time to track the progress of GPS that will lead to a practical implementation of WFQ/PGPS. The virtual time, \( v(t) \), is defined in [26] as follow:

![Figure 3.6 Arrival pattern and WFQ/PGPS service order](image)
\[ v(t_2) - v(t_1) = (t_2 - t_1) \left( \sum_{j \in B_j} r_j \right)^{-1} \]  \hspace{1cm} (6)

where \( v(t) \) is piecewise linear with the slope

\[ \frac{v(t_2) - v(t_1)}{t_2 - t_1} = \frac{dv(t)}{dt} = \left( \sum_{j \in B_j} r_j \right)^{-1} \]  \hspace{1cm} (7)

where \( B_j \) is the set of backlogged connections at time \( t \) under the reference of GPS.

In GPS, if connection \( j \) is backlogged at time \( t \), it receives service at a rate of \( \phi \frac{dv(t)}{dt} \).

Therefore \( v(t) \) can be interpreted as increasing at the marginal rate at which backlogged connections receive services in GPS. With this virtual time, the virtual finish number is defined as [17]:

\[ F_j^k = \frac{L_j^k}{\phi} + \max\{F_j^{k-1}, v(a_j^k)\} \]  \hspace{1cm} (8)

and

\[ F_j^0 = 0 \]

where \( j \) is the connection number, \( k \) is the \( k \)th packet, \( \phi \) is the allocated rate and \( L_j^k \) is the maximum packet size.

There are three attractive properties of the virtual time interpretation from the standpoint of implementation [24]. First, the virtual finish time can be determined at the packet arrival time. Second, the packets are served in the order of virtual finish time, which makes it easy for the scheduler to select which queue to serve. Finally, only the virtual time has to be updated when
there are events in the GPS system. However, there is an overhead in keeping track of the sets of $B_j$ and thus the calculation of virtual finish time $v(t)$. These updates have to be done whenever a session changes from backlogged to non-backlogged or vice versa. Nowadays, ATM switches can have as much as 64,000 or more connections per link. There may be a large number of sessions changing states at the same time. Consider an example of having three sessions sharing the bandwidth of a link. Their shares are 50%, 25% and 25% for session 1, 2, 3 respectively. Session 1 starts service at time 0. It receives 100% of the link, that is 2 times it is assigned, as a result the slope of the virtual time function is 2. The virtual time function in figure 3.7 is used to track down all these changes in service rate. At interval 2 to 4 microseconds, the slope changes to

$$\frac{1}{(0.5 + 0.25)} = 1.33,$$

indicating that sessions 1 and 2 are receiving 1.33 times of their reserved rates. As shown in figure 3.7, whenever a session starts service or ends service, the virtual time has to be recalculated. When the number of connections is 64,000, it would take a long time for the scheduler to calculate this virtual time because the calculation is O(N), where N is the number of active sessions. Moreover, the evaluation of $v(t)$ has to be performed in real time for WFQ. The implementation of a precise WFQ system based on GPS emulation is either very expensive in hardware resources or very slow to compute. The main time consuming operation is maintaining $v(t)$. From equation (7), the service rate of each backlogged session = $\phi \frac{dv(t)}{dt}$. The evaluation of $\frac{dv(t)}{dt}$ grows as number of sessions sharing the link increases. In addition, for a large number of sessions, chances of the system changing state (from backlog to idle, or vice versa) are considerably bigger.
A number of approximation to WFQ have been proposed, such as Worst Case Weighted Fair Queuing ($W^2FQ$) in [27] and Start-Time Fair Queuing (STFQ) in [28]. In the next section, we explain one such technique, Self-Clocked Fair Queuing (SCFQ), which is proposed by Golestani in [29] to simplify the calculations of the finish numbers.

![Backlog Chart and Virtual Time Graph](image)

**Figure 3.7** An example of virtual time $v(t)$ in PGPS

### 3.3.2 Self-Clocked Fair Queuing (SCFQ)

One of the biggest problems with WFQ/PGPS is implementing the algorithm for updating the round number on packet arrival. Self-Clocked Fair Queuing (SCFQ) is a scheduling discipline designed to speed up the finish number calculation [30].

#### 3.3.2.1 The SCFQ Algorithm

As argued in the last section, WFQ suffers from the complexity of calculating the virtual time. The system has to compute the virtual time whenever a session queue changes from busy to
idle, and vice versa. It is possible for the system sessions to change states many times for a short period of time. SCFQ adopts a different notion of virtual time. Instead of linking virtual time to work progress, it uses a measure of progress in the system itself. The finish number is calculated as follow [29]:

\[
F_j^k = \frac{L^k_j}{r_j} + \max\{F_j^{k-1}, \nu(a_j^k)\} \quad \text{and} \quad F_j^0 = 0 \tag{9}
\]

where \(\nu(t)\) = finish number of the packet in service at time \(t\), \(j\) is the session number, \(k\) is the \(k\)th packet, \(r\) is the allocated rate and \(L^k_j\) is the maximum packet size.

Equation (9) is similar to that of equation (8) of GPS. The only difference is the way the virtual time is calculated. The virtual time in SCFQ is just the finish number of the packet currently receiving service. In his literature [29], Golestani has shown that SCFQ scheme is a near optimal fair queuing scheme.

3.3.2.2 Mechanism of SCFQ

![Functional Diagram of SCFQ](image)

Figure 3.8 Functional Diagram of SCFQ:
The SCFQ scheme works as follow:

a) Compute the tag, \( F_j \), for the received packet according to equation (9)
b) Store the packet in the FIFO queue belonging to session \( j \).
c) Identify any new head-of-line packet for each class queue and insert it into a sorted list of head-of-line tags.
d) Transmit the packet corresponding to the lowest tag in the sorted list.
e) Store the lowest tag value which is used for the computation of tags of subsequently arriving packets.

The advantage of this scheme is that the computation of virtual time is negligible because it is simply extracted from the packet which is being serviced.

3.3.2.3 An SCFQ Example

This section illustrates the service order of SCFQ corresponding to the same arrival pattern. Figure 3.9 below shows an example of SCFQ service order. The number of sessions, link rates and arrival patterns are all the same as the ones in section 3.3.2.2.

```
Connection 1
     ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑
Connection 2
     ↑
            ....
Connection 11
     ↑
SCFQ service order
     ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑
```

Figure 3.9 Arrival pattern and SCFQ service order
As shown in figure 3.8, session 1 sends 11 back-to-back packets, while sessions 2 to 11 send only 1 packet. At time 0, \( p_1 \) has finish number of 2 while other packets in sessions 2 to 11 have finish number of 20. At time 1, packet in session 2 is served, and the current finish number is set to 20. As a result, the finish number of \( p_2 \) will be equal to \( \max(2, 20) + 2 = 22 \). As a result, \( p_2 \) won’t start service until all other 10 \( p_i \) finish services. This is shown in figure 3.9. In this example, it shows that even though connection 1 sends packets according to the specified average rate, its packets still get delayed significantly. In general, SCFQ has a higher end-to-end delay bound than that of WFQ.

### 3.4 Schedule Based Scheduling Algorithms for ATM

As presented in section 3.2, a schedule-based scheduler analyses the potential interactions between packets of different connections, and determines if there is any possibility of a deadline being missed [31]. The most common schedule-based disciplines is of the Earliest-Due-Date (EDD) emulated type. In traditional EDD scheduling, each packet is assigned a deadline, and the scheduler serves packets in order of their deadlines. The deadline of a packet is the sum of its arrival time and the period of the traffic stream. At connection setup, the scheduler uses a schedulability test to ensure that the new connection as well as the established ones can meet their corresponding deadlines [32]. However, this kind of schedulability test is not feasible in packet network system due to the large number of connections. Moreover, EDD does not guarantee the delay jitter, which is important for real-time traffic. Delay-EDD and Jitter-EDD are algorithms which can provide this quality.
3.4.1 Delay Earliness Due Date (D-EDD)

Delay-EDD is an extension of EDD which specifies a process by which the scheduler assigns deadlines to packets [32]. During connection setup, each source negotiates a service contract with the scheduler. The contract states that if a source obeys its promised traffic specification, such as peak and average sending rate, then the server will provide a delay bound. The scheduler ensures not only the peak rates of the admitted calls is smaller than the link capacity, but also that even in the worst case, when every connection sends traffic at its peak rate, it meets its delay bound.

3.4.1.1 Delay-EDD Concepts

The key to delay-EDD lies in the way the scheduler assigns deadlines to packets. The scheduler sets a packet's deadline to the time at which it should be sent had it been received according to the contract. The scheduler thus reserves bandwidth at a connection’s peak rate. By doing this, it ensures that it has served the previous packet before the next one packet arrives, so that packets from each connection obeying the traffic constraint can have a hard delay bound. Unlike rate-based algorithms, the delay bound is independent of its bandwidth reservation. This is important for low bandwidth connection which also needs a tight delay bound, such as low bit-rate video and voice [31].
3.4.1.2 Deadline/Eligibility Time

In WFQ and SCFQ, there is a state variable called virtual finish time ($F$). In delay-EDD, the state variable is called Expected Deadline ($ExD$). The $F$ and $ExD$ are used as the priority indices of packets. $ExD$ is calculated as follow:

$$ ExD^k_{i,j} = \max\{a^k_{i,j} + d_{i,j}, ExD^k_{i,j} + X_{\text{min}} j\} $$  \hspace{1cm} (11)

where $i$, $j$ and $k$ denote the server number, connection number and packet number respectively. $X_{\text{min}} j$ is the minimum packet interarrival time for connection $j$, $d_{i,j}$ is the local delay bound assigned to session $j$ at server $i$ at connection setup.

As shown in equation (11), delay-EDD uses two parameters to update this priority index, namely $X_{\text{min}}$ and $d_{i,j}$. The use of these two parameters solves the problem of coupling between the delay bound and bandwidth which GPS-emulation scheduler has. However, EDD techniques suffer from fairness problems.

3.4.2 Jitter Earliest Due Date (J-EDD)

In a jitter-EDD scheduler, a delay-jitter regulator precedes the EDD scheduler [33]. With a delay-jitter regulator, all packets on a connection receive similar delays through their path. Thus, a network of jitter-EDD schedulers can give connections end-to-end delay and delay-jitter bounds.

