FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
SESSION 2000/2001

DEVELOPMENT OF OBJECT-ORIENTED COMPONENTS FOR ATM NETWORK SIMULATION WITH EMPHASIS ON SCHEDULE POLICIES IN TRAFFIC SHAPING

2
1

BY

WONG CHEE SUM

UNDER THE SUPERVISION OF

MR. LING TECK CHAW

DISSENTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR DEGREE OF BACHELOR OF COMPUTER SCIENCE UNIVERSITY OF MALAYA

ABSTRACT

Asynchronous Transfer Mode (ATM) is considered to be the ground on which B-ISDN is to be built. The traffic control of ATM network is important to provide Quality of Service (QoS) for network applications. Therefore, different traffic management procedures have been introduced to achieve network performance objectives. Traffic shaping is one of the traffic management mechanisms that able to protect the network and the end-system from congestion problem.

This project focuses on the development for simulation of different scheduling policies in traffic shaping with object-oriented approach. The simulator is designed and implemented to ensure the correctness of scheduling within the network nodes, fairness of scheduling and to guarantee quality of service for ATM application. Simulation modeling is an ideal domain for the application of object-oriented methods, both in the modeling of problem domain entities and in the implementation of simulators.

A development of an object-oriented, platform independent, web-enabled, and multithread discrete-event simulator will build using Java programming language.

ACKNOWLEDGEMENT

Sincere appreciation is dedicated specially to my supervisor, Mr. Ling Teck Chaw for his generous guidance and supervision for the project. His precious advises and sound understanding throughout the project is deeply appreciated

I also wish to express my greatest gratitude to my moderator Mr. Ibrahim Bin Abu Bakar for his assistance and tolerance.

Also special thanks to the other members of the project team Lim Shiau Hong, Chin Kim Choo, Sin Wai Kit, Jimmy Tan, Yu Soon Lye, Phung Jacen, Teh Boon Pin and Wong Wing Hong for their cooperation and assistance in putting the project together.

Last but not least, I will like to thanks all my coursemates for theirs generous sharing of knowledge and support. These direct and indirectly contribute to the completion of the project.

TABLE OF CONTENTS

ABSTRACT i

ACKNOWLEDGEMENT ii

TABLE OF CONTENTS iii

LIST OF ILLUSTRATIONS vii

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION TO ASYNCHRONOUS TRANSFER MODE (ATM) 1

1.2 INTRODUCTION TO TRAFFIC SHAPING 2

1.2.1 Definition of Traffic Shaping 2

1.2.2 Advantages of Traffic Shaping 3

1.3 INTRODUCTION TO SCHEDULING POLICIES 4

1.3.1 The Needs of Scheduling Policies 4

1.3.2 Priority Scheduling 4

1.4 INTRODUCTION TO COMPUTER SIMULATION 5

1.4.1 Advantages and Disadvantages of Simulation 5

1.4.2 Types of Simulation 6

1.5 PROJECT OBJECTIVES 6

1.6 PROJECT SCHEDULE 8

1.7 REPORT ORGANIZATION 9

CHAPTER 2: A STUDY ON ATM NETWORK SIMULATORS

2.1 STUDY OF EXIST SIMULATORS 10

2.1.1 NIST ATM/HFC Network Simulator 10

2.1.2 REAL Network Simulator 11

2.2 STUDIES ON SEVERAL NETWORK SIMULATION MODELS 13

2.2.1 Modeling Traffic Management In ATM Networks With OPNET2 13

2.2.2 ATM Switch Simulation Tool based on C++ Object-oriented Programming Language 14

2.3 TRAFFIC MANAGEMENT IN ATM NETWORK 15

2.3.1 Traffic Management and Congestion Control 15

2.3.2 Service Categories 16

2.3.2.1 Definition For Service Categories 16

2.3.3.2 ATM Service Categories Parameters and Attributes 17

2.3.3 Quality of Service Parameters 18

2.3.3.1 Peak-to-peak Cell Delay Variation (peak-to-peak CDV) 19
2.3.3.2 Maximum Cell Transfer Delay (maxCTD) 19
2.3.3.3 Cell Loss Ratio (CLR) 20
2.3.3.4 Cell Error Ratio (CER) 20
2.3.3.5 Severely Errored Cell Block Ratio (SECBR) 20
2.3.3.6 Cell Misinsertion Rate (CMR) 21
2.3.4 Function and Procedures for Traffic Management *21*

2.4 Examples of Traffic Shaper **23**
2.4.1 CMOS 0.35 ATM Traffic Shaper *23*
2.4.2 Cell-Space Shaper *24*
2.5 Types of Scheduling Policies **24**
2.5.1 Dynamic Scheduling Algorithms *24*
2.5.1.1 Earliest Deadline First (EDF) 25
2.5.1.2 Virtual Clock 25
2.5.1.3 Dynamically Weighted 25
2.5.1.4 Minimum Laxity Threshold 26
2.5.1.5 Queue Length Threshold 26
2.5.2 Static Scheduling Algorithms *26*
2.5.2.1 First In First Out (FIFO) or First Come First Serve (FCFS) 26
2.5.2.2 Static Priority Scheduling (SPS) 27
2.5.2.3 Round Robin (RR) 27
2.5.2.4 Packet-by-packet Generalized Processor Sharing (PGPS) 28
2.6 Examples of Scheduling **28**
2.6.1 Algorithms on Dynamic Priority Scheduling for Heterogeneous Traffic in ATM *28*
2.6.1.1 Double Minimum Laxity Threshold 28
2.6.1.2 Double Queue Length Threshold 29
2.6.1.3 Hysteresis effect Queue Length Threshold 29
2.6.2 Time Dependent Priority Scheduling *30*
2.6.3 Traffic-Controlled Rate-Monotonic Priority Scheduling of ATM Cells *30*

2.7 Programming Language Approach **31**
2.7.1 Procedural Programming *31*
2.7.2 Object-oriented Programming *33*
2.7.2.1 Java Programming 34
2.7.2.2 C++ Programming 36
2.7.2.3 Java Programming Tools 38
2.8 Chapter Summary **38**

CHAPTER 3: JAVASIM AND PRIORITY SCHEDULING ALGORITHMS OVERVIEW

3.1 Javasim **39**
3.1.1 Simulation Engine *41*
3.1.2 Simulation Component *42*

3.2 Priority Scheduling Algorithms 43
3.2.1 Hysteresis effect Queue Length Threshold (HQLT) 43
3.2.2 Round Robin (RR) 44
3.2.3 Select Largest Queue (SLQ) 44

3.3 Chapter Summary 45

CHAPTER 4: SYSTEM ANALYSIS

4.1 Programming Language Choice 46
4.1.1 Object-oriented Programming Techniques 46
4.1.2 The Java Programming Language 46

4.2 Development Tools 47
4.2.1 Window 98 47
4.2.2 Borland Jbuilder 47

4.3 System Requirements 48
4.3.1 Functional Requirements 48
4.3.1.1 Topology Design System 48
4.3.1.2 User Input System 48
4.3.1.3 Generate Output 49
4.3.2 Non-Functional Requirements 49
4.3.2.1 Flexibility 49
4.3.2.2 Usability 49
4.3.2.3 Correctness 49
4.3.2.4 Scalability 49
4.3.2.5 Reusability 50
4.3.2.6 Portability 50
4.3.2.7 Maintainability 50
4.3.2.8 Manageability 50

4.4 Chapter Summary 50

CHAPTER 5: DESIGN OF ATM NETWORK SIMULATION

5.1 The Simulation Model 51
5.1.1 Overview 51
5.1.2 Output Ports of Switch 51
5.1.3 Multithreading 51

5.2 Queue Architecture Design 52
5.2.1 Output Port Queuing Model 52
5.2.2 Priority Scheduling At Output Port 54
5.2.2.1 Demultiplexing 54
5.2.2.2 Schedule Output 55

5.3 Modules Design 55

5.4 Component Class Design 57

5.5 CHAPTER SUMMARY 59

CHAPTER 6: IMPLEMENTATION

6.1 SYSTEM IMPLEMENTATION 60

6.2 CLASS IMPLEMENTATION 62

6.2.1 SPATMSwitch.java 62

6.2.2 SimParamPScheduler.java 64

6.2.3 CBRApp.java 65

6.2.4 VBRApp.java 65

6.3 PRIORITY SCHEDULING ALGORITHMS IMPLEMENTATION 67

6.3.1 FCFS and Fixed Priority (Javasim) 67

6.3.2 Hysteresis effect Queue Length Threshold 68

6.3.3 Round Robin 68

6.3.4 Select Largest Queue 69

6.4 CHAPTER SUMMARY 70

CHAPTER 7: TESTING

7.1 CLASS AND UNIT TESTING 71

7.1.1 SimParamPScheduler Class 71

7.1.2 Constant (CBR or ABR) and Poisson (VBR or ABR) Application Class ... 73

7.1.3 HQLT With Different Values of Upper Limit And Lower Limit 76

7.1.4 Performance of Round Robin (RR) Scheduling Algorithm 77

7.1.5 Performance of Select Largest Queue Scheduling (SLQ) Algorithm 78

7.2 SYSTEM TESTING 79

7.2.1 Javasim(FCFS & Fixed Priority) vs Hysteresis effect Queue Length Threshold (HQLT) 80

7.2.2 Round Robin (RR) vs Select Largest Queue (SLQ) For Same Priority Class 81

7.3 CHAPTER SUMMARY 83

CHAPTER 8: CONCLUSIONS 84

REFERENCES 85

LIST OF ILLUSTRATIONS

TABLES

Table 1.1 Project Schedule 8

Table 2.1 ATM Service Category Attributes 18

Table 5.1 Modules Function 56

Table 7.1 Simulation Results for Testing of CBR and VBR Applications 75

Table 7.2 Values of Output Parameters Generated During Simulation 77

Table 7.3 Data Generated for System Testing of Javasim vs HQLT 81

Table 7.4 Data Generated for System Testing of RR vs SLQ 83

FIGURES

Figure 1.1 Traffic Shaping 2

Figure 1.2 Three Sub-Fields of Computer Simulation 5

Figure 2.1 The OPNET ATM Model 14

Figure 2.2 Continuous Times Shaper Model 24

Figure 2.3 Minimum Laxity Threshold 29

Figure 2.4 A Typical C++ Environment 37

Figure 3.1 Hierarchy of all the significant objects in the Java Network Simulator .. 40

Figure 5.1 Switch Architecture With Output Port Queuing Model 52

Figure5.2 Methods of Cell Demultiplexing At Output Port (a) Within Different Priority Classes and (b) Within Same Priority Class 54

Figure 5.3 Modules Design 55

Figure 5.4 Switch Parameters' List 56

Figure 5.5 GUI Design for Priority Scheduler's Input Window 56

Figure 6.1 Flow of Cells In Switch 61

Figure 7.1 Input Window for Priority Scheduler Algorithms' Parameters 72

Figure 7.2 Network Topology for Testing of Constant and Poisson Applications 74

Figure 7.3 Parameters Input Window for Constant and Poisson Applications 74

Figure 7.4 Sample Output Data Generate During Simulation 75

Figure 7.5 Network Topology for System Testing of HQLT 76

Figure 7.6 Network Topology for System Testing of Javasim vs HQLT 80

Figure 7.7 Network Topology for System Testing of RR vs SLQ 82

CHAPTER 1: INTRODUCTION

This chapter begins with an introduction on asynchronous transfer mode (ATM), traffic shaping, scheduling policies and network simulation. The next section included the descriptions of project's objectives, project schedule and report organization.