3.4.2.1 Theory of Jitter-EDD

The jitter-EDD discipline extends delay-EDD to provide delay-jitter bounds. After a packet has been served at each server, a field in its header is stamped with the difference between
its deadline and the actual finishing time. A regulator at the input of the next server holds the packet for this period before it is made eligible for scheduling. Jitter-EDD scheduler uses a delay-jitter regulator, which guarantees that the sum of the queuing delay in the previous switch and the regulation delay in the current switch is constant. This removes the effect of queuing-delay variation in the previous switch.

3.4.2.2 Eligibility Times

As delay-EDD utilities an expected deadline \( (ExD) \), jitter-EDD also relies on an eligibility time.

The eligibility time is given as follow:

\[
ExD_{i, j}^k = ExD_{i-1}^k + d_{i-1, j} + L_i^k
\]  

(12)

where \( d_{i-1, j} \) is the local delay bound for the connection at the \( (i-1)th \) server, that is the delay bound at the previous switch, and \( L_i \) is the maximum link delay between switch \( i-1 \) and switch \( i \). For the \( kth \) packet, it is eligible for service at the \( 0th \) switch the moment it arrives. However, at subsequent switches, it becomes eligible for service only after a fixed time interval of length \( d_i + L_i \).

3.5 Performance and Implementation Comparison

In this section, the performance of the scheduling disciplines discussed in the last two sections will be compared. Since, this thesis is focused on real-time transport, the performance of
a scheduling discipline will be based on its delay bound and jitter-delay bound. In addition, the implementation complexity of these algorithms will be discussed.

3.5.1 Quality-of-Service Performance Bound

The most important parameters in the service contract are the specifications of performance requirements and traffic characteristics. There are four performance parameters that are commonly used. They are cell loss, end-to-end delay, jitter-delay and bandwidth. Among these four parameters, end-to-end delay bound and end-to-end delay-jitter bound are of the most important to real-time traffic. The delay jitter for a packet stream is defined as the maximum difference between delays experienced by any two packets [17]. It is especially useful in the context of playback applications. If the delay jitter over a connection is bounded, the receiver can calculate the amount of buffer space needed to eliminate the jitter. The smaller the jitter bound, the less amount of buffer space is needed. For a guaranteed service, it is more important to provide end-to-end delay and delay-jitter bounds than low average delay, as early packet arrivals may be undesirable. This is one of the most significant difference between the performance requirements of guaranteed-performance service and best-effort service.

3.5.1.1 End-to-End Delay Bound

As proved from theorem 2 of [24], the behavior of GPS and PGPS are identical. Thus all the bounds developed for GPS can be applied to PGPS/WFQ. In [26], Parekh proved that the end-to-end delay bound, $D_n$, of GPS and PGPS/WFQ is:
\[ D_{i}^{GPS} \leq \frac{\sigma_i}{\rho_i} \]  \hspace{1cm} (13)

\[ D_{i}^{PGPS} \leq \frac{\sigma_i + (K - 1)L_{\text{max}}}{\rho_i} + \sum_{i=1}^{K} \frac{L_{\text{max}}}{r_i} \]  \hspace{1cm} (14)

where \( \sigma_i \) is the burst size, \( K \) is the number of hops traversed by session \( i \), \( \rho_i \) is the guaranteed rate for the connection, \( L_{\text{max}} \) is the maximum size of the packet and \( r \) is the link speed.

In equation (13), it basically states that the delay bound for GPS is just the maximum burst size divided by the allocated rate. In equation (14), the first term is the queuing delay, while the second term is the propagation delay. In high speed networks, the first term of equation (14) is likely to dominate.

Assume that the link speed, \( r^m \rightarrow \infty \), equation (14) becomes:

\[ D_{i}^{PGPS} \leq \frac{\sigma_i + (K - 1)L_{\text{max}}}{\rho_i} \]  \hspace{1cm} (15)

The term \( \frac{(K - 1)L_{\text{max}}}{\rho} \) in equation (15) is the delay incurred when a packet travels through nodes at which there are all \( K-1 \) hops for session \( i \) to travel. This equation strongly indicates that smaller packets should be chosen for service, so that the term \( \frac{L_{\text{max}}}{\rho} \) is small. Consequently, PGPS/WFQ is very suitable for ATM networks since ATM packets are small in size (424 bits).
As mentioned in previous section, PGPS has a higher implementation complexity than SCFQ. The end-to-end delay bound of SCFQ is given in [29][34] as follow:

\[ D_{i}^{SCFQ} \leq \frac{\sigma_i}{\rho_i} + (K - 1)L_{\text{max}} + \sum_{i=1}^{K} N_i \frac{L_{\text{max}}}{r^i} \]  

(16)

where \( \sigma_i \) is the burst size, \( K \) is the number of hops traversed by session \( i \), \( \rho_i \) is the guaranteed rate for the connection, \( L_{\text{max}} \) is the maximum size of the packet, \( r \) is the link speed and \( N_i \) is the number of sessions sharing the link with connection \( i \) at the \( ith \) switch.

The delay bound provided by SCFQ is larger than that provided by WFQ/PGPS. This is due to the inaccuracy introduced by the approximation algorithm in calculating the virtual time. For both SCFQ and WFQ, there is a coupling problem between the end-to-end delay bound and bandwidth provided to each connection. Specifically, the end-to-end delay bound is inversely proportional to the allocated average rate. Thus, in order for a connection to get low delay, it has to request a high bandwidth. This will result in a waste of bandwidth when a low delay connection requests low bandwidth. In contrast, the Delay-EDD algorithm and Jitter-EDD algorithm do not have the delay and jitter coupling problem. Their delay bounds are as follow [34]:

\[ D_{i}^{D-EDD} \leq \sum_{i=1}^{n} d_{i,j} \]  

(17)

\[ D_{i}^{J-EDD} \leq \sum_{i=1}^{n} d_{i,j} \]  

(18)

where \( d_{i,j} \) is the delay at the \( ith \) hop for session \( j \).
Equation (17) and (18) state that the end-to-end delay bound of them are just equal to the total worst case delay along the connection path. Note that the delay bound for both EDD schemes are the same.

3.5.1.2 End-to-End Delay-Jitter Bound

The end-to-end delay jitter bounds for the D-EDD and J-EDD algorithms are given below [34]:

\[ J_{i}^{PGPS} \leq \frac{\sigma_i + (K - 1)L_{\text{max}}}{\rho} \]

(19)

\[ J_{i}^{SCFQ} \leq \frac{\sigma_i + (K - 1)L_{\text{max}}}{\rho} + \sum_{i=1}^{K} (N_i - 1) \frac{L_{\text{max}}}{r_i} \]

(20)

\[ J_{i}^{D-EDD} \leq \sum_{i=1}^{n} d_{i, j} \]

(21)

\[ J_{i}^{J-EDD} \leq D_r + \sum_{i=1}^{n} d_{i, j} \]

(22)

where \( D_r \) is the worst case delay of the regulator.

Recall that the delay jitter bound is the maximum difference between delays of any two packets. The delay jitter bounds for WFQ, SCFQ and D-EDD are equal to their maximum end-to-end queuing delay. Since these three disciplines are all work-conserving, a packet may experience little or zero queuing delay when the network traffic is light while another packet may experience a longer queuing delay when the network traffic is heavy [34]. On the other hand, the J-EDD uses a jitter-delay regulator to regulate the input traffic stream. By setting up the
parameters for a jitter-delay regulator properly, the jitter-EDD algorithm can guarantee a tight bound on delay jitter.

3.5.2 Implementation Complexity

In a high-speed network, a server may need to select the next eligible packet for transmission after every time packet departure. For example, on a SONET OC-12 link operating at 622 Mb/s, an ATM packet can arrive every 0.7 microsecond. Thus, the server has only little time to make a decision. A scheduling discipline in a high-speed network should require only a few simple operations. In particular, the number of operations to implement a discipline should be as independent of the number of connections as possible. As a result, it is important to have a scheduler which is both efficient and easy to implement. All of four algorithms that are discussed in previous sections, namely WFQ/PGPS, SCFQ, delay-EDD and jitter-EDD, use timestamping techniques, and sorted-priority queues [17]. In addition, jitter-EDD uses a delay-jitter controller to control the traffic pattern entering the scheduler.

3.5.3 Sorted-Priority Queues

With timestamping schedulers, cells are served in the order of timestamps. The scheduling unit is a priority queue which contains up to N head-of-line cells. The insertion operation for a sorted priority queue has an intrinsic complexity of $O(\log N)$ [34]. This can be implemented by a software approach, such as binary heap.
3.5.3.1 Software Sorting

Using a software approach, in the worst-case, enqueue and dequeue require about $k\log_2 N$ operations, where $k$ is a constant number of operations which include a read, a write and a compare. In an output-buffered switch, the operations must be executed up to $P$ times per slot, where $P$ is the number of input or output ports. In a network that is designed to support a large number of connections, it may be infeasible to implement an operation that has an $O(\log N)$ complexity at very high speed. The example below shows a typical operation speed required.

Assume $P=16$ times per slot, $c=OC.48$ bandwidth=2.4Gbps, $N=32000$ connections, size of ATM cell = 424 bits, in this case, the $k$ operations must be done in:

$$\frac{424}{c*P*\log_2 N} = \frac{424}{2.4G*16*\log_2 32000} = 0.7ns$$

Even when we consider some of the industries’ very fast memory chips, which run speed range of few hundred MHz, they are still not fast enough to implement this example. As a result, most of the sorting schedulers are implemented in hardware.

3.5.3.2 Hardware Sorting

There are some efficient ways to do hardware sorting, for instance, the sequencer developed by Jonathan Chao in [35] has a $O(1)$ operation time. Kazemi-Nia and Alnuweiri [36] also present a parallel priority queue (PPQ) architecture with $O(1)$ enqueue and dequeue time.
3.5.4 Timestamp Calculation

Another issue that affects the implementation of a scheduler is the calculation of the timestamp, which must be done \( P \) times per cell slot. To calculate the timestamps for either SCFQ or delay-EDD, the scheduler requires two memory accesses for its rate and the last timestamp as well as an addition operation. SCFQ also requires a maximum selection operation. Because the number of operations is fixed per packet, SCFQ has \( O(1) \) time complexity for maintaining the scheduler state. All of these operations only take \( O(1) \) time. For WFQ/PGPS, an iterative deletion algorithm that has \( O(N) \) complexity is needed to calculate the virtual time function [23]. Consequently, WFQ/PGPS is considered to be more complex.

3.5.5 Memory Storage

Since there is a timestamp for each cell going into the scheduler, timestamps require substantial amount of memory. The above scheduling algorithms all require a timestamp per cell storage which can take up a lot of memory space. An implementation of SCFQ was given in [37] where the timestamping is only done when cells reach HOL position. The implementation shows the same ordering as the timestamp-per-cell SCFQ. It has the advantage that only the timestamps for head-of-line cells need to be stored.

3.5.6 Implementation of Jitter-Delay Regulator

In jitter-EDD, a jitter-delay regulator is located before the delay-EDD scheduler. A common mechanism to implement a jitter-delay regulator is by means of calendar queues [38]. It consists if a clock and an array of pointers to lists of eligible cells. The clock ticks at fixed time
intervals. In every tick of the clock, the linked lists in the array indexed by the current time are appended at the end of the scheduler's linked list. Cells from the linked list of one priority level in the regulator are appended to the same priority level in the scheduler. The scheduler just selects the first packet at the highest priority queue that is nonempty. The operations of array indexing, insertion at the tails, deletion from the heads, have $O(1)$ complexity.
Chapter 4 Simulation Model

This chapter develops a model for evaluating various scheduling disciplines. In the following section, the modeling of discrete rt-VBR traffic will be presented. Then, the architecture of an output queuing switch with a scheduler will be introduced for use in the simulation model. In the last section, the implementations of three scheduling disciplines will be presented. All of these three algorithms are based on SCFQ.

4.1 Input Traffic Model

This section describes an input traffic generator which is based on a discrete-time model. The model includes traffic generators and leaky buffers. This traffic model is used to generate rt-VBR traffic. Due to its' traffic characteristics and QoS guarantee provided, rt-VBR traffic model is especially appropriate to model traffic utilized by real-time applications.

4.1.1 Applications of rt-VBR Traffic

There are wide ranges of real-time applications, such as two-way interactive voice calls, video conferencing, and TV distribution. Due to the bursty nature of compressed video, rt-VBR is very suitable for real-time transmission of compressed video. In such an application, the sender digitizes and compresses a continuous audio/video stream, and sends it to the receiver, which plays back the stream [17]. This playback receiver should choose playback instants so that when it is ready to output the information contained in a packet, the packet has already been completely received. One way to ensure that is to have the scheduler which guarantees a delay bound on the
packet. In addition, if the delay jitter over a connection is bounded, the receiver can eliminate delay variations in the network by delaying the first packet on that connection by the delay-jitter bound using an elastic buffer. Thus, the larger the delay-jitter bound, the larger the elastic buffer required at the receiver. Packets arriving early must be stored in the receiver buffers. As a result, it may be better to delay such packets in the network, so that packets from other delay-sensitive connections can get lower queuing delays. In order to guarantee the delay and delay-jitter, the traffic source has to conform to certain parameters, such as PCR and MBS [39]. In the next section, traffic generators, which are used in this thesis to generate rt-VBR traffic in the simulations, will be discussed.

4.1.2 Traffic Generators

For our simulation purposes, the traffic source is modeled in the C language while other components, such as the memory buffers and scheduler are implemented in VHDL. This technique is sometimes called co-simulations. The use of C language gives more flexibility in mathematical modeling and calculations. The traffic source is linked with other VHDL components by using the Synopsys C-Language Interface (CLI).

The traffic model is a deterministic one. It contains a leaky buffer policer which uses an algorithm similar to the ITU-T I.371 Generic Cell Rate Algorithm (GCRA). A leaky buffer is used to shape the traffic going into the switch. Figure 4.1 below shows where shaper is located in a real network. In the ingress direction, traffic is going from physical layer device (PHY) to switch core. A PHY-to-ATM layer device would shape the traffic so that the bandwidth is
smoothed out. For example, in figure 4.1, a connection is sending traffic which peaks at 155Mbps. When it passes through the PHY-to-ATM device, it will be shaped (or policed) by the traffic shaper to an average 34Mbps. In addition to traffic shaping, a PHY-to-ATM device usually includes functions like header translation, performance monitoring (PM) and operation and administration maintenance (OAM). In egress direction, traffic is coming out from switch core back to PHY device. An ATM traffic manager is used to manage the traffic. This is where our scheduler is located. Since scheduling is done with a Traffic Manager, the ATM-to-PHY device usually does not need traffic shaping functions [40].

Figure 4.1 Traffic shaper in ATM networks

The implementation of the whole GCRA is fairly complicated. As a result, this thesis uses a simplified version of GCRA which can still provide a decent policing capability. According to
the current ATM Forum Traffic Management Specification Version 4.0 [4], a rt-VBR connection is characterized by three traffic parameters, Peak Cell Rate (PCR), Sustained Cell Rate (SCR) and Maximum Burst Size (MBS). Given the three traffic parameters, PCR, SCR, MBS, three traffic control functions are derived [39]:

(1) traffic constraint function, $f(t)$

$$f(t) = MBS + SCR \cdot t$$  \hspace{1cm} (23)

Equation (23) states that the maximum number of cells generated by a conforming rt-VBR source during any time interval $t$ has to be less than $f(t)$.

(2) The minimum time interval $I$ between maximum size bursts of cells

$$I = MBS \cdot \left( \frac{1}{SCR} - \frac{1}{PCR} \right)$$  \hspace{1cm} (24)

(3) The maximum earliness $\tau$ of a cell

$$\tau = (MBS - 1) \cdot \left( \frac{1}{SCR} - \frac{1}{PCR} \right)$$  \hspace{1cm} (25)

which is the time a user may send traffic bursts beyond the average rate. This is also called the burst tolerance (BT). Note that cells from different connections are multiplexed at each switch. Traffic will be distorted if rate restoration is not done at each switch. Two more parameters are introduced in modeling the rt-VBR source traffic, the first is

$$BS = \text{current burst size}$$

which varies randomly with uniform distribution between 1 and MBS. The second parameter is

$$IS = \text{idle slots}$$
which is the number of idle slots separating consecutive bursts. From these three traffic control functions, the number of idle slots between burst size is related as follow:

\[ IS = I \times \frac{BS}{MBS} \]

where \( I \) is the minimum time interval between maximum size cell bursts.

The above equation suggests that at current burst size equals to maximum burst size, a connection has to be idled for \( I \) time units, which is calculated by the traffic functions. With the burst size and idle time, a cell stream is formed for a connection. For example, a connection has a maximum burst size of 10 cells, a sustained cell rate = 10 cells/s and a peak cell rate = 100 cells/s, then if this connection bursts out 10 cells, it will have to be idled for a period of:

\[ I = MBS \times \left( \frac{1}{SCR} - \frac{1}{PCR} \right) = 10 \text{ cells} \times \left( \frac{1}{10 \text{ cells/s}} - \frac{1}{100 \text{ cells/s}} \right) = 0.9s = 90 \text{ cell slots} \]

### 4.2 Topologies of Output Queue Management Model

After shaping, the generated traffic for each connection is fed to an output-buffered ATM switch, where the cells are routed to their destination links. This section will introduce the model of an output queue manager and the memory structure used in our work.

Figure 4.2 shows a simple architecture of the ATM switch. When cells arrive at the switch, they will be looked up in the VC table which was constructed during initial connection setup. If the cells are provisioned, then they will be switched out according to the VC table.
Otherwise, cells that are not provisioned will be dropped out for diagnosis or just be thrown away depending on the settings of the switch. Since this thesis concentrates on scheduling, details of the switching architecture are not considered. It is assumed that the switch fabric is non-blocking, and it routes each output queue to receive one cell at most in each time slot. After the switching process, cells destined to the same output link are multiplexed together and are passed to the output queue manager. Each of these output queue managers consists of an output memory buffer, a queue management unit and a scheduler to process the outgoing cells. These output queue managers are responsible for storing the cells waiting for the link and arranging them for transmission in the proper order. In the following sub-sections, the implementation and operation of the output memory buffer, queue management unit and the scheduler will be introduced.

4.2.1 Output Memory Buffer

The output memory is used to store the cells and their associated local information, such as a time-stamp and an address pointer. In this thesis, the memory being modeled is a
synchronous dual-port RAM [41]. The use of synchronous RAM (SRAM) ensures precise cycle control. The dual port feature allows the scheduler to do read and write operations at the same time. Figure 4.3 shows the block diagram of the memory employed in our scheduler.

<table>
<thead>
<tr>
<th>Input/Output Pins</th>
<th>Pin Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK, CLK2</td>
<td>Input Clock to memory. Upon the rising edge, an access cycle begins.</td>
</tr>
<tr>
<td>CEB, CEB2</td>
<td>Clock Enable. When CEB(2) is low, the active edge of CLK(2) is enabled and either a read or write cycle is performed. When CEB is high, the active edge of CLK is ignored and no read or write can be performed.</td>
</tr>
<tr>
<td>RWB, RWB2</td>
<td>Read Write Bar. When RWB(2) is high, read is selected. When RWB is low, write is selected.</td>
</tr>
<tr>
<td>OEB, OEB2</td>
<td>Output Enable Bar. It controls the output drivers from driven to tri-state condition. If enabled during a write access, the output will reflect the data currently being written.</td>
</tr>
<tr>
<td>ADDR, ADDR2</td>
<td>Address Bus [7 downto 0]. It selects the location to be accessed. The 8 bit wide address can access up to 256 memory locations.</td>
</tr>
<tr>
<td>DIN, DIN2</td>
<td>Data In Bus [15 downto 0]. The Word value on this bus is written to the SRAM location in a write access.</td>
</tr>
<tr>
<td>DOUT, DOUT2</td>
<td>Data Out Bus [15 downto 0]. During a read access, when OEB(2) is enabled, the stored data will appear at the DOUT(2) ports. During a write access, when OEB(2) is enabled, DIN(2) will appear at the DOUT bus.</td>
</tr>
</tbody>
</table>

Figure 4.3 Block Diagram of Dual-port Memory
Figure 4.4 below shows a simple timing diagram of a read and write access to the memory block. During a read access, CEB, address bus and RWB are asserted before the CLK goes high. At the first rising edge of CLK, the ADDR value is latched into the memory. On the other hand, during a write access, DIN, CEB, ADDR, RWB have to be asserted before the CLK. OEB can either be held high (for tri-state DOUT bus) or held low (for outputting DIN on DOUT bus). Also shown in figure 4.4 are the setup and hold time used in our simulation.