1.1 Introduction to Asynchronous Transfer Mode (ATM)

Asynchronous Transfer Mode (ATM) is a set of international interface & signaling standard define by the ITU-T. ATM is a high performance, cell-oriented switching and multiplexing technology that utilizes fixed-length packets to carry different types of traffic. Furthermore, ATM can handle any kind of information such as voice, data, image, text & video in an integrated manner.

ATM is a connection-oriented packet switching technique in which all packets are of fixed length. ATM is a connection-oriented technology in the sense that before two systems on the network can communicate, they should inform all intermediate switches about their service requirements and traffic parameters.

ATM is a transmission technology that uses fixed and relatively short packets called cells. A cell is a 53 byte packet with 5 bytes of header/descriptor and 48 bytes of payload, or user traffic (voice, data or video) to combine data, voice and video. The reason for choosing a fixed-size packet is to ensure that the switching and multiplexing function could be carried out quickly and easily. The cells are transmitted over a connection called Permanent or Switched Virtual Circuit Connection (PVC or SVC). Even ATM does not protect data from errors, but it able to works well on digital lines with low bit error rates.

ATM is able to reduce infrastructure costs through efficient bandwidth management, operational simplicity and the consolidation of overlay networks. Beside that, it also provides a good bandwitdh flexibility and can be used efficiently from desktop computers to local area & wide area networks. Therefore, ATM technology has been implemented in a very broad range of networking devices such as, PC, workstation, and server network interface cards, workgroup and campus ATM switch, ATM multiplexers and so on.[1]

1.2 Introduction to Traffic Shaping

1.2.1 Definition Of Traffic Shaping

Traffic shaping is a mechanism that alters the traffic characteristics of a stream of cells on a connection to achieve better network efficiency whilst meeting the QoS objectives, or to ensure conformance at a subsequent interface. Traffic shaping mechanism is important in ATM networks, especially those that are interconnected or provide service guarantees.

Traffic shaping attempt to modify the temporal distribution of traffic to ensure that it meets the applicable traffic constraints. Shaping can be used in a source terminal connected to the network, within the network to account for jitter introduced by network element, between interconnected networks, or in a destination terminal to reconstruct timing information for jitter-sensitive applications such as video. A constraint can be apply to a particular connection, source (e.g. for charging), or (intermediate) destination (e.g. for rate adaptation).[2]

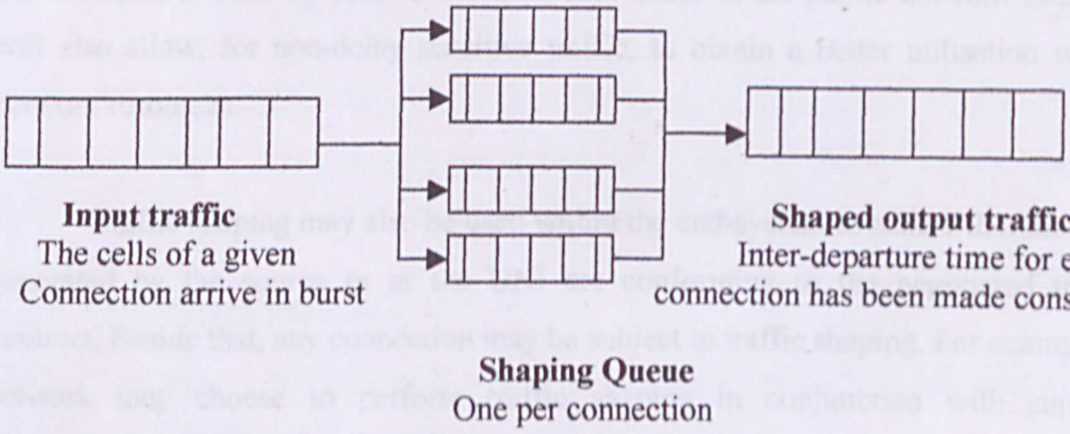

Input traffic
The cells of a given
Connection arrive in burst
Shaped output traffic
Inter-departure time for each
connection has been made constant
Shaping Queue
One per connection

Figure 1.1 Traffic Shaping [3]

Examples of traffic shaping are peak cell rate reduction, burst length limiting, reduction of CDV by suitably spacing cells in time, and cell scheduling policy.[2]

1.2.2 Advantages Of Traffic Shaping

Traffic shaping, provides an opportunity to boost statistical multiplexing gain by shaping traffic streams into much "smoother" form prior to merging. Also, a shaped traffic stream is predictable, allowing easy use of statistical models for accurate resource allocation. And traffic shaping allows interswitch trunks to operate at a higher level of utilization.

The traffic shaping function is important for the new broadband services that are being deployed. It is because traffic shaping is able to avoid information loss, provide the end user several traffic contract options in terms of bandwidth and insure an use of the communication channels.

In addition, traffic shaping able smoothes the cell stream of every connection to create a more predictable traffic profile. A more predictable traffic profile leads to better fairness, lower cell loss, and less stress on network resources.

With traffic shaping, each VC is prevented from bursting into the output link at an excessive speed. Traffic shaping at the egress of a network reduces CDV (Cell Delay Variation) across the User-to-Network Interface (UNI) to the ATM end system. Furthermore, it not only able to avoiding cell losses in the public network nodes, tt will also allow, for non-delay sensitive traffic, to obtain a better utilisation of the network resources.

Traffic shaping may also be used within the end-system to ensure that the cells generated by the source or at the UNI are conforming to the negotiated traffic contract. Beside that, any connection may be subject to traffic shaping. For example, a network may choose to perform traffic shaping in conjunction with suitable UPC/NPC functions and/or virtual source/destination. [4]

1.3 Introduction to Scheduling Policies

1.3.1 The Needs of Scheduling Policies

Scheduling policy is use to determines which queue is given the opportunity to transmit a cell that is stored in the buffer and to divide the available bandwitdh among various contending classes of service. Therefore, in an ATM network, the scheduling policy at each output link has a direct impact on message delays when the aggregate incoming traffic is higher that the output link capacity of an ATM traffic shaper. Thus admission criteria for real-time applications should take into account the specific scheduling policy used.[18]

Scheduling policies maybe implement in 2 way, that are:

- Coarse granularity (per class scheduling) that divide bandwidth among different service classes.
- Fine level granularity (per Virtual Circuit scheduling) that divide bandwidth between various connections in a service class.[5]

The scheduling policies strongly influence the cell delay which is most important for delay sensitive services. It is because the scheduling policies are able to maintain a queue containing all packets of cells which waiting for transmission.

There are different algorithms for scheduling policies such as first come first serve, round robin, select largest queue and so on. However, overall the scheduling policies can be divided into two main categories: static or dynamic.[6]

1.3.2 Priority Scheduling

Priority scheduling is used to determine which connections are served first. The main design objectives for this scheduling policy are:

- Fairness: The scheduling policy must serve egress network connections of the same priority class with equal probability.
- Little delay: The scheduling policy should minimize the cell delay for all service categories [7]

By priority scheduling, one may serve first cells that can tolerate less delay. Besides, priority scheduling coupled with a certain amount of buffering is able to improve the resource utilization efficiency and met the service quality requirements for all services.

1.4 Introduction to Computer Simulation

Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analyzing the execution output. Simulation embodies the principle of "learning by doing" - to learn about the system a model must be built and then operate the model. To understand reality and all of its complexity, artificial objects must be built and dynamically act out roles with them. Within the overall task of simulation, there are three primary sub-fields: model design, model execution and model analysis [8].

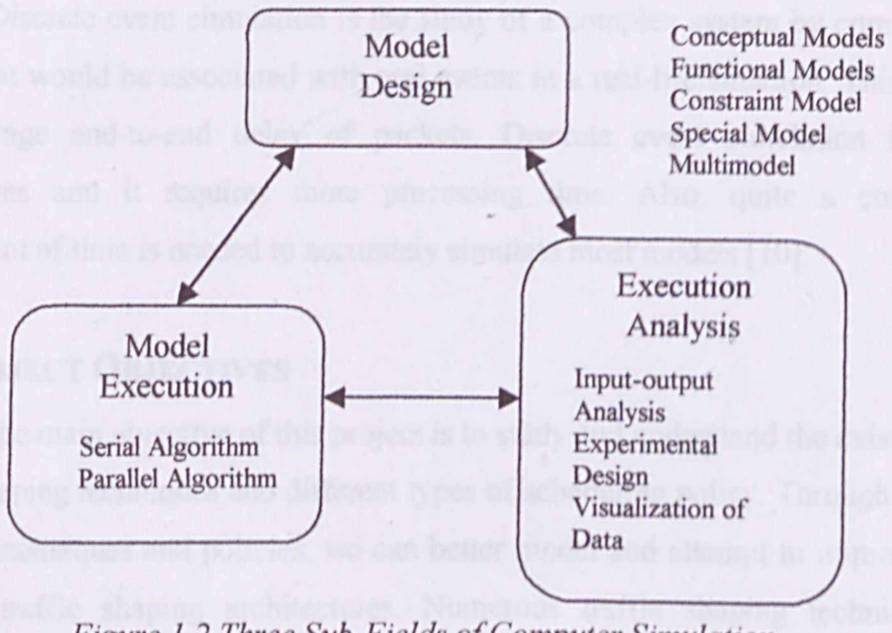


Figure 1.2 Three Sub-Fields of Computer Simulation

1.4.1 Advantages and Disadvantages of Simulation

The advantages of simulation are described as below:

- Economical and quick to assemble.
- Given sufficient computing resources, can do large-scale tests.
- Tests are controlled, reproducible.

However, certain problems exist with simulation too:

- Need to redo code for simulation environment.
- Simulation implementation may differ considerably from real one.
- Synthetic environment may also poorly represent real one [9].

1.4.2 Types of Simulation

Essentially there are two forms of network simulation: Analytical modeling and Discrete event simulation.

Analytical modeling is a mathematical technique that characterizes a network as a set of equations. The main disadvantage associated with analytical models is that it is over simplistic view of the network and their inability to simulate the dynamic nature of a computer network.

Discrete event simulation is the study of a complex system by computing the times that would be associated with real events in a real-life situation. This could be the average end-to-end delay of packets. Discrete event simulation has many advantages and it requires more processing time. Also, quite a considerable investment of time is needed to accurately simulate most models [10].

1.5 Project Objectives

The main objective of this project is to study and understand the existing ATM traffic shaping techniques and different types of scheduling policy. Through the study of these techniques and policies, we can better model and attempt to improve on the existing traffic shaping architectures. Numerous traffic shaping techniques and scheduling policies have been proposed since the implementation of the ATM network. The thesis begins with a study on the current traffic shaping techniques and scheduling policies.

The second objective is to develop an ATM traffic shaper simulator which is uses to provide different scheduling policies. Object-oriented approach is uses to develop the simulator in order to take advantage of the features such as modularity, extensibility, reusability and others. Moreover, applying multithreading into a simulator design can be used to closely model the real ATM traffic shaper.

The final objective is the creation of a portable, cross-platform and web-enabled simulator.

1.6 Project Schedule

Project Schedule										
Task	**Start**	**Finish**	**June**	**July**	**Aug**	**Sept**	**Oct**	**Nov**	**Dec**	**Jan**
1. Research and Review	7/6/2000	18/8/2000	▨	▨	▨					
2. Requirement & Analysis	1/7/2000	7/9/2000		▨	▨	▨				
3. System Design	1/9/2000	31/10/2000				▨	▨			
4. Modules Development *	1/10/2000	15/12/2000					▨	▨	▨	
5. System Integration & testing	1/12/1999	15/1/2001							▨	▨
6. Documentation	1/9/1999	22/1/2001				▨	▨	▨	▨	▨

* Modules Development includes modules design and coding.

Table 1.1 Project Schedule

1.7 Report Organization

Chapter 1 gives an overview of ATM and the definition of traffic shaping and scheduling policies in ATM networks. Brief description on computer simulation is also available. Chapter 2 introduces several types on simulator available currently and studies are put on to their simulators. Also, different types and examples of traffic shaping and scheduling policies are discussed in this chapter.

Chapter 3 explains the programming language and development tools that will be used in assisting the development of the system. Further details of the functional and non-functional requirement for the system development also explained in this chapter. Beside that, an overall of system design will be shown in this chapter, followed by module design, objects design, user interface design and others.

First section of Chapter 4 discusses the two important part of Javasim ATM Simulator: *Simulator Engine and Simulator Components.* The second section discusses proposed priority scheduling algorithms to schedule the outgoing cells at the output port of a switch for different class of service and same class of service.

Chapter 5 discusses the programming language and development tools chosen to create the network simulator components. Beside that, brief description about functionality requirements, non-functionality requirements and development tools for simulation is also available.

Chapter 6 discusses implementation of priority scheduling algorithms at output ports of the switch for the simulator. Classes with important data items and its associate data are discussed further in the implementation section. Further more, the details implementation for the proposed algorithms also included.

Description of class, unit and system testing that had been done for the simulator is included in Chapter 7. Beside that, different priority scheduling algorithms also can be compare base on the simulation results of the system testing. Finally, an overview conclusion of this project is included at Chapter 8 of this report.

CHAPTER 2: LITERATURE REVIEWS

2.1 Study of Exist ATM Simulators

As the emerging of speeds and dynamic of the computer networks today, management of the network traffic is highly important.

Therefore, there are a few simulation packages to describe a number of simulation experiments performed on the number of different configurations without the expanse of building a real network. The simulator actually offers a practical means of obtaining accurate information on which to plan and design a new system. Simulation is a useful technique for computer systems to perform analysis for high-speed network, especially for ATM.

2.1.1 NIST ATM/HFC Network Simulator

The ATM/HFC network simulator is a tool to analyze the behavior of ATM and HFC networks without the expense of building a real network. This simulator was developed at the National Institute of Standards and Technology (NIST). This simulator is written in C Language whereby it is written in structural programming approach. Typically, the simulator program includes a graphical interface which provides the user with a means to display the topology of the network, define the parameters and connectivity of the network, log data, and to save and load the network configuration. In addition to the user interface, the simulator has an event manager, I/O routines, and various tools that can be used to build components [11].

Advantages

The user can create different network topologies, adjust the parameters of each component's operation, measure network activity, save/load different simulation configuration and log data during simulation execution. The simulator is equipped with graphical user interface.

Limitations

Users of the simulator might face problems setting up the network topology because of the requirement to consider a large number of parameters.

User or programmers needs to have strong foundation in C programming language to customize the simulator's components. Besides, it is using procedural approach whereby the components have overlapped functions between the components. This is not supposed to happen in object-oriented programming approach.

The simulator only can run is UNIX or LINUX platform. This makes the simulator can only run in limited platforms and it is not widely used.

2.1.2 REAL Network Simulator

The REAL network simulator is a network simulator designed for testing on congestion and flow control mechanisms. The simulator takes as input a scenario, which is description of network topology, protocols, workload and control parameters. It produce as output statistics such as the number of packets sent by each source of data, the queuing point, the number of dropped and retransmitted packets and other similar information. This simulator has many different versions.

For REAL version 5.0, it provides users with a way of specifying such networks and to simulate their behavior. It provides around 30 modules (written in C) that exactly emulate the action of several well-known flow control protocols (such as TCP), and 5 research scheduling discipline (such as Fair Queuing and Hierarchical Round Robin). Besides, it includes a graphical user interface (GUI) written in Java. This allows users to quickly build simulation scenarios with a point-and-click interface [12].

This REAL 5.0 simulator runs on Sun3s, Sparcs, MIPS boxes, Vaxen and 3B2, under 4.3BSD-like operating systems: SunOS, IRIX, UMIPS, Ultrix etc. Besides, the REAL version 4.0 has been successfully ported to i386/FreeBSD 2.0.5 platform and the Linux (Red Hat Release) platform [9].

The files in REAL 5.0 can be divided into the following logical classes:

- *Node functions*: These are the functions that execute protocols in nodes.
- *Queue management and routing:* These manage buffers in nodes and gateways, and perform packet switching.

Node functions implement computation at each of the nodes in the network. There are three types of node function: source, router and sink.

The queue management functions are written in an object oriented and layered style. The queue objects are manipulated by a small set of functions. Each layer provides services that the layer above uses to provide its own services. Packets are buffered in a per-conversation linked list and are accessed by two pointers: one points to the packet at the head of the queue, and another points to the tail. Each packet has a field, which points to the next packet in the queue.

Comparing REAL 5.0 to REAL 4.0 in October 1993, and in the last four years, many changes have been made. These include

- many new simulation modules
- a Java-based GUI.
- faster, smaller, cleaner simulation engine
- ports to FreeBSD, Solaris, and Digital Unix (OSF/3)
- simulation exercises based on my book
- minor bug fixes

Advantages

This simulator provides a flexible testbed for studying the dynamic behaviour of flow and congestion control schemes in packet switch data networks. Besides, the user can modify the simulator software to accommodate network components.

Limitations

Some of the REAL version's network simulator does not give user an interactive modelling environment with a graphical user interface (GUI) representation capabilities but it is available in REAL version 5.0. This version includes a graphical user interface (GUI) written in Java and it allows users to quickly build simulation scenarios with a point-and-click interface. Besides, knowledge of C programming language and different platforms is a must for programmers to make changes from the source code provided because this simulator only will run in several platforms.

2.2 Study on Several Network Simulation Models

A survey is made on several network simulation and management models done by several researchers, master and phD students of universities, which are available on many journals and papers. In this survey, there are a few simulator models, which will be discussed: An object-oriented network model for development of modeling traffic management In ATM networks with OPNET and an ATM switch simulation tool based on the C++ object-oriented programming language. Basically, the models are mainly based on object-oriented approaches.

2.2.1 Modeling Traffic Management In ATM Networks With OPNET

This paper [13], describe the OPNET models that have been developed for ATM and ABR design and analysis.

The example network topology used for the design and development of traffic management functions within AMS represents an N-source configuration shown in *Figure 2.1.* Source and destination end-systems are connected to a pair of ATM switches that communicate via a bottleneck link.

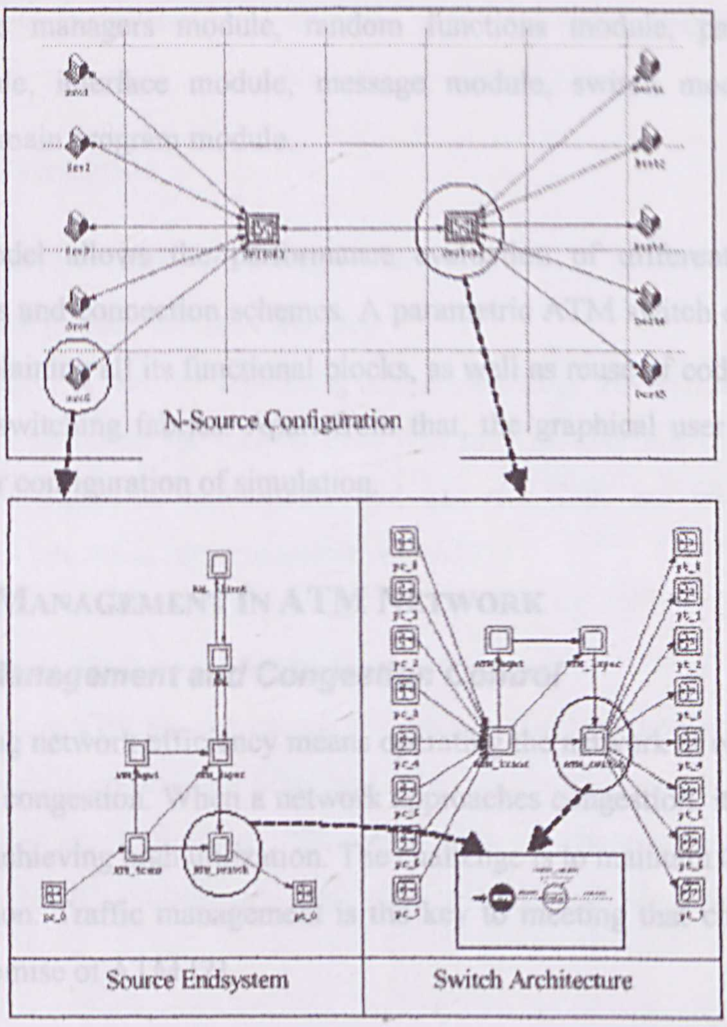


Figure 2.1 The OPNET ATM Model

The OPNET process modeling methodology was used in the development of the switch process model that delivered basic capabilities of the core ATM switching fabric, ABR feedback control, buffer management and scheduling. The key steps of this modeling methodology include: definition of the system context, identifying interdependent modules, enumeration of events, selection of states of a process, construction of an event response table and construction of the finite state machine.