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CEB Setup Time</td>
<td>1ns</td>
</tr>
<tr>
<td>2</td>
<td>CEB Hold Time</td>
<td>2ns</td>
</tr>
<tr>
<td>3</td>
<td>ADDR Setup Time</td>
<td>1ns</td>
</tr>
<tr>
<td>4</td>
<td>ADDR Hold Time</td>
<td>2ns</td>
</tr>
<tr>
<td>5</td>
<td>RWB Setup Time</td>
<td>1ns</td>
</tr>
<tr>
<td>6</td>
<td>RWB Hold Time</td>
<td>2ns</td>
</tr>
<tr>
<td>7</td>
<td>OEB Tri-&gt; Active</td>
<td>2ns</td>
</tr>
<tr>
<td>8</td>
<td>Read Access</td>
<td>5ns</td>
</tr>
<tr>
<td>9</td>
<td>OEB Active-&gt; Tri</td>
<td>2ns</td>
</tr>
<tr>
<td>10</td>
<td>DIN Setup Time</td>
<td>2ns</td>
</tr>
<tr>
<td>11</td>
<td>DIN Hold Time</td>
<td>4ns</td>
</tr>
</tbody>
</table>

Figure 4.4 Timing Diagram of memory block
As discussed in chapter 2, output buffering has the advantage of achieving optimal bandwidth. Output buffering is chosen because it does not suffer from the head-of-line blocking problem that plagues input buffering. Figure 4.4 below shows the worst case routing when all input ports are transmitting cells simultaneously to the same output port. If input buffering is used in this case, cells will have to wait in the input buffers until cells in front of them are transmitted. Moreover, an input buffered switch also needs an arbitrator to choose which queue is to be served when more than one HOL cell is destined to the same output.

![Figure 4.4 Worst Case for Routing](image)

Despite output buffering can achieve optimal bandwidth, it exerts poor utilization of bandwidth as buffers are exclusively devoted to each output. Cells destined to some outputs may be discarded if the buffers of their outputs are full even if buffers of other outputs have vacancies [42]. In our output-buffered switch model, we employ a per-VC-queue shared memory. The buffer is basically a shared synchronous memory, which is logically organized into VC queues by the queue manager. This approach not only provides better memory utilization, it also provides constant-time access to each VC by the scheduler.

In order to satisfy the delay bounds from (19) to (22), the size of the memory has to be at least:
Number of connections * (\(\sigma_i + h * L_{max}\)) \hspace{1cm} (26)

where \(\sigma_i\) is the burst size of connection \(i\), \(L_{max}\) is the maximum size of the packet and \(h\) is the number of switches the cell traverses through.

Equation (26) states that even during the worst case when maximum sized burst comes from each connection, the memory size specified can still be able to store all of these cells. We are assuming that cell loss cannot occur in the buffer because the traffic has been shaped (using a leaky-bucket algorithm [12]) and policed.

4.2.2 Queue Management Unit

The main functions of a queue manager at each output port of an ATM output-buffered switch, as shown in figure 4.2, are service scheduling and buffer management. As discussed in the last section, the cell buffer is divided logically into VC queues. Each VC queue is implemented as a linked list. The queue management unit is responsible for maintaining these linked lists. Figure 4.5 shows the architecture of the output queue manager, which consists of the memory buffer, queue management unit and the scheduler. The queue management unit contains a current queue occupancy list, which stores the idle address of the memory unit. Besides the queue occupancy list, the queue manager also keeps track of the head, tail, cell count, VC number and the smallest timestamp for each queue. When a cell arrives, it is registered with the queue manager and scheduler by passing the cell header to them. The cell payload, (or cell body), is stored in the buffer memory. The address of the stored cell body will be provided by the queue occupancy list. The cell header will then be appended to the tail of the corresponding queue.
When a cell leaves, the head of the respective queue is removed, and the address is recycled to the queue occupancy list. Both read and write operations require $O(1)$ time. The smallest timestamp is just the timestamp for the head of line cell. The scheduler chooses which cell to be transmitted. For our simulation purposes, the queue manager also maintains a local counter to keep track of the delay of each cell. In the next section, detailed implementations and operations of the schedulers will be discussed.

![Figure 4.5 Architecture of Output Queue Manager](image)

### 4.3 Hardware Implementations and Operations of Schedulers

The basic functionality of a scheduler is to decide the order in which cells are served. The scheduling algorithm has to be efficient, protective and simple to implement. As discussed in chapter 3, Self-Clocked Fair Queuing (SCFQ) achieves near optimal fair queuing performance and its' implementation complexity is low compared to other rate-based methods. SCFQ algorithm has been extensively studied by many researchers [29] [30] [34]. However, the
The hardware implementation of SCFQ has not been addressed in depth in these literatures. Most of the previous studies used a software simulation approach. This thesis focuses using a hardware description language, like VHDL, not only to simulate and evaluate the performance of the scheduler, but also to investigate the feasibility of implementation. In addition to SCFQ, two variations based on the SCFQ algorithm will also be introduced in this section. One algorithm combines SCFQ with delay-EDD while the other one is a combination of SCFQ and jitter-EDD.

4.3.1 Implementations and Operations of SCFQ

Recall from section 3 that SCFQ operates as follow:

a) Compute the tag, $F_j^k$, for the received packet according to equation (9)
b) Store the packet in the FIFO queue belonging to session $j$.
c) Transmit the packet corresponding to the smallest tag from the sequencer.
d) Store the smallest tag value which is used for the computation of tags of subsequently arriving packets.

Figure 4.6 shows a block diagram of the scheduler. It consists of the following blocks: Input Bus Interface, Output Bus Interface, Search Engine, Timestamp Calculation, Per-VC Statistics, Memory Interface and Output Selection. Just as other timestamping scheduling disciplines require sorting, so does SCFQ. Sorting can be done efficiently by an external sorting hardware, such as the sequencer described in [43]. The sequencer operates in $O(1)$ time. Also shown in figure 4.6 are the major input and output pins of the scheduler, sequencer and queue management unit.

The Pin Descriptions are shown as follow:
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scheduler (S) - Switch Controller (SC)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSOC</td>
<td>Input from SC</td>
<td>Receive Start of Cell. It identifies the first word of a cell.</td>
</tr>
<tr>
<td>RCA</td>
<td>Input from SC</td>
<td>Receive Cell Available. It signifies that a cell is ready to transmit into S.</td>
</tr>
<tr>
<td>RDENB</td>
<td>Input from SC</td>
<td>Read Enable. It starts the cell transmission from SC to S.</td>
</tr>
<tr>
<td>TSOC</td>
<td>Output to SC</td>
<td>Transmit Start of Cell. It identifies the first word of a cell.</td>
</tr>
<tr>
<td>WRENB</td>
<td>Output to SC</td>
<td>Write Enable. It starts the cell transmission from S to SC.</td>
</tr>
<tr>
<td>TCA</td>
<td>Input from SC</td>
<td>Transmit Cell Available. It signifies that a cell is ready to transmit out of S.</td>
</tr>
<tr>
<td><strong>Scheduler (S) - Queue Management Unit (QM)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CycleCount</td>
<td>Input from QM</td>
<td>Cycle Count. It counts the number of cycles that the system has been running.</td>
</tr>
<tr>
<td>VC_read</td>
<td>Output to QM</td>
<td>VC Read. It signifies to QM that a VC has to be read out from it.</td>
</tr>
<tr>
<td>VC_out</td>
<td>Output to QM</td>
<td>VC Out. It identifies which VC has to be read out.</td>
</tr>
<tr>
<td>VC_write</td>
<td>Output to QM</td>
<td>VC Write. It signifies to QM that a cell has to be written into QM.</td>
</tr>
<tr>
<td>VC_record</td>
<td>Output to QM</td>
<td>VC Record. It provides the information of the current cell written to QM.</td>
</tr>
<tr>
<td>dataIN</td>
<td>Input from SC to QM</td>
<td>Data In Bus. It carries the cell coming from SC to QM.</td>
</tr>
<tr>
<td>dataOUT</td>
<td>Output from QM to SC</td>
<td>Data Out Bus. It carries the cell going out from QM to SC.</td>
</tr>
<tr>
<td>clk_count</td>
<td>Input from SC to QM</td>
<td>Clock Counter. It provides the number of cycles for delay calculations.</td>
</tr>
<tr>
<td>VC_delay</td>
<td>Output from QM to SC</td>
<td>VC Delay Time. It provides the SC the delay time of cell due to scheduling.</td>
</tr>
<tr>
<td><strong>Scheduler (S) - Sequencer (SEQ)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sch_mode</td>
<td>Output to SEQ</td>
<td>Scheduler mode. It identifies whether S is running SCFQ, DEDD or JEDD.</td>
</tr>
<tr>
<td>low_tag_VC</td>
<td>Input from SEQ</td>
<td>Lowest Tag VC. The VC number that has the lowest tag.</td>
</tr>
<tr>
<td>lowest_tag</td>
<td>Input from SEQ</td>
<td>Lowest Tag. The tag of low_tag_VC which is used to calculate the timestamps.</td>
</tr>
<tr>
<td>low_tag_list</td>
<td>Output to SEQ</td>
<td>Lowest Tag List. The list that contains the lowest tag for each connection.</td>
</tr>
<tr>
<td><strong>Miscellaneous</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSTB</td>
<td>Input to all blocks</td>
<td>Global Reset. It resets the S, SEQ and QM.</td>
</tr>
<tr>
<td>CLK</td>
<td>Input to all blocks</td>
<td>System Clock. It clocks the S, SEQ, and QM.</td>
</tr>
</tbody>
</table>

![Block Diagram of Scheduler](image)

Figure 4.6 Block Diagram of Scheduler
The scheduler is triggered by clock signal, CLK. Cells flow in from the receive side and go out to the transmit side. The major input/output signals are listed in the diagram as well. Each block works as follow:

Input Interface Block

The input bus interface block is responsible for the transfer of cells. It has an interface similar to that of the ATM Forum’s Utopia Level 1 interface [44]. The scheduler checks whether the Cell Available (CA) from the switch controller is asserted in every CLK cycle. When CA is asserted and the scheduler can accept the cell, it asserts the Write Enable (WREN) signal to signify that it is ready to accept the cell. The switch controller then asserts the Start of Cell (SOC) signal and puts the data on the dataIN bus. In the meantime, the scheduler starts to read in the first three words, which constitutes the header of the cell, starting from the SOC position. After the header is latched in, the scheduler de-asserts the WREN, and then passes it to the Search Engine Block.

Search Engine Block

The responsibility of the search engine block is to gather per-VC information for the current cell from the queue management unit. It accesses this per-VC information through the memory control interface. When a cell arrives at this block, it will first be checked to see if the header is valid, that is whether it is provisioned or not. If it is provisioned, the search engine gathers some local information, such as the current cell count, current finish number and last finish number of connection, and passes them to other blocks for processing.
Timestamp Calculation Block

From the search engine block, the timestamp calculation block gathers the cell information and calculates the finish number of the cell according to the following equation (see chapter 3):

\[ F_j^k = \frac{L_j}{r_j} + \max\{F_j^{k-1}, v(a_j^k)\} \]  \hspace{1cm} (9)

The calculations involve one division, one addition and one comparison. After the finish number is calculated, it is appended to the cell and passed to the per-VC statistics block and the memory control block.

Per-VC Statistics Block

The per-VC statistics block keeps track of the delay time for each cell. It has a 32-bit local counter to serve as a local clock. This local counter increments at every CLK cycle. Whenever a cell is output, it subtracts the original cycle count from this local counter to calculate the delay time for the cell.

Memory Interface Block

The memory interface block is responsible of reading and writing from/to the memory buffer. Its operation has already been discussed in section 4.2.1.
**Output Selection Block**

The output search engine block selects the VC to be served. The sequencer keeps a list of finish numbers for each VC queue. During a cell transmission, the output selection block reads the smallest finish number and its associated VC number from the sequencer. Then it sets the current finish number to this smallest finish number and passes them to the output interface block and memory interface block.

**Output Interface Block**

The output bus interface block is similar to that of the input interface block. When a cell is ready to be transmitted, the output interface block asserts the Cell Available Out (CAout) signal. When the switch controller is ready, it responses by assert its' Read Enable (RDENB). As soon as the output interface sees this RdENB, it asserts a local signal to request the queue manager to start sending the cell. The cell transmission starts from the Start of Cell Out (SOCout) position. When the last word of the cell has been sent, the switch controller de-asserts the RDENB signal to end the transmission.

### 4.3.2 Hardware Implementations and Operations of DEDD-SCFQ and JEDD-SCFQ

Although SCFQ provides a delay bound, it suffers from the coupling problem between delay and bandwidth. A low bandwidth VC has a high delay bound in SCFQ. In addition, since SCFQ is a work-conserving algorithm, it cannot provide a non-trivial jitter bound. The delay-jitter bound of SCFQ is just its maximum delay bound. Thus, it is desirable to have a scheduling discipline which provides fair bandwidth allocation as SCFQ, while decoupling the bandwidth
and delay for VCs and provides them with non-trivial jitter bounds. In this thesis, two models are developed, one of them combines SCFQ and delay-EDD, while the other combines SCFQ and jitter-EDD. The analytical and simulation results of these two disciplines will be presented in chapter 5. The following sections will introduce the implementations and operations of these two algorithms.

4.3.2.1 DEDD-SCFQ

The main idea of DEDD-SCFQ is to provide not only a guaranteed bandwidth, but also a delay bound to some connections, especially for those low bandwidth VCs requiring a low delay. DEDD-SCFQ algorithm divides VCs into two groups, which are called SCFQ VCs (or SCFQ connections) and DEDD VCs (DEDD connections) respectively. DEDD VCs require tight delay bound even at low bandwidth allocation, while SCFQ VCs require some kinds of delay bounds, but not as strict as DEDD VCs. The operations of DEDD-SCFQ algorithm are similar to that of pure SCFQ. It works as follow:

a) During connection setup, register the connections which need to be tightly delay bounded (DEDD connections). Specify the delay bounds required for such connections.

b) When a cell arrives, compute the tag, \( F_j \), as in SCFQ, according to equation (9). For DEDD connections, also compute a threshold count. This threshold count stores when this cell is eligible for transmission in order to satisfy the delay requirement.

c) Store the cell along with the threshold count in the FIFO queue belonging to class \( j \).

d) Before selecting the lowest tag as in SCFQ, check the DEDD bounded connections. If there is any cell from these DEDD connections reaching its threshold, blocks the lowest tag SCFQ cell and transmit the DEDD cell first. If there is no DEDD cell reaching its threshold, transmit cells as in SCFQ.
e) Transmit either the DEDD cell or the SCFQ cell corresponding to the lowest tag.
f) Store the lowest tag value which is used for the computation of tags of subsequently arriving packets.

Besides the implementation costs to SCFQ, DEDD-SCFQ requires more memory spaces for two lists. The first list keeps track of the DEDD VC number, while the second list records the threshold count for its corresponding VC. Another sequencer has to be implemented in order to search through the DEDD connections. Besides that, in the timestamp calculation block, one addition, one subtraction and a comparison operation are required in addition to SCFQ computation. The effects of adding the DEDD to SCFQ will be analyzed in chapter 5.

4.3.2.2 JEDD-SCFQ

Although DEDD-SCFQ can provide non-trivial delay bounds to the traffic, it would be better if a scheduler can also provide non-trivial jitter bounds as well. JEDD-SCFQ provides jitter bounds to VCs which require non-trivial jitter bound. Similar to DEDD-SCFQ, JEDD-SCFQ algorithm divides VCs into two groups, namely JEDD VCs and SCFQ VCs. JEDD VCs require a tight jitter delay bound, while the delay and jitter bound of SCFQ VCs are boounded by pure SCFQ algorithm. The operations of DEDD-SCFQ are similar to that of DEDD-SCFQ. It works as follow:

a) During connection setup, register the connections which need to be jitter-delay bounded (JEDD connections). Specify the minimum and maximum delay required for such connections.
b) When a cell arrives, compute the tag, $F^j_x$, as in SCFQ, according to equation (9). For JEDD connections, also compute an eligibility count which is stored in the eligibility list and a threshold count which is stored in threshold list. The eligibility count is used to regulate the cell until it is eligible to be transmitted. This threshold count stores when this cell is required to transmit in order to satisfy the delay requirement.

c) Store the cell along with these two counts in the FIFO queue belonging to session $j$.

d) Before selecting for the lowest tag as in SCFQ, check the JEDD bounded connections. If there is any cell from these JEDD connections reaching its threshold and it is eligible for transmission, then blocks the lowest tag SCFQ cell and transmit the JEDD cell first. If there is no JEDD cell reaching its threshold or is eligible, transmit cells as in SCFQ.

e) Transmit either the JEDD cell or the SCFQ cell corresponding to the lowest tag.

f) Store the lowest tag value which is used for the computation of tags of subsequently arriving packets.

The implementations include both parts in SCFQ and SCFQ-DEDD. Similar to a jitter-delay regulator, the JEDD-SCFQ maintains a minimum delay before the cells are transmitted. The overhead of JEDD-SCFQ includes maintaining an eligibility list. Another sequencer has to be implemented in order to search through the JEDD eligibility list. The effects of adding the JEDD to SCFQ will be analyzed in chapter 5.

### 4.4 Implementation Summary

All three schedulers are implemented in VHDL, so that it can be shown that they are implementable algorithms. Using 0.25 micron BICMOS technology, the size and speed of these schedulers are found. If the link is an OC12, the cell slot time is about 700ns. Since the SCFQ-
JEDD scheduler works at speed faster than 50MHz, it provides a sufficient operation margin. By using more advanced technology, the speed and area can be improved.
Chapter 5 Simulation Results

Despite the fact that schedule-based scheduling algorithms can provide end-to-end and jitter delay bounds which are easy to compute, rate-based algorithms are still more favourable by most of the switch vendors. The main drawback of schedule-based algorithms is that they must reserve bandwidth at peak rate instead of at the average rate. If all connections in a network reserve bandwidth at their peak rate, it is likely that either quality-of-service guarantee cannot be provided to all connections or only a smaller number of connections can be admitted. In contrast, rate-based methods reserve bandwidth at the average rate instead of peak rate. This allows the network to accept more connections while providing quality-of-service. Among rate-based algorithms, Self-Clocked Fair Queuing is chosen for investigation because it does not require complicated round-number computation and its performance is very close to PGPS.

This chapter discusses the performance of Self-Clocked Fair Queuing scheduling algorithm and two variations that we have developed in this thesis. The first one combines Delay-Earliest-Due-Date (DEDD) and SCFQ, while the second one combines Jitter-Earliest-Due-Date (JEDD) and SCFQ. The performance of these algorithms has been extensively studied by simulation under various traffic patterns. In the next section, the simulation model used in the performance evaluation is described. The topology of the simulated network along with traffic parameters will be introduced. Then in the following sections, both analytical and simulation results of the above algorithms will be discussed.
5.1 Network Topology

Figure 5.1 shows the network topology used in our simulations. It consists of three traffic sources (S1, S2, and S3) and generic ATM switches (SW1, SW2, and SW3). Each traffic source sends out ATM packets at a rate of 155Mbps. At the input of each switch, there is a traffic shaper (SH1, SH2, and SH3). Before cells going into the core switch, the traffic shaper has to make sure that the traffic obeys the parameters as discussed in section 4.1. Cells will then be routed by the core switch and passed to the output scheduler.