2.2.2 ATM Switch Simulation Tool based on C++ Object-oriented Programming Language

In [14], a software package to simulate ATM switches written in C++ programming language is presented and its main modules and objects are described. The package consists of several modules: General purposed module, input/output module, storage managers module, random functions module, pattern module, buffering module, interface module, message module, switch module, statistics module and the main program module.

This model allows the performance evaluation of different input traffic characterizations and connection schemes. A parametric ATM switch element model is described explaining all its functional blocks, as well as reuse of codes to construct more complex switching fabrics. Apart from that, the graphical user interface also implemented for configuration of simulation.

2.3 Traffic Management In ATM Network

2.3.1 Traffic Management and Congestion Control

Achieving network efficiency means operating the network at or near capacity — approaching congestion. When a network approaches congestion, it is functioning efficiently and achieving high utilization. The challenge is to maintain QoS in the face of high utilization. Traffic management is the key to meeting that challenge and to realizing the promise of ATM.[2]

Traffic management is concerned with ensuring that users get their desired quality of service. The problem is especially difficult during periods of heavy load particularly if the traffic demands cannot be predicted in advance. This is why congestion control, although only a part of the traffic management issues, is the most essential aspect of traffic management.

When two bursts arrive simultaneously at a node, the queue lengths may become large very fast resulting in buffer overflow. Also, the range of link speeds is growing fast as higher speed links are being introduced in slower networks of the past. At the points, where the total input rate is larger than the output link capacity, congestion becomes a problem.

Proper traffic management and congestion control helps ensure efficient and fair operation of networks in spite of constantly varying demand. This is particularly important for the data traffic which has very little predictability and, therefore, cannot reserve resources in advance as in the case of voice telecommunications networks.[15]

2.3.2 Service Categories

2.3.2.1 Definition For Service Categories

Constant Bit Rate (CBR) Service Category

This category is used for emulating circuit switching. The cell rate is constant. Cell loss ratio is specified for CLP=0 cells and may or may not be specified for CLP=1 cells. Examples of applications that can use CBR are telephone, video conferencing, and television (entertainment video).[15]

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

The real-time VBR service category is intended for real-time applications, i.e., those requiring tightly constrained delay and delay variation, as would be appropriate for voice and video applications. rt-VBR connections are characterized in terms of a Peak Cell Rate (PCR), Sustainable Cell Rate (SCR), and Maximum Burst Size (MBS).[15]

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

The non-real-time VBR service category is intended for non-real-time applications which have bursty traffic characteristics and which are characterized in terms of a PCR, SCR, and MBS. For those cells which are transferred within the traffic contract, the application expects a low cell loss ratio. Non-real-time VBR service may support statistical multiplexing of connections. No delay bounds are associated with this service category.[15]

Unspecified Bit Rate (UBR) Service Category

This category is designed for those data applications that want to use any left-over capacity and are not sensitive to cell loss or delay. Such connections are not rejected on the basis of bandwidth shortage (no connection admission control) and not policed for their usage behavior. During congestion, the cells are lost but the sources are not expected to reduce their cell rate. In stead, these applications may have their own higher-level cell loss recovery and retransmission mechanisms. Examples of applications that can use this service are email, file transfer, news feed, etc. Of course, these same applications can use the ABR service, if desired.[15]

Available Bit Rate (ABR) Service Category

This category is designed for normal data traffic such as file transfer and email. Although, the standard does not require the cell transfer delay and cell loss ratio to be guaranteed or minimized, it is desirable for switches to minimize the delay and loss as much as possible. Depending upon the congestion state of the network, the source is required to control its rate. The users are allowed to declare a minimum cell rate, which is guaranteed to the VC by the network. Most VCs will ask for an MCR of zero. Those with higher MCR may be denied connection if sufficient bandwidth is not available.[15]

Guaranteed Frame Rate (GFR) Service Category Definition

The GFR service category is intended to support non-real-time applications. It is designed for applications that may require a minimum rate guarantee and can benefit from accessing additional bandwidth dynamically available in the network. It does not require adherence to a flow control protocol. The service guarantee is based on AAL-5 PDUs (frames) and, under congestion conditions, the network attempts to discard complete PDUs instead of discarding cells without reference to frame boundaries.[2]

2.3.2.2 ATM Service Categories Parameters and Attributes

Table 2-1 provides a list of ATM attributes (traffic parameters, QoS parameters, and feedback characteristics) and identifies whether and how these are supported for each service category.

	ATM Layer Service Category					
Attribute	CBR	rt-VBR	nrt-VBR	UBR	ABR	GFR
Traffic Parameters$_4$:						
PCR and CDVT$_5$	Specified			Specified	Specified$_3$	Specified
SCR, MBS, CDVT$_{5,6}$	n/a	Specified		n/a		
MCR	n/a				Specified	n/a
MCR, MBS, MFS, CDVT$_5$	n/a					Specified
QoS Parameters$_2$:						
Peak-to-peak CDV	Specified		Unspecified			
MaxCTD	Specified		Unspecified			
CLR	Specified			Unspecified	See Note 1	See Note 7
Other Attributes:						
Feedback	Unspecified				Specified$_6$	Unspecified

Table 2.1 ATM Service Category Attributes

Notes:

1. CLR is low for sources that adjust cell flow in response to control information. Whether a quantitative value for CLR is specified is network specific.
2. Might not be subject to CAC and UPC procedures.
3. Represents the maximum rate at which the ABR source may ever send. The actual rate is subject to the control information.
4. These parameters are either explicitly or implicitly specified for PVCs or SVCs.
5. CDVT refers to the Cell Delay Variation Tolerance (see Section 4.4.1). CDVT is not signaled. In general, CDVT need not have a unique value for a connection. Different values may apply at each interface along the path of a connection.
6. See Section 2.4.
7. CLR is low for frames that are eligible for the service guarantee. Whether a quantitative value for CLR is specified is network specific. [2]

2.3.3 Quality of Service Parameter

Six QoS parameters that correspond to network performance objectives are identified in this specification. Three of these may be negotiated between the end-systems and the networks. One or more values of the QoS parameters may be offered on a per connection basis, corresponding to the number of related performance objectives supported by the network. Support of different performance objectives can be done by routing the connection to meet different objectives, or by implementation-specific mechanisms within individual network elements.

The following QoS parameters are negotiated:

- Peak-to-peak Cell Delay Variation (peak-to-peak CDV)
- Maximum Cell Transfer Delay (maxCTD)
- Cell Loss Ratio (CLR)

The following QoS parameters are not negotiated:

- Cell Error Ratio (CER)
- Severely Errored Cell Block Ratio (SECBR)
- Cell Misinsertion Rate (CMR) [2]

2.3.3.1 Peak-to-peak Cell Delay Variation (peak-to-peak CDV)

The peak-to-peak CDV is the difference between the (1 - α) quantile of the CTD and the fixed CTD that could be experienced by any delivered cell on a connection during the entire connection holding time. The term "peak-to-peak" refers to the difference between the best and worst case of CTD, where the best case is equal to the fixed delay, and the worst case is equal to a value likely to be exceeded with probability no greater than a. Assuming that the fixed delay is the reference delay for the two point CDV, the range of the distribution of the two-point CDV is the same as the peak-to-peak CDV. Refer to UNI Signaling 4.0 for the coding of peak-to-peak CDV.

Networks have a finite ability to control peak-to-peak CDV. Therefore, end-systems cannot expect to negotiate arbitrarily small values of peak-to-peak CDV as their sole means of meeting jitter and wander tolerances.[2]

2.3.3.2 Maximum Cell Transfer Delay (maxCTD)

The maximum Cell Transfer Delay (maxCTD) specified for a connection is the (1 - α) quantile of CTD. The CLR at connection request time is used to place an upper bound on a. When a switch accumulates maxCTD or CDV it may choose a smaller α which may have the effect of over-estimating the cumulative maxCTD or CDV. The assumed relationship between CLR and α is for further study.[2]

2.3.3.3 Cell Loss Ratio (CLR)

The Cell Loss Ratio is defined for a connection as:

$$\text{CLR} = \frac{\text{Lost Cells}}{\text{Total Transmitted Cells}}$$

Lost and transmitted cells counted in severely errored cell blocks should be excluded from the cell population in computing cell loss ratio. Each ATM cell has a "Cell Loss Priority (CLP)" bit in the header. During congestion, the network first drops cells that have CLP bit set. Since the loss of CLP=0 cell is more harmful to the operation of the application, CLR can be specified separately for cells with CLP=1 and for those with CLP=0.[2]

2.3.3.4 Cell Error Ratio (CER)

The Cell Error Ratio (CER) is defined as follows for a connection:

$$\text{CER} = \frac{\text{Errored Cells}}{\text{Successfully Transferred Cells} + \text{Errored Cells}}$$

Successfully transferred cells and errored cells contained in cell blocks counted as severely errored Cell Blocks should be excluded from the population used in calculating cell error ratio.[2]

2.3.3.5 Severely Errored Cell Block Ratio (SECBR)

The Severely Errored Cell Block Ratio (SECBR) is defined as follows for a connection:

$$\text{SECBR} = \frac{\text{Severely Errored Cell Blocks}}{\text{Total Transmitted Cell Blocks}}$$

A cell block is a sequence of N cells transmitted consecutively on a given connection. A severely errored cell block outcome occurs when more than M errored cells, lost cells, or misinserted cell outcomes are observed in a received cell block.

For practical measurement purposes, a cell block will normally correspond to the number of user information cells transmitted between successive OAM cells. Refer to ITU-T Recommendation I.610 for the size of cell blocks.[2]

2.3.3.6 Cell Misinsertion Rate (CMR)

The Cell Misinsertion Rate (CMR) is defined as follows for a connection:

$$\text{CMR} = \frac{\text{Misinserted Cells}}{\text{Time Interval}}$$

Severely Errored Cell Blocks should be excluded from the population when calculating the cell misinsertion rate. Cell misinsertion on a particular connection is most often caused by an undetected error in the header of a cell being transmitted on a different connection. This performance parameter is defined as a rate (rather than the ratio) since the occurrence of misinserted cells is independent of the number of transmitted cells received on the corresponding connection.[2]

2.3.4 Function and Procedures for Traffic Management

The following functions form a framework for managing and controlling traffic and congestion in ATM networks and may be used in appropriate combinations depending on the service category.[2]

- ***Connection Admission Control (CAC)*** is defined as the set of actions taken by the network during the call set-up phase in order to determine whether a connection request can be accepted or should be rejected (or whether a request for re-allocation can be accommodated).

- ***Feedback controls*** are defined as the set of actions taken by the network and by end-systems to regulate the traffic submitted on ATM connections according to the state of network elements. This specification defines one network feedback control mechanism: the ABR flow control. The ABR flow control may be used to adaptively share the available bandwidth among participating users.

- ***Usage Parameter Control (UPC)*** is defined as the set of actions taken by the network to monitor traffic and enforce the traffic contract at the User Network. Network Parameter Control (NPC) is a similarly defined set of actions at the Network Node Interface. The main purpose of UPC and NPC is to protect network resources from malicious as well as unintentional misbehavior, which can affect the QoS of other already established connections, by detecting violations of negotiated parameters and taking appropriate actions. Such actions may include cell discard and cell tagging.

- ***Cell Loss Priority control***: For some service categories the end system may generate traffic flows of cells with Cell Loss Priority (CLP) marking. The network may follow models which treat this marking as transparent or as significant. If treated as significant, the network may selectively discard cells marked with a low priority to protect, as far as possible, the QoS objectives of cells with high priority.

- ***Traffic Shaping***: Traffic shaping mechanisms may be used to achieve a desired modification to the traffic characteristics of a connection. The objectives of this function are to achieve a better network efficiency whilst meeting the QoS objectives and/or to ensure connection traffic conformance at a subsequent interface.

- ***Network Resource Management***: The service architecture allows logical separation of connections according to service characteristics. Although cell scheduling and resource provisioning are implementation and network specific, they can be utilized to provide appropriate isolation and access to resources. Virtual Paths are a useful tool for resource management.

- ***Frame Discard***: A congested network that needs to discard cells may discard at the frame level rather than at the cell level.

2.4 Examples of Traffic Shaper

2.4.1 CMOS 0.35 ATM Traffic Shaper

The design and implementation of CMOS 0.35 ATM Traffic Shaper was outlined in this paper [3]. The main function of this ATS is the collection of low bit rate traffics to fill a higher bit rate pipe in order to reduce the cost of ATM based services, nowadays mainly influenced by transmission cost. The chip was designed with a Top-Down methodology using as HDL, Verilog.

The ATS circuit is sort of queue administrator; it stores each cell in a different queue according to its connection identifier. The time interval between two consecutive cells from the same queue can be programmed independently for every active connection.

The chosen shaping mechanism is based on a memory-less algorithm: the outcoming bit rate for each connection (queue) is calculated in terms of the speed that can be handled by the link (network connection).

$$V_c = V_l \,.\, \frac{N}{2^k} \quad \text{with} \quad N \in [0, 2^k] \quad \text{and} \quad V \in [V_l, {}^{V_l}\!/_{2^k}]$$

When V_c in the connection speed, V_l the link speed, N is the parameter used to specify the connection speed.

2.4.2 Cell-Space Shaper

The proposed cell-space shaper [16] is modeled as a FIFO server. The service time depends on the interarrival times of the arrival process, and on whether the shaper is empty upon arrival. The service time is optimized, assuming the shaper is empty, so that the interdeparture time variance of cells leaving the shaper queue, heuristics are included in the optimization.

In order to shape sources where the cell arrival process is known and is analytically tractable, a continuous time shaper model is discussed. On the other hand, for cases when the cell arrival process is intractable, or only a cell interarrival histogram is available, a discrete time version of the shaper model is discussed too.

The shaper can represented as a queue, as shown below:

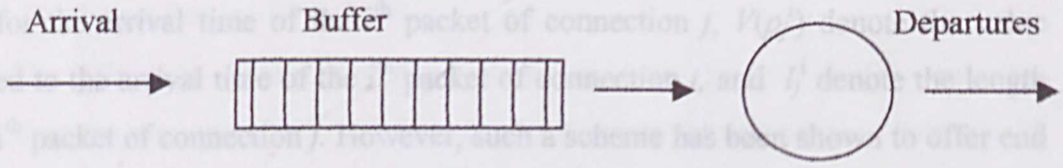


Figure 2.2 Continuous Times Shaper Model

This simple traffic shaping model found an optimal shaping parameter of source, based on its statistical characteristics. Using this parameter, cell arrivals were delayed, as needed, in order to realise a less bursty source. The reduction in burstiness was characterised by a reduction in the cell interarrival variance of the source.

2.5 Types of Scheduling Policies

2.5.1 Dynamic Scheduling Algorithms

Scheduling with real-time constraints requires dynamic algorithms to increase utilization and for advanced processing of multiple class traffics. Dynamic Scheduling Algorithms policy provides a very efficient scheme for real time traffic and non real time traffic.

2.5.1.1 **Earliest Deadline First (EDF)**

With EDF scheduling, each connection $i \in N$ is assigned a delay bound d_j, where the delay bound maybe different for each connection. A n EDF scheduler selects packets for transmission in increasing order of packet deadlines, where the deadlines are calculated as the sum of arrival time and delay bound of the packet. EDF is relatively difficult to implement and entails significant overhead.

EDF is optimal with respect top Lateness, but Locke's experiments have shown that the algorithm performs very poorly in overhead conditions. This is because it gives highest priority to packets that are close to missing their deadlines.[6]

2.5.1.2 **Virtual Clock**

The virtual clock scheduling algorithm assigns each packet a value upon its arrival and transmits packets. Let p_j^i stand for the ith packet of connection j, $A(p_j^i)$ stand for the arrival time of the ith packet of connection j, $V(p_j^i)$ denote the value assigned to the arrival time of the ith packet of connection j, and l_j^i denote the length of the ith packet of connection j. However, such a scheme has been shown to offer end to end delay deadline by effecting a trade-off between bandwidth and buffer space. The ratio of this trade-off is inflexible, and can not be altered due to the nature of the algorithm.[6]

2.5.1.3 **Dynamically Weighted**

The dynamically weighted priority scheduling algorithm provides a means to design a traffic shaper that is priority based but contains a delay sensitive portion to prevent starvation of data. The dynamically weighted priority scheduling algorithm provides a mechanism for simultaneously improving the balance of cell loss and delay.

We consider a time dependent "instantaneous priority index" *Pj(t)* for a jth class of service (a queue) at a given time *t* to be

$$P_j(t) = \frac{U_j}{[W_j(t)]}$$

Where *Uj* is the associated fixed priority number, a lower *Uj* means a higher priority, and *Wj(t)* is the amount of time the oldest cell in the j th class has waited in queue j. In our implementation of this algorithm, the priority index for each queue is recalculated for every output time slot (based on the speed of the outbound link). The queue with the lowest value of priority index is awarded the time slot and is permitted to transmit a cell during that time. The fixed priority values, *U*j, can be chosen arbitrarily (given that the higher priority queue gets the smaller number). For example, we can choose the values 1 and 2 for CBR and VBR classes, respectively. The value for *Wj* is specified in units of 1/100 th of a microsecond. Note that for ß=0 we have a fixed priority scheduling where *Pj(t)=U*j. For a very large ß, *Pj(t)* is heavily influenced by the wait time of the cell and the scheduling mechanism behaves as a *FIF*O. This algorithm can be considered a *dynamically weighted priority* scheme where the weights depend on the state of the queues at a given time. [5]

2.5.1.4 **Minimum Laxity Threshold**

The Minimum Laxity Threshold algorithm gives priority to the real time traffic when the minimum laxity of a real time cell is less than or equal to a threshold, LTH: otherwise priority is given to the non real-time traffic (NRT). The minimum laxity is defined as the amount of time until the first deadline of all queued real time cells expires.[17]

2.5.1.5 **Queue Length Threshold**

The Queue Length Threshold policy gives priority to the non real time traffic (NRT) when the number of queued non real-time cell is above a threshold, QTH; otherwise real time traffic (RTT) take priority. [17]

2.5.2 Static Scheduling Algorithms

Static Scheduling Algorithms is utilization within one priority class.

2.5.2.1 **First In First Out (FIFO) or First Come First Serve (FCFS)**

This mechanism represents a scheduling algorithm where cells are processed as they are received. To ensure that a fair comparison is made, this scheduling mechanism was implemented with the same rules for sharing buffer space as the other scheduling schemes analyzed later. However, the policy for determining which data was to be "de-queued" was simply the data that was "First In" (or oldest).

FIFO schedulers transmit all packets in order of their arrival. Since the maximum delay in a FIFO scheduler is the same for all streams $i \in N$, all steams must have identical delay bound. It is easy to implement with very low overhead and therefore attractive.[6]

2.5.2.2 Static Priority Scheduling (SPS)

Static priority scheduling, which gives the priority to the delay sensitive class, can be regarded as the simplest one. However, in SPS scheme, the traffic is processed without flexibility, so low priority class (loss sensitive traffic) will be sacrificed relatively. Owing to these defects, the SPS scheme hardly satisfy RTT and NRT spontaneously.

Static priority scheduling is popular for traffic scheduling in ATM switches because it is less costly than dynamic priority scheduling while being sensitive to the delay constraints of connections.

In a Static Priority scheduler, each stream $i \in N$ is assigned a priority p with $1 \leq p \leq P$, where a lower priority index indicates a higher priority. All connection with priority p have the same delay bound d_p, with $d_p < d_q$, if p < q. SP maintains one FIFO queue for each priority level, always selecting the first packet in the highest priority FIFO queue for transmission. This offers a choice of different deadlines and is implementable with very low overhead. However, lower priorities may experience strict in order to avoid such a scenario. This can result in low utilization of the network. [6]

2.5.2.3 Round Robin (RR)

The round robin scheduling algorithm treats each queue with equal priority. The "fairness" of the *round robin* scheme occurs when data is to be "de-queued" from the traffic shaper and sent on the outbound OC-3 link. If data exists in more than one queue, then each queue will be serviced in order with an equal share of the outgoing bandwidth. If data exists in only one queue, then that queue will be able to transmit data at the rate of the outbound link (minus any processing delay of the traffic shaper). [18]

2.5.2.4 **Packet-by-packet Generalized Processor Sharing (PGPS)**

Packet-by-packet Generalized Processor Sharing (PGPS) is an approximation of Generalized Processor Sharing (GPS), which work as follows. When choosing a packet to send, PGPS sends the packet that would be the next packet to finish its transmission if GPS were being used and no new packets arrive.