In our simulation, S1 acts as the traffic source in the network. It consists of eight virtual connections (VC1 to VC8). S2 and S3 model aggregated sources that create cross traffic flowing into switches SW2 and SW3. It should be noted that S2 and S3 are meant to create additional traffic to compete with traffic from source S1. SW1 can be viewed as an access switch which aggregates the VCs from different traffic sources. In reality, commercial switches multiplex as
many as 64,000 VCs on each link. For practical purposes, the number of VCs per link in our simulation is scaled down to 8 VCs. In addition, we found out that if the traffic is properly policed, the performance of the schedulers can be studied by just looking at the single node model. For each simulation, a number of parameters need to be specified. These parameters include source traffic parameters and scheduler parameters which are shown in table 5.1 below.

<table>
<thead>
<tr>
<th>Traffic Parameter</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak Cell Rate (PCR)</td>
<td>in specifying VBR and CBR traffic</td>
</tr>
<tr>
<td>Sustain Cell Rate</td>
<td>in specifying VBR traffic</td>
</tr>
<tr>
<td>Maximum Burst Size</td>
<td>in specifying VBR traffic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scheduler Parameters</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth Distribution</td>
<td>Used by SCFQ, DEDD-SCFQ and JEDD-SCFQ to allocate bandwidth</td>
</tr>
<tr>
<td>Delay Bound</td>
<td>Used by DEDD-SCFQ to set maximum delay</td>
</tr>
<tr>
<td>Jitter Bound</td>
<td>Used by JEDD-SCFQ to set maximum jitter</td>
</tr>
</tbody>
</table>

Table 5.1 Traffic and Scheduler Parameters

5.2 Analytical Model

This section will present the analytical model using the network model discussed in section 5.1. All three scheduling algorithms discussed in chapter 4 will be analyzed. These analytical results will be used to compare with the actual simulation results. Recall from Chapter 2.1 that in order for a scheduling algorithm to guarantee quality-of-service for real-time traffic, it has to provide bandwidth guarantees, delay bound and delay-jitter bound [45]. Self-Clocked Fair Queuing (SCFQ) scheduling algorithm not only provides these bounds, but it also posses a simple implementation complexity. As a result, SCFQ were used as a foundation for building our DEDD-SCFQ and JEDD-SCFQ schedulers. For evaluations of these algorithms, we will look at the maximum delay bound and maximum delay-jitter bound. It should be noted that there are other parameters, such as buffer size and implementation overhead, that are important to a
scheduling algorithm. As far as real-time traffic is concerned, the delay bound and delay-jitter bound are more important than others.

In chapter 3, the delay bound and the delay-jitter bound of SCFQ were expressed by the following equations - equations (16) and (20):

\[ D_{i}^{SCFQ} \leq \frac{\sigma_i + (K - 1)L_{\text{max}}}{\rho_i} + \sum_{j=1}^{K} N_j L_{\text{max}} \frac{r_j}{r} \]  
\[ J_{i}^{SCFQ} \leq \frac{\sigma_i + (K - 1)L_{\text{max}}}{\rho_i} + \sum_{j=1}^{K} (N_j - 1) L_{\text{max}} \frac{r_j}{r} \]  

where \( \sigma_i \) is the burst size, \( K \) is the number of hops traversed by session \( i \), \( \rho_i \) is the guaranteed rate for the connection, \( L_{\text{max}} \) is the maximum size of the packet and \( r \) is the link speed.

The end-to-end delay bound of SCFQ is given in equation (16). In [24], Parekh and Gallager have proved the delay bound for the ideal GPS is \( \frac{\sigma_i}{\rho_i} \). In other words, in the ideal GPS algorithm, the delay bound is just the maximum burst size divided by the connection's allocated rate. In reality, schedulers are not cut-through devices, a packet can be serviced only after its last bit has arrived into the buffer. Thus, GPS is not an implementable algorithm. As a result, the delay bounds of real schedulers are much larger than that of GPS.

Assume that a session \( i \) becomes backlogged by the arrival of the \( kth \) packet. Consider the \( n \)th packet of the same backlogged period of session \( i \), that is the \((k+n-1)th\) packet of session \( i \). [46]. To calculate the delay bound, we need to consider the worst case scenario where all other
sessions have a packet to send just before session \( i \). Moreover, every other connection can transmit a maximum of \( \frac{nL_i}{\rho_j} \) traffic, \( \rho_i \) and \( \rho_j \) are the guaranteed rates for the connections and \( L_i \) is the size of the connection \( i \). Thus the time at which this \( nth \) packet will be transmitted after the following delays:

\[
D_i^{SCFQ} \leq \frac{\sigma_i}{\rho} + (N - 1) \frac{L_{max}}{r} + \frac{1}{r} \left( \sum_{j=k}^{k+\pi-1} L_i + \sum_{j=1}^{N} n \frac{L_i}{\rho_i} \rho_j \right)
\]

\[
\leq \frac{\sigma_i}{\rho} + (N - 1) \frac{L_{max}}{r} + \frac{1}{r} \left( nL_i + n \frac{(r - \rho_i)}{\rho_i} L_i \right)
\]

\[
\leq \frac{\sigma_i}{\rho} + (N - 1) \frac{L_{max}}{r} + \frac{n L_i}{\rho_i}
\]

If the packet travels through \( k \) SCFQ schedulers, then

\[
D_i^{SCFQ} \leq \frac{\sigma_i + (K - 1)nL_{max}}{\rho} + \sum_{i=1}^{k} nN_i \frac{L_i}{r^i}
\]

For ATM system, all packets have same packet size, \( L_{max} \), and for \( n=1 \), the delay bound becomes the same as (16):

\[
D_i^{SCFQ} \leq \frac{\sigma_i + (K - 1)L_{max}}{\rho_i} + \sum_{i=1}^{K} N_i \frac{L_{max}}{r^i}
\]

The delay-jitter bound of SCFQ is just the maximum delay bound.

As introduced in chapter 4, DEDD-SCFQ and JEDD-SCFQ are two algorithms which add delay-EDD and jitter-EDD scheduling into SCFQ to provide some connections with shorter
delay bound and delay-jitter bounds. For DEDD-SCFQ and JEDD-SCFQ, the delay bounds are given by:

\[ D^\text{DEDD-SCFQ}_i = J^\text{DEDD-SCFQ}_i \leq (N^\text{DB}_D - 1) \times MBS \] (23)

\[ D^\text{JEDD-SCFQ}_i = J^\text{JEDD-SCFQ}_i \leq (N^\text{JDB}_J - 1) \times MBS \] (24)

where \( N^\text{DB}_D \) and \( N^\text{JDB}_J \) are the number of delay bounded connections in DEDD-SCFQ and JEDD-SCFQ, respectively, and MBS is the maximum burst size.

Equation (23) and (24) specify that for delay-bounded VCs, the minimum delay bound and delay-jitter bound cannot be smaller than \( (N^\text{DB}_D - 1) \times MBS \). For example, if number of delay bounded connections is equal to 2 and MBS is 10 cells, then the best delay bound that the system can provide is 20 cell slots. The delay-jitter bounds of these two algorithms are equal to their delay bound. However, in JEDD-SCFQ, the delay-jitter bound is used to specify a delay range. Take another example, in JEDD-SCFQ, if the number of delay bounded connections is equal to 2, MBS is 10 and the minimum delay requirement is 20 cell slots; then the maximum delay is \( 20 + (N^\text{DB}_D - 1) \times MBS = 30 \) cell slots. The delay bound range provided by JEDD-SCFQ in this case is 20 to 30 cell slots. So, in the simulation, we have to set minimum delay to 20 and maximum delay to 30 for this scenario. Figure 5.2 below shows this scenario graphically. VC1 and VC2 are two JEDD-delay bounded connections. The setting of the scheduler is as specified in previous example.
Equation (23) and (24) give the delay and delay-jitter bound for those delay-bounded connections. These bounds are achieved at the expense of other connections which are not JEDD delay-bounded, but are served according to SCFQ. The delay bounds for the SCFQ connections are:

\[
D_{i}^{\text{non-DEDD-SCFQ}} = J_{i}^{\text{non-DEDD-SCFQ}} \leq D_{i}^{\text{SCFQ}} + (N_{DB} \times MBS) \quad (25)
\]

\[
D_{i}^{\text{non-JEDD-SCFQ}} = J_{i}^{\text{non-JEDD-SCFQ}} \leq D_{i}^{\text{SCFQ}} + (N_{DB} \times MBS) \quad (26)
\]

Equation (25) and (26) specify that the delay bound of those non-DEDD/JEDD-delay bounded connections is penalized at worst by \((N_{DB} \times MBS)\). This maximum value is achieved only when all delay bounded connections send traffic at maximum burst at the same time.

In the following sections, the actual simulation results will be given. These results will be compared with the analytical ones which are calculated using the above equations.
5.3 Simulation Results

In order to study the performance of the schedulers, six different scenarios are considered:

1. Effect of burst size on SCFQ
2. Delay bound and delay-jitter bound of SCFQ
3. Delay bound and delay-jitter bound of DEDD-SCFQ
4. Delay and delay-jitter bound of JEDD-SCFQ
5. Impact of delay-bounded connections on non-delay-bounded connections in JEDD-SCFQ
6. Impact on number of delay-bounded connections in JEDD-SCFQ

5.3.1 Effect of burst size on SCFQ

In this first case, the effect of traffic burst size on SCFQ was analyzed. The maximum burst size (MBS) was varied from 10 to 55. Table 5.2 shows the bandwidth share allocated to each VC. In each case, the source sent out bursts ranging randomly from 1 cell to MBS cells. The maximum delay and average delay were taken.

<table>
<thead>
<tr>
<th>VC Number</th>
<th>Bandwidth Share (SCR)</th>
<th>Miscellaneous</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC1</td>
<td>25%</td>
<td></td>
</tr>
<tr>
<td>VC2</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>VC3</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>VC4</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>VC5</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>VC6</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>VC7</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>VC8</td>
<td>10%</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2 Bandwidth Allocation in Scenario 1
Figure 5.3 below shows the maximum and average delay versus the maximum burst size of VC1. As shown in the figure, the larger the maximum burst size, the higher the delay. In equation (16), the maximum size of a burst is one of the variables in calculating delay bound and delay-jitter bound. The maximum burst size contributes the major part of the delay bound. The term $\frac{\sigma}{\rho}$ in equation (16), which is the maximum burst size divided by the bandwidth allocated, dominates the delay bound. This is one of the characteristics of PGPS-based scheduling algorithms. As a result, when implementing a scheduling algorithm, the maximum burst size has to be known appropriately.

![Maximum and Average Delay VS Maximum Burst Size](image)

**Figure 5.3 Delay vs Maximum Burst Size for VC1**

### 5.3.2 Delay Bound and delay-jitter bound of SCFQ

In this case, the delay and delay-jitter bound of SCFQ were analyzed. The maximum burst size (MBS) was set to 10 cells. The bandwidth of VC1 was varied from 5% to 80% of the total bandwidth, while VC2 to VC8 shared the remaining bandwidth. In each case, the source
sent out traffic burst ranging randomly from 1 cell to 10 cells. The maximum delay and average delay of VC1 were taken. Table 5.3 summarizes the bandwidth allocations for the VCs.

As discussed in chapters 3 and 4, SCFQ does not require complicated finish-number calculation, it can also provide bandwidth, delay and delay-jitter bounds. Figure 5.3 shows the maximum delay and average delay provided by SCFQ.

<table>
<thead>
<tr>
<th>VC Number</th>
<th>Bandwidth Share (SCR)</th>
<th>Miscellaneous</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC1</td>
<td>5% to 80%</td>
<td></td>
</tr>
<tr>
<td>VC2</td>
<td>(95%-VC1)/7</td>
<td></td>
</tr>
<tr>
<td>VC3</td>
<td>(95%-VC1)/7</td>
<td></td>
</tr>
<tr>
<td>VC4</td>
<td>(95%-VC1)/7</td>
<td>Burst Size:</td>
</tr>
<tr>
<td>VC5</td>
<td>(95%-VC1)/7</td>
<td>MBS=10</td>
</tr>
<tr>
<td>VC6</td>
<td>(95%-VC1)/7</td>
<td>SCFQ scheduler</td>
</tr>
<tr>
<td>VC7</td>
<td>(95%-VC1)/7</td>
<td></td>
</tr>
<tr>
<td>VC8</td>
<td>(95%-VC1)/7</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3 Bandwidth Allocation for Scenario 2

![Maximum delay and Average Delay VS Bandwidth - SCFQ](image)

Figure 5.4 Maximum and Average Delay vs Bandwidth in SCFQ for VC1
The average delay curve goes down smoothly as the bandwidth was increased. However, the gap between the two curves became wider as the bandwidth was decreased. Specifically, the maximum delay was inversely proportional to the allocated bandwidth. Thus, in order for a connection to get a low delay bound, a high bandwidth channel needs to be allocated. This coupling of bandwidth and maximum delay is not desirable for real-time traffic, as it requires tight delay-jitter bound. In the next simulation case, a variation of SCFQ called DEDD-SCFQ, is studied.

### 5.3.3 Delay bound and delay-jitter bound of DEDD-SCFQ

The DEDD-SCFQ algorithm was introduced in section 4.3 as a technique that combines DEDD and SCFQ. The delay bound of a connection is specified by a threshold count, which signifies when the cell should be transmitted. This section will show the simulation results. The maximum burst size (MBS) was set to 10. The bandwidth of VC1 was varied from 5% to around 80% in different trails. The threshold count (delay bound) was set to 20 so that the maximum delay was bounded at 20 cell slots. VC2 to VC8 shared the remaining bandwidth in each trail. In

<table>
<thead>
<tr>
<th>VC Number</th>
<th>Bandwidth Percentage (SCR)</th>
<th>Miscellaneous</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC1</td>
<td>5% to 80%</td>
<td>Burst size:</td>
</tr>
<tr>
<td>VC2</td>
<td>(95%-VC1)/7</td>
<td>MBS=10</td>
</tr>
<tr>
<td>VC3</td>
<td>(95%-VC1)/7</td>
<td>VC1 = delay-bounded VC</td>
</tr>
<tr>
<td>VC4</td>
<td>(95%-VC1)/7</td>
<td>Delay bound set to 20</td>
</tr>
<tr>
<td>VC5</td>
<td>(95%-VC1)/7</td>
<td>DEDD-SCFQ scheduler</td>
</tr>
<tr>
<td>VC6</td>
<td>(95%-VC1)/7</td>
<td>SCFQ scheduler</td>
</tr>
<tr>
<td>VC7</td>
<td>(95%-VC1)/7</td>
<td></td>
</tr>
<tr>
<td>VC8</td>
<td>(95%-VC1)/7</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4 Bandwidth Allocation for Scenario 3
each case, the source sent out traffic bursts ranging randomly from 1 to 10 cells. This simulation was then run using both pure SCFQ scheduler and DEDD-SCFQ scheduler for comparison. Table 5.4 summarizes the bandwidth allocations for the VCs.

Figure 5.5 shows the maximum delay comparison between SCFQ and DEDD-SCFQ. Since DEDD-SCFQ is based on SCFQ algorithm, they share the same delay characteristics, the delay decreases as the allocated bandwidth increases. However, DEDD-SCFQ provides better delay bounds for low bandwidth VCs. In this simulation, the delay bound was set to 20 cell slots for a connection. As shown in figure 5.6, the maximum delay for DEDD-SCFQ was 20 cell slots, while the maximum delay for SCFQ was 84 cell slots. It is clear that DEDD-SCFQ reduces the delay-jitter which is a significant factor for real-time traffic. The two delay curves of SCFQ and DEDD-SCFQ become very close to each other as the bandwidth allocated to VC1 is
increased to over 30%. In fact, when VC1 bandwidth share is more than 30%, SCFQ would give similar delay bound as DEDD-SCFQ. In other words, for a 20 cell slot delay bound, SCFQ requires a 30% average allocated rate, while DEDD-SCFQ requires a much lower average bandwidth. Therefore, DEDD-SCFQ solves the coupling problem of delay and bandwidth incurred from SCFQ.

5.3.4 Delay bound and delay-jitter bound of JEDD-SCFQ

As introduced in earlier chapters, JEDD-SCFQ is an algorithm which combines the idea of JEDD and SCFQ algorithms. In addition to a threshold count, JEDD-SCFQ also keeps an eligibility count for each JEDD bounded cell. The simulation is run with the following parameters. The maximum burst size (MBS) was set to 10. The bandwidth of VC1 was varied from 5% to around 80% in different trails. The maximum delay bound number was set to 20 while the minimum delay bound was set to 10. The threshold and eligibility counts are calculated within the timestamp calculation block. VC2 to VC8 shared the remaining bandwidth in each trail. In each case, the source sent out traffic burst ranging randomly from 1 cell to 10 cells. The

<table>
<thead>
<tr>
<th>VC Number</th>
<th>Bandwidth Percentage (SCR)</th>
<th>Miscellaneous</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC1</td>
<td>5% to 80%</td>
<td>Burst Size: MBS=10</td>
</tr>
<tr>
<td>VC2</td>
<td>(95%-VC1)/7</td>
<td>VC1 = delay-bounded VC</td>
</tr>
<tr>
<td>VC3</td>
<td>(95%-VC1)/7</td>
<td>Max Delay bound set to 20</td>
</tr>
<tr>
<td>VC4</td>
<td>(95%-VC1)/7</td>
<td>Min Delay bound set to 10</td>
</tr>
<tr>
<td>VC5</td>
<td>(95%-VC1)/7</td>
<td>JEDD-SCFQ scheduler</td>
</tr>
<tr>
<td>VC6</td>
<td>(95%-VC1)/7</td>
<td>DEDD-SCFQ scheduler</td>
</tr>
<tr>
<td>VC7</td>
<td>(95%-VC1)/7</td>
<td>SCFQ scheduler</td>
</tr>
<tr>
<td>VC8</td>
<td>(95%-VC1)/7</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5 Bandwidth allocation for Scenario 4
simulation was then run under pure SCFQ, DEDD-SCFQ and JEDD-SCFQ for comparisons. Table 5.5 summarizes the bandwidth allocations for the VCs.

Figure 5.6 below shows the maximum delay comparisons of the three algorithms, SCFQ, DEDD-SCFQ and JEDD-SCFQ. As discussed in previous section, DEDD-SCFQ provides better delay bounds for low bandwidth VCs. However, as bandwidth increases, DEDD-SCFQ tends to follow the SCFQ delay characteristics to have delay bound decreases sharply. This creates a jitter problem for real-time traffic which requires memory spaces to buffer up these early arrived cells. In this simulation, JEDD-SCFQ is setup to provide a maximum delay bound of 20 as in DEDD-SCFQ, and a minimum delay bound of 10. This significantly reduced the delay-jitter which is a significant factor for real-time traffic. It should be noted that since JEDD and SCFQ are combined, the JEDD-SCFQ scheduler still posses the work-conserving characteristics. Whenever there is not a JEDD connection eligible for transmission, the scheduler will serve those non-JEDD cells in accordance to SCFQ algorithm. The next section will show the effect of this JEDD-SCFQ on SCFQ.

Figure 5.6 Maximum Delay vs Bandwidth of Delay-Bounded VC
5.3.5 Impact of delay-bounded connections on non-delay-bounded connections in JEDD-SCFQ

As shown in the previous section, JEDD-SCFQ tries to provide some connections better delay jitter bounds than that of SCFQ. In order to provide these delay bounds to those delay-bounded connections, other non-delay bounded connections have to scarify. This simulation shows the effect of delay-bounded connections to non-delay-bounded connections. The maximum burst size (MBS) was set to 10. The bandwidth of VC1 and VC2 were set to the same percentage and were varied from 5% to around 50% in different trails. VC1 was the delay-bounded connection, the maximum delay bound number was set to 20 while the minimum delay bound was set to 10. VC2 and VC3 to VC8 were normal connections, that is, without JEDD delay bound. The simulation was then run under pure SCFQ and JEDD-SCFQ for comparisons.