PGPS must timestamp every new message and maintain a sorted queue of the message at the output link. Thus, the run time overhead of PGPS is significantly higher than that for FCFS or RR. In the study [18], an idealized implementation of PGPS with no run time overhead is considered. Thus, the performance of the PGPS scheduler acts as a benchmark for comparing the performance of FCFS and RR.[18]

2.6 Examples of Scheduling Policies

2.6.1 Algorithms on Dynamic Priority Scheduling for Heterogeneous Traffic in ATM

In this paper [17], Double Minimum Laxity Threshold, Double Queue Length Threshold and Hysteresis Queue Length Threshold algorithms are proposed as Dynamic Priority Scheduling techniques for advanced processing of multiple class traffics. According to the simulation results, it can be shown that the proposed algorithms enhance the processing performance versus conventional algorithm for 2 or more classes delay sensitive traffics and for 2 or more classes of non real time traffics.

2.6.1.1 **Double Minimum Laxity Threshold**

The Double Laxity Threshold algorithm proposed in this paper, is the mechanism which divide the RTT into two classes; real time traffic 1(RTT1) and real time traffic 2 (RTT2) according to the delay sensitivity of traffics. The priority is assigned to the traffic according to the selective judgement from two comparison results between the laxity of RTT1 and the laxity threshold 1 (LTH1), the laxity of RTT52 and the laxity threshold 2 (LTH2), respectively. Here, the reason of setting the two laxity threshold is that there are two kinds of RTT in ATM traffic class.

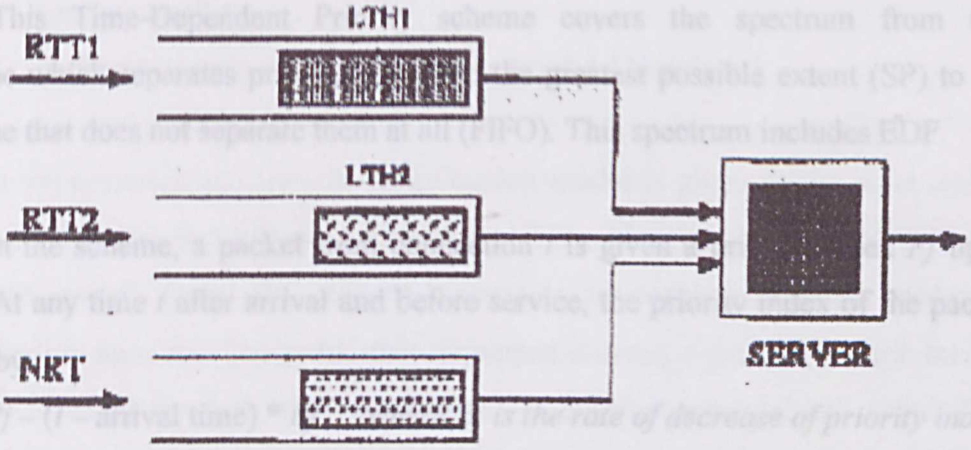


Figure 2.3 Minimum Laxity Threshold

2.6.1.2 Double Queue Length Threshold

For Double Queue Length Threshold algorithm proposed in this paper, under the hypothesis that different delay sensitivities of traffic classes according to the characteristics of services in non real-time traffic (NRT) be distinguishable, are set queue threshold value QTH 1 and QTH 2 corresponding to the two classes of non real time traffic 1(NRT 1) and non real time traffic 2 (NRT 2) respectively. Here, since there are two kinds of traffic class in AAL, two queue threshold values are needed. Under this Double Queue Length Threshold policy, the priority is given to the queue of which number of cell is larger than each threshold value QTH 1 and QTH 2.

2.6.1.3 Hysteresis effect Queue Length Threshold

The proposed Hysteresis Queue Length Threshold algorithm decides the service priority according to the old state value of buffer. Under the Hysteresis Queue Length Threshold policy, RTT is served until the number of non real time cell is equal to the threshold upperlimit. And non real-time traffic (NRT) is served when the number of non real time cell is more than threshold upperlimit until non real time cells are decreased to the point of the threshold lowerlimit.

2.6.2 Time Dependent Priority Scheduling

A Time-Dependent Priority (TDP) scheme is proposed in this paper [6], and its schedulability conditions and implementation are also presented.

This Time-Dependent Priority scheme covers the spectrum from that discipline which separates priority groups to the greatest possible extent (SP) to the discipline that does not separate them at all (FIFO). This spectrum includes EDF.

In the scheme, a packet from connection i is given a priority index Pj upon arrival. At any time t after arrival and before service, the priority index of the packet is given by

$Pj - (t - \text{arrival time}) * Bj$ *where Bj is the rate of decrease of priority index.*

No preemption is allowed and whenever the service facility (link) is free, the packet with the lowest priority index is chosen for the transmission. Whenever a tie for highest priority (lowest priority index) occurs the tie is broken by servicing packets with a higher rate of increase of priority first, and in case we still have a tie, the tie is broken by the FCFS rule.

The schedulability condition is free of input traffic model can be used. Further, the algorithm is capable of achieving up to maximum efficiency possible at each switch.

An implementation of complexity is proposed in this paper. The complexity is independent of the number of streams admitted. If the number of switch queues is limited, the number of packets scheduling per second is large and the link can be potentially fully utilized. With this switch implementation, the proposed queues are easily realizable.

2.6.3 Traffic-Controlled Rate-Monotonic Priority Scheduling of ATM Cells

In this paper [19] , an ATM cell multiplexer called the Traffic-Controlled Rate-Monotonic Priority Scheduling (TCRM) are proposed to realize performance guaranteed real-time communication on ATM networks.

The Traffic-Controlled Rate-Monotonic Priority Scheduling (TCRM) has the following function:

(i) provides bounded end-to-end delays which are essential for real-time communication

(ii) is simple enough to operate in high-speed ATM network

In the proposed scheme, the leaky bucket model is given as the input traffic description. Besides, this scheme also requires User Network Interface (UNI) and each ATM switch along the path to cooperate in order to provide real-time communication services. To ensure that requested channel *i* gets its service rate at every switch along the path, the switch needs to enforce a special cell scheduling policy.

2.7 Programming Language Approach

Several programming approaches can be taken as approach to the development of the ATM Network simulator. Therefore, it is a need to consider many advantages and disadvantages of several programming approaches. Here, procedural programming and object-oriented programming are both discussed.

2.7.1 Procedural Programming

In the early days of computing, programming was an extremely procedural approach. The procedural approach makes use of procedural languages, in which program codes were placed into blocks that are referred to as *procedures* or *functions*. With the use of procedural languages, tasks were broken down into separate blocks, in which separate blocks would perform separate tasks.

Procedure programming specifies an exact sequence or order of operation. That is, procedural program is written as a list of instructions, telling the computer, step-by-step, what to do: Open a file, read a number, multiply by 4, display something Program logic in procedural programming determines the next step to execute and this logic determination is made in response to conditions and user action. A function or procedure is a relatively simple program that is called by other programs and returns a value to the program that called it. The code can be put into blocks called procedures or functions. The goal of each of these blocks was to act like a black box which completed one task or another. Most traditional computer languages like Pascal, C and FORTRAN are procedural.

Procedural programming is fine for small projects. It is the most natural way to tell a computer what to do, and the computer processor's own language, machine code, is procedural, so the translation of the procedural high-level language into machine code is straightforward and efficient. What is more, procedural programming has a built-in way of splitting big lists of instructions into smaller lists: the functions [MONET].

But even in the case of a more complex logical flow, the main idea remains the same: a certain set of instructions is followed through from beginning to end, each step building upon and tied to every previous step. Procedural programming language is a powerful for solving particular problem. However, it is suitable for small and simple program only. When program become more complex, it will make the program source code harder to read and understand.

It was believed that functions could always be written without modifying external data. Essentially, this meant that building functions would be akin to building black boxes, which could be left to stand alone once completed and tested.

However, realistically, it was found to be difficult to construct functions in such a way that they do not modify data outside their boundary. Frequently, the restriction of not being able to change external data would turn out to be too constraining. Therefore, functions began to change data outside of their scope, for example with the use of pointers. This resulted in the problem of *coupling*, which in itself results in increasing difficulty to test. The presence of coupling meant that when each method cannot just be tested individually, for it must be made sure that the method did not corrupt external data. Ultimately, it meant that each black box needs to be tested with all its black boxes in place. If any black box were changed, the parent black box would have to be re-tested.

- Such complicated testing problems meant that for increasingly complex and large programs, the difficulty in testing increasingly became more unmanageable.

2.7.2 Object-oriented Programming

Object-oriented programming (OOP) is different from procedural programming language in several ways. Everything in OOP is grouped as *object*. Every object communicates with each other by sending message. Objects are the central idea behind OOP. A method is similar to a procedure. The basic idea behind an object is that of simulation. Most programs are written with very little reference to the real world objects the program is designed to work with; in object-oriented methodology, a program should be written to simulate the states and activities of real world objects. This means that apart from looking at data structures when modeling an object, we must also look at methods associated with that object, in other words, functions that modify the objects attributes.

Object-oriented programming (OOP) has its key component technologies – inheritance and polymorphism. Inheritance is a form of software reusability in which new classes are created from existing classes by absorbing their attributes and behaviors and embellishing these with capabilities the new classes require. Polymorphism is a character of assigning different meanings to a particular symbol or object, depending upon the context in which it is used. This allows objects to act differently within different situations, it enable the flexibility of a program design.

Encapsulation concept is also one of the features of OOP, it is the inclusion within a object of all the resources need for the object to function - basically, the data items and the methods. Methods are responsible for manipulating the data in order to reflect its behaviours. The public interface defines how to use this object. Other objects adhere to these interfaces to use the object without having to be concerned with how the object accomplishes it. The idea is "don't tell me how you do it; just do it".

The following describe some advantages of object-oriented programming language.

- ***Simplicity***: software objects model real world objects, so the complexity is reduced and the program structure is very clear.
- ***Modularity***: each object forms a separate entity whose internal workings are decoupled from other parts of the system.
- **Modifiability**: it is easy to make minor changes in the data representation or the procedures in an OO program. Changes inside a class do not affect any other part of a program, since the only public interface that the external world has to a class is through the use of methods.
- ***Extensibility***: adding new features or responding to changing operating environments can be solved by introducing a few new objects and modifying some existing ones.
- ***Maintainability***: objects can be maintained separately, making locating and fixing problems easier.
- ***Reusability***: objects can be reused in different programs [23]. Programming Language.

2.7.2.1 Java Programming

Java is just a small, simple, safe, object-oriented, interpreted or dynamically optimized, byte-coded, architecture-neutral, garbage-collected, multithreaded programming language with a strongly typed exception-handling mechanism for writing distributed, dynamically extensible programs. Java is developed by Sun Microsystems in California. It is a powerful programming language built to be secure, cross-platform and international [20].

Java is an object-oriented programming language especially designed for use in Internet environment. Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build applets for use as part of a Web page.

As in a modern software development, Java is object-oriented from the ground up. The point of designing an object-oriented language is not simply to jump on the latest programming fad. The object-oriented paradigm meshes well with the needs of client-server and distributed software. Benefits of object technology are rapidly becoming realized as more organizations move their applications to the distributed client-server model.

An important characteristic that distinguishes objects from ordinary procedures or functions is that an object can have a lifetime greater than that of the object that created it. This aspect of objects is subtle and mostly overlooked. In the distributed client-server world, it is possible to have potential for objects to be created in one place, passed around networks, and stored elsewhere, possibly in databases, to be retrieved for future work.[21]

The major advantages of Java are:

- ***Simple*** – Java is simple for building a system that could be programmed easily without a lot of esoteric training and which leveraged today's standard practice.
- ***Object-oriented*** – Object-oriented facilities of Java are essentially those of C++, with extensions from Objective C for more dynamic method resolution.
- ***Network-Savvy*** – Java has an extensive library of routines for coping easily with TCP/IP protocols like HTTP and FTP. This makes creating network connections much easier than in C or C++.
- ***Robust*** – Java objects can contain no references to data external or other known objects. This ensures that an instruction cannot contain the address of data storage in another application or in the operating system itself, either of which would cause the program and the operating system itself to terminate or "crash". Garbage collection is another powerful feature that makes no chance for a Java program to corrupt the memory via a dangling pointer. The Java virtual machine makes a number of checks on each object to ensure integrity.

- ***Secure*** – Java's run-time system performs checking to ensure that programs transmitted over a network have not been tampered with. The code produced by the Java compiler is checked for validity, and the program is prevented from performing unauthorised actions.
- ***Architecture neutral*** – Networks are composed of a variety of systems with a variety of CPU and operating system architectures. To enable a Java application to execute anywhere on the network, the compiler generates an architecture-neutral object file format – the compiled code is executable on many processors, given the presence of the Java runtime system.
- ***Portable*** – Java program is compiled into bytecode instead of bitcode. This feature make Java can be run anywhere in a network on a server or client.
- ***Interpreted*** – Java bytecodes are translated on the fly to native machine instructions (interpreted) and not stored anywhere. Since linking is a more incremental and lightweight process, the development process can be much more rapid and exploratory.
- ***High performance*** – In addition to being executed at the client rather than the server, a Java applet has other characteristics designed to speed up the execution.
- ***Multithread*** – Java has built-in support for multitasking. A Java program may create any number of threads, which appear to execute in parallel.
- ***Dynamic*** – Java is a more dynamic language than C or C++. It was designed to adapt to an evolving environment [24][25].

2.7.2.2 C++ Programming

C++, an extension of C, was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories. C++ provides a number of features that "spruce up" the C language, but more importantly, it provides capabilities for *object-oriented programming*. C++ is a hybrid language – it is possible to program in C++ in either C-like style, or both.

The C++ language facilities structured and disciplined approach to computer program design. C++ programs consist of pieces called *classes* and *function*. Programmer can program each piece that programmer need to form a C++ program. But most C++ programmers take advantage of the rich collections of existing classes and functions in the C++ standard library.

C++ programs typically go through six phases to be executed. These are: *edit*, *preprocess*, *compile*, *link*, *load*, and *execute*.[22]

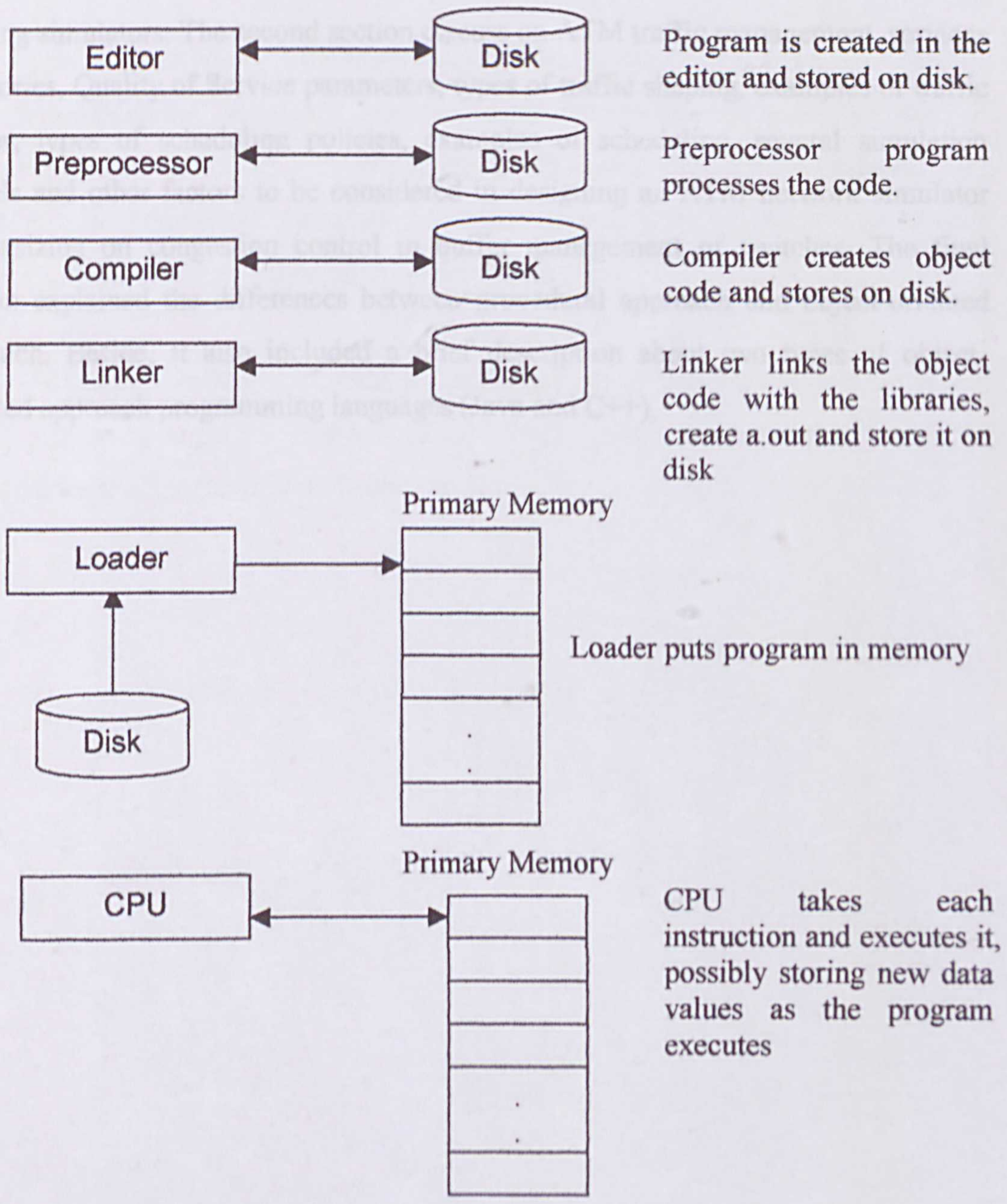

Editor
Disk
Program is created in the editor and stored on disk.
Preprocessor
Disk
Preprocessor program processes the code.
Compiler
Disk
Compiler creates object code and stores on disk.
Linker
Disk
Linker links the object code with the libraries, create a.out and store it on disk
Primary Memory
Loader
Disk
Loader puts program in memory
Primary Memory
CPU
CPU takes each instruction and executes it, possibly storing new data values as the program executes

Figure 2.4 A Typical C++ Environment

2.7.2.3 Java Programming Tools

There are many types of Java programming tools available in market. One can develop Java programming using pure Java SDK without the supporting of integrated development environment (IDE), or just selects tools like Microsoft J++, Borland JBuilder, or Symantec Visual Café.

2.8 Chapter Summary

This chapter has three main sections. First section introduces and analyzes on existing simulators. The second section discuss on ATM traffic management, services categories, Quality of Service parameters, types of traffic shaping, examples of traffic shaper, types of scheduling policies, examples of scheduling, several simulation models and other factors to be considered in designing an ATM network simulator emphasizing on congestion control in buffer management of switches. The final section explained the differences between procedural approach and object-oriented approach. Beside, it also included a brief description about two types of object-oriented approach programming languages (Java and C++).

CHAPTER 3: JAVASIM AND PRIORITY SCHEDULING ALGORITHMS OVERVIEW

3.1 JAVASIM

Javasim is an ATM simulator tool that gives the user an interactive modeling environment with a graphical user interface. With this tool the user may create different network topologies, control component parameters, measure network activity, and log data from simulation runs.

The JavaSim object is the main object of the simulator. It keeps a list of all the network components (all are descendents of SimComponent), and a list (a queue) of all events (in the form of SimEvent). Every component contains a set of parameters (all inherit SimParameter). All other classes are mostly helpers that provide certain services such as time service, logging, meter display, etc.

All classes within the dotted rectangle belong to the simulation engine and should not be changed by component developers. Each of these object, will be discussed in detail in the following section. [26]