<table>
<thead>
<tr>
<th>VC Number</th>
<th>Bandwidth Percentage (SCR)</th>
<th>Miscellaneous</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC1</td>
<td>5% to 50%</td>
<td>MBS=10</td>
</tr>
<tr>
<td>VC2</td>
<td>5% to 50%</td>
<td>VC1 = delay-bounded VC</td>
</tr>
<tr>
<td>VC3</td>
<td>(95%-2*VC1)/6</td>
<td>Max Delay bound set to 20</td>
</tr>
<tr>
<td>VC4</td>
<td>(95%-2*VC1)/6</td>
<td>Min Delay bound set to 10</td>
</tr>
<tr>
<td>VC5</td>
<td>(95%-2*VC1)/6</td>
<td>VC2 = non-delayed bounded VC</td>
</tr>
<tr>
<td>VC6</td>
<td>(95%-2*VC1)/6</td>
<td>JEDD-SCFQ scheduler</td>
</tr>
<tr>
<td>VC7</td>
<td>(95%-2*VC1)/6</td>
<td>SCFQ scheduler</td>
</tr>
<tr>
<td>VC8</td>
<td>(95%-2*VC1)/6</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.6 Bandwidth Allocation for Scenario 5

Figure 5.7 below shows the maximum delay vs bandwidth for a JEDD-delay bounded connection, a non-JEDD-delay bounded connection and a SCFQ connection. As discussed in previous section, JEDD-SCFQ provides tighter delay-jitter bounds for low bandwidth VCs. In figure 5.7, the JEDD-delay bounded connection has its' maximum delay at 20. On the other hand, the non-JEDD-delay bounded connection has its' maximum delay increased to values higher than
SCFQ. As chapter 5.2 discussed, the delay bound of non-JEDD-delay bounded connection in JEDD-SCFQ is:

\[ D_{i}^{\text{JEDD-SCFQ}} = j_{i}^{\text{JEDD-SCFQ}} \leq (N_{JDB} - 1) \times MBS \]

In this simulation, \((N_{JDB} - 1) \times MBS = 10\) cell slots. As shown in figure 5.6, the maximum difference between SCFQ and JEDD-SCFQ is indeed around 10 cell slots. In addition, as this non-JEDD-delay bounded connection is served under SCFQ scheduling, the delay characteristics follow to that of SCFQ.

![Max Delay vs Bandwidth](image)

Figure 5.7 Maximum Delay vs Bandwidth - delay bounded VC and non-delay bounded VC

5.3.6 Impact on Number of Delay-Bounded Connections in JEDD-SCFQ

In this simulation, the impact on number of delay-bounded connections in JEDD-SCFQ scheduling algorithm is analyzed. As mentioned in the previous section, in order to provide delay bounds to those delay-bounded connections, other non-delay-bounded connections have to scarify. The more the number of JEDD-delay bounded connections, the higher the delay for
other non-JEDD delay bounded connections. This scenario shows this effect by using different number of JEDD-delay bounded connections. Two trails were run in this scenario. The first one with VC1 to VC4 JEDD-delay bounded and the second one with no VC JEDD-delay bounded. The bandwidth of VC1 to VC4 was set to 5%. The maximum delay bound number was set to 40 (by using equation (26)), while the minimum delay bound was set to 10 for them. Bandwidth of VC5 was varied from 5% to 70%, while VC6 to VC8 shared the remaining bandwidth. VC5 to VC8 were not JEDD-delay bounded connections. The maximum burst size (MBS) was set to 10. The simulation was then run under JEDD-SCFQ for trail 1, and pure SCFQ for trail2 for comparisons. Table 5.7 summarizes the bandwidth allocations for the VCs.

<table>
<thead>
<tr>
<th>VC Number</th>
<th>Bandwidth Percentage (SCR)</th>
<th>Miscellaneous</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC1</td>
<td>5%</td>
<td>Trail 1: VC1 to VC4 JEDD-delay bounded Max Delay = 40, Min Delay = 10 JEDD-SCFQ scheduler</td>
</tr>
<tr>
<td>VC2</td>
<td>5%</td>
<td>Trail 2: no JEDD-delay bounded VCs SCFQ scheduler</td>
</tr>
<tr>
<td>VC3</td>
<td>5%</td>
<td>MBS all for cases = 10</td>
</tr>
<tr>
<td>VC4</td>
<td>5%</td>
<td></td>
</tr>
<tr>
<td>VC5</td>
<td>5% to 70%</td>
<td></td>
</tr>
<tr>
<td>VC6</td>
<td>(95%-VC1-VC2-VC3-VC4-VC5)/3</td>
<td></td>
</tr>
<tr>
<td>VC7</td>
<td>(95%-VC1-VC2-VC3-VC4-VC5)/3</td>
<td></td>
</tr>
<tr>
<td>VC8</td>
<td>(95%-VC1-VC2-VC3-VC4-VC5)/3</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.7 Bandwidth Allocation for the Two Trails in Scenario 6

Figure 5.8 below shows the maximum delay and average delay vs bandwidth for the two trails shown in table 5.7. All four curves in figure 5.7 belong to VCs that are not JEDD-delay bounded. When we look at the maximum delay curves, we can clearly see that the maximum delay provided for non-JEDD-delay bounded VCs by JEDD-SCFQ algorithm is higher than that provided by pure SCFQ. According to equation (25) and (26), this difference should be \( N_{DB} \times \text{MBS} \). Indeed, the results of our simulations showed that the delay difference was \( \leq N_{DB} \times \text{MBS} \).
= 4 \times 10 = 40 \text{ cell slots. As bandwidth becomes lower, the delay difference is also lowered. This}
\text{is because the lower the bandwidth, the smaller the chances of having all of these VCs sending}
cells together, and hence, a lower maximum delay for these non-delay bounded VCs.

Figure 5.8 Maximum Delay vs Bandwidth – delay bounded VC and non-delay bounded VC

Figure 5.8 also shows two average delay curves for non-JEDD delay bounded VCs under
JEDD-SCFQ and pure SCFQ. The difference of average delay between the two algorithms is
smaller than that of maximum delay. This is because the penalty of having JEDD-delay bounded
VCs is distributed over all the non-JEDD delay bounded VCs. At the point when bandwidth is
greater than 30%, the average delay provided by JEDD-SCFQ is higher than the maximum delay provided by pure SCFQ. This happens because of that the JEDD-delay bounded VCs are punishing those non-JEDD-delay bounded VCs. The higher the bandwidth of those non-JEDD delay bounded VCs, the bigger the punishment, that is the higher the average delay.

Although, JEDD-SCFQ algorithm provides guaranteed jitter-delay to some connections. We should note that this algorithm is efficient for only a small proportion of VCs. For any number of JEDD-delay bounded VC, $N_{DB}$, it adds $N_{DB} \times MBS$ cell slots to non-JEDD delay bounded VCs. In this simulation, there are 4 out of 8 VCs are JEDD-delay bounded. This adds $(N_{DB} \times MBS = 4 \times 10) 40$ cell slots to maximum delay for those non-JEDD delay bounded VCs.
Chapter 6 Conclusion and Future Work

6.1 Conclusion

The performance and implementability of ATM scheduling algorithms were evaluated, specifically a rate-based algorithm, Self-Clocked-Fair-Queuing algorithm was investigated. This thesis has concentrated on a deterministic analysis of queuing delay in ATM scheduler. The issues and solutions discussed provided two basic services that an integrated network needs, that is, a method for providing simple best-effort traffic service by using SCFQ and a method for providing real-time traffic service with reasonable delay bounds and jitter-delay bounds by using DEDD-SCFQ and JEDD-SCFQ.

The use of deterministic delay bounds to guarantee performance is a crucial first step in designing a scheduler. The second step is to identify whether an algorithm is implementable. As presented in chapter 3, there is a wide variety of scheduling algorithms, ranging from first-come-first-serve to different priorities, from rate-based to schedule-based, and from work-conserving to non-work-conserving. The performance and implementability of some algorithms, such as GPS, PGPS/WFQ, SCFQ have been studied closely. In addition, the implementation issues were discussed in chapter 4. Although the implementation presented in this thesis is much simpler than a real commercial traffic management device, it laid out a foundation for developing a scheduler with efficient delay management.

Our simulation results, presented in chapter 5, show that when SCFQ is used for evenly distributed bandwidth allocation, it provides a fairly decent delay bound. However, when the
bandwidth allocation is uneven, the delay and delay-jitter incurred for those low bandwidth connections are extremely high. This coupling effect does not fair well with real-time traffic, in which the delay-jitter is more important than the delay itself. Two varied SCFQ algorithms, DEDD-SCFQ and JEDD-SCFQ, are developed to solve this bandwidth-delay coupling problem. These two algorithms provide real-time traffic connections a much tighter delay-jitter bound. They do so by penalizing best-of-effort connections. As shown in chapter 5, the maximum delay of those non-jitter-delay bounded connections is much higher with JEDD-SCFQ scheduler than that with pure SCFQ scheduler. Since this best-of-effort traffic, such as documents or emails, does not have tight delay requirements, the higher delay does not affect it critically.

This thesis has made extensive use of deterministic bounds to provide QoS. Statistical bounds exist for schedulers but they are heavily dependent on factors such as traffic load, traffic models and network topology. The use of statistical bounds may be possible in future. If statistical bounds for delay can be found with a very high probability, then a notion of statistical guarantees can be formulated [47]. However, such methods require a very robust statistical model, which is hard to model.

6.2 Future Work

In our research, we also looked at few other scheduling algorithms, such as start-time fair queuing and start-potential fair queuing. These algorithms differ from SCFQ in that they use different ways to calculate the finish number. However, they still suffer from the bandwidth and
delay coupling problem that we discussed before. It would be ideal to come up with an algorithm similar to JEDD-SCFQ, but have a more general and deterministic bounds to solve this problem.

Currently, the research is mature enough that work should focus on implementing the theory in this field. Although these scheduling algorithms have been developed for a while, commercial switches with non-trivial scheduling ability have just came out of market. This may due to the past experiences of viewing the network and technological issues of VLSI designs. In the past, no researchers could have envisioned the popularity of the Internet. In particular, researchers have to identify the direction of the future integrated network, whether is should be IP over ATM, Packet over SONET, or others. With the pace of VLSI technology is improving, these theoretical algorithms can now actually be implemented.

Finally, one interesting issue is about access schemes for ATM over satellite network. It has the advantage over terrestrial ATM in that it can cover a large area and is immune to network breakdown. It also presents a relatively low cost solution since terminals can access the network without infrastructure costs needed for ground networks using fiber optics. One practical question that needs to be answered is how efficient these scheduling algorithms can be if some parts of the network are connected by satellite links that have delays of more than 250ms [48].
BIBLIOGRAPHY