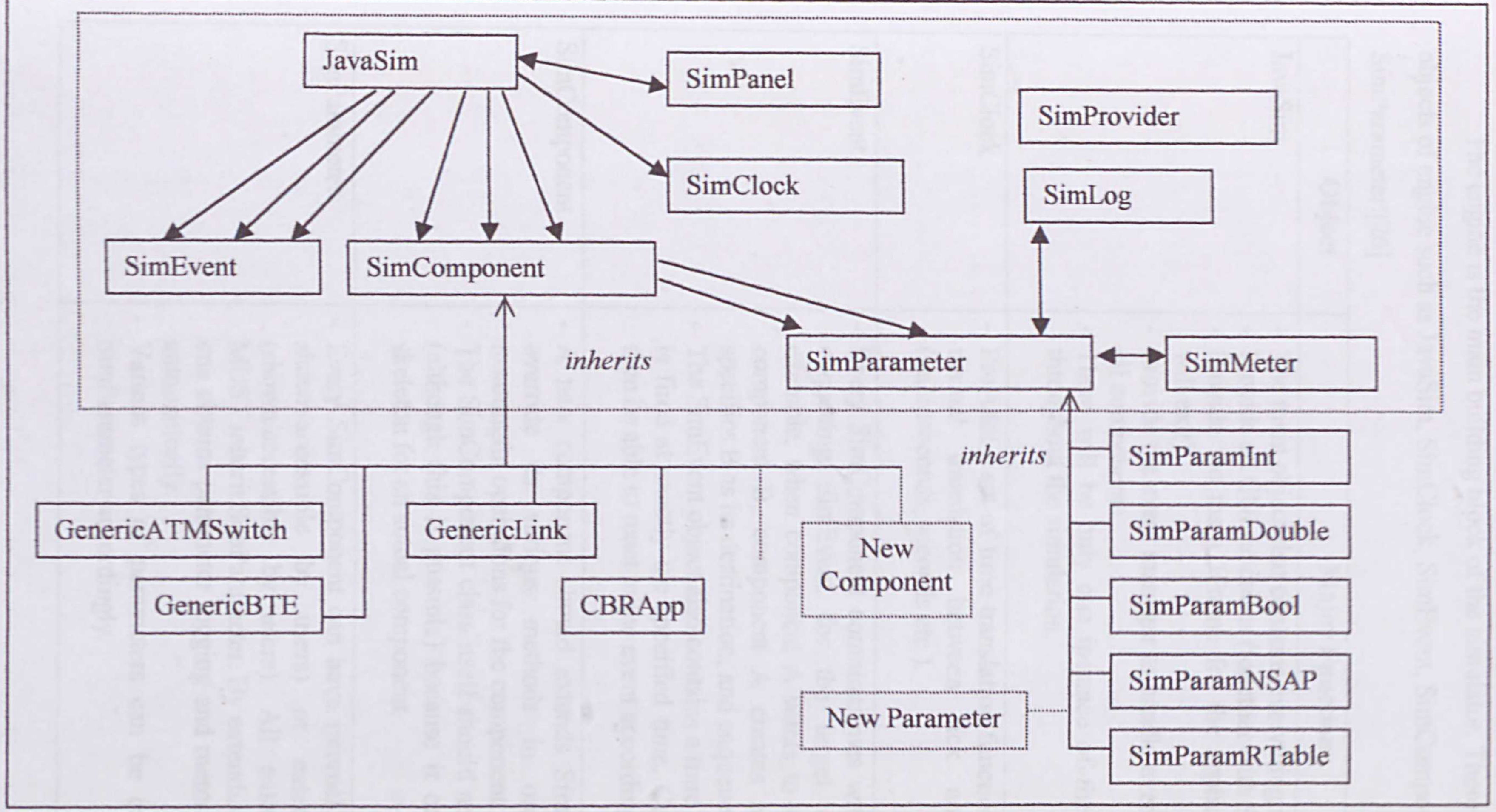


Figure 3.1 Hierarchy of all the significant objects in the Java Network Simulator

3.1.1 Simulation Engine

The engine is the main building block of the simulator. There are different objects of engine such as JavaSim, SimClock, SimEvent, SimComponent and SimParameter.[26]

Object	Major Functions
JavaSim	- The main object that contains everything in the simulator - Provide all GUI functions (together with SimPanel) - Provide the main JFrame for the application (closing it will exit) - Provide the event manager to handle event-passing among all components - There will be only one instance of the JavaSim object throughout the simulation.
SimClock	- Provides a set of time translation functions (all static) for normal translation between tick and actual time (microseconds, seconds etc.).
SimEvent	- Every SimComponent communicates with each other by enqueuing SimEvent for the target component. For example, when component A wants to send a packet to component B, component A creates a SimEvent that specifies B as its destination, and enqueue the event. - The SimEvent object also contains a time so that this event is fired at exactly the specified time. Component B will then be able to react to the event accordingly.
SimComponent	- A new component should extends SimComponent and override its various methods in order to provide meaningful operations for the component. - The SimComponent class itself should not be instantiated (although this is possible) because it only provides the skeleton for an actual component.
SimParameter	- Every SimComponent can have internal parameters (not shown/accessible by users) or external parameters (shown/accessible by users). All external parameters MUST inherit SimParameter. By extending SimParameter, one obtains parameter logging and meter display features automatically. - Various types of parameters can be create by extend SimParameter accordingly.

3.1.2 Simulation Component

The component is the basic building block of the simulator. There are different objects of components such as switches, physical links, terminal equipment, and ATM applications.[11]

Object	Major Functions
Switch	- The switch is the component that switches or routes cells over several virtual channel links. - When a switch accepts an incoming cell from a Physical Link it looks in its routing table to determine which outgoing link should send it. If the outgoing link is busy, the switch will queue the cells destined for that link and not send them until free cell slots are available for transmission.
Link	- This component simulates the physical medium (copper wire or optical fiber) on which cells are transmitted.
Broadband Terminal Equipment (B-TE)	- The B-TE component simulates a Broadband ISDN node, e.g., a host computer, workstation, etc. - A B-TE component has one or more ATM Applications on one side and a physical link on the other side. Cells received from the Application side are forwarded to the physical link; if the link is busy the cells go into a queue.
Application	- The ATM Applications are logical entities that run on B-TE (hosts). - This is a component to emulate the behavior of an ATM application at the end-point of a link. - The Applications may be considered as traffic generators that are capable of emulating variable (VBR) or constant bit rate traffic (CBR) sources. - For variable bit rate applications the user sets the burst length, and interval between bursts. - For lower priority traffic, the user may create an available bit rate (ABR) application.

3.2 PRIORITY SCHEDULING ALGORITHMS

A priority scheduling algorithm is used to determined which connection are served first. The main design objective for the scheduler is to maintain fairness. The scheduler allocates resources to the virtual channel (queue) according to their priority class. The scheduler must serve every Virtual Channels (VCs) of the same priority class with equal probability. VCs of higher priority class are served before those of the lower priority classes.

For most of the priority scheduler examples, the performance measures are cell delay, queue length and cell loss ratio (CLR) .The scheduler algorithms strongly influence the cell delay which is most important for delay sensitive services.

The scheduling algorithm is the component that determines which queue is given the opportunity to transmit a cell that is stored in the buffer. The ideal algorithm would have properties of *efficiency* and *fairness*. The aspect of *efficiency* can be easily measured. However, *fairness*, is not so easily understood. In a system where each class of service has different requirements for acceptable latency and for cell loss that can be tolerated, determining which metrics to use for *fairness* is a bit more subjective. In our attempt to provide fairness, we consider a class of service to be treated fairly if it continues to be serviced and its requirements for cell delay and cell loss are fulfilled. Some of the simplest scheduling algorithms are first-in-first-out (FIFO), round robin (RR), and a "fixed priorities" scheme where a queue with higher priority is always served before a queue with a lower priority. [5]

One algorithm (*Hysteresis effect Queue Length Threshold)* is proposed for scheduling the ATM traffic within the different priority class and two algorithms (*Round Robin and Select Largest Queue*) are proposed for scheduling the ATM traffic within the same priority class.

3.2.1 Hysteresis effect Queue Length Threshold (HQLT)

Hysteresis effect Queue Length Threshold algorithm is proposed as Dynamic Priority Scheduling techniques for advanced processing of multiple class traffics. HQLT algorithm decides the service priority according to the old state value of buffer.

Under the HQLT policy, Real Time Traffics (CBR and rt-VBR) is serve when the total number of Non Real Time Traffic (NRT) cell is less then to the threshold lower limit. And NRT queue is served when the number of NRT cells is more than threshold upper limit until NRT cells are decreased to the point of the threshold lower limit.

HQLT algorithm is aim to makes an output trunk become more stable and upgrade the QoS with upper limit (QTH_U – Queue Length Threshold for Upper Limit) and lower limit threshold (QTH_L– Queue Length Threshold for Upper Limit).[17]

3.2.2 Round Robin

Round Robin algorithm is one of the algorithms that proposed for scheduling the ATM traffic within the same priority class in this project.

The Round Robin (RR) scheduling algorithm treats all connections equally. The job sequencing by the scheduler is as follow:

- Every connection is assigned a queue Q_i, and the queues are served in round robin fashion. Thus after a serving Q_i, the server moves to serve queue Q_{i+1}. Every time that a queue is selected, the server transmits one cell from the queue (if the queue is not empty) before moving to next queue. For example, after transmitting some cells from the second queue in class p=4, the scheduler will start checking the third queue next time it return to category p=4. Thus, each connection gets a guaranteed minimum service rate of 1/N, where N is the number of connections.[18]

In contrast with FCFS, RR is not biased against short messages and provide a certain isolation for each connection. In addition, RR resulted in a smaller average delay as compared to FCFS. RR can be implemented with low run time overheads, although it has higher complexity than required for FCFS.

3.2.3 Select Largest Queue

Select largest queue algorithm is other algorithm that proposed for scheduling the ATM traffic within the same priority class in this project.

First, a search for a non-empty queue of highest priority class is executed. If all queues in highest priority class are empty, then the queue for the lower priority class is examined. Generally, only if all queues in priority class p are zero, the queue of the next lower priority class (p+1) is examined. In case all the queues are empty, in the next time slot the search starts again at first queue of highest priority class.

If at same priority class there exist a non-empty queues, then the largest queue in the priority class will be allowed to send cell to its destination. After that a new search for non-empty queue will always start with the table of highest priority class in order to guarantee that high priority cells are scheduled and transmitted before the others.[7]

3.3 Chapter Summary

First section of this chapter discusses the two important part of Javasim ATM Simulator: *Simulator Engine and Simulator Components*. The second section discusses proposed priority scheduling algorithms to schedule the outgoing cells at the output port of a switch for different class of service and same class of service.

CHAPTER 4: SYSTEM ANALYSIS

4.1 Programming Techniques and Language Choice

4.1.1 Object-oriented Programming Techniques

Object-oriented Programming (OOP) is the technique chosen for project development. It is because OOP provides a good practice in programming with a lot of benefits compare to procedural programming techniques. The advantages of object-oriented programming language include simplicity, modularity, modifiability, extensibility, maintainability, and reusability,

4.1.2 The Java Programming Language

As an object-oriented language, Java draws on the best concepts and features of previous object-oriented languages, primarily Eiffel, SmallTalk, Objective C, and C++. Java goes beyond C++ in both extending the object model and removing the major complexities such as pointer-referencing problem of C++. With the exception of its primitive data types, everything in Java is an object, and even the primitive types can be encapsulated within objects if the need arises. Besides, only Java supports for multithreading, platform and browser independent, and is object-oriented.

In Java, only single inheritance is supported but multiple implementations of interface class is allowed [20]. Security and safety are main features of Java programming language. Its execution semantics guarantees that every run-time error is detected and reflected in a throw exception. Java eliminated the use of pointers of C++ and replaced it with references, which prevents program from accessing illegal areas of the system's memory. Besides, Java also supports dynamically run time method identification. Libraries of Java is supported through the use of packages and allow complicate programs to be build but the overhead of keeping track of all libraries is reduced.

Concurrency is very important in a simulation model as there might be many objects doing their own process at the same time. It is impossible to let them execute in a sequential methods, as this could not be appropriate as compared to real time simulation result. Most of the programming languages do not enable programmers to specify concurrent activities, rather they provide only simple set of control structures where one action is performed after one another [21]. In Java, programmer specifies that application contains threads of execution and the program may execute concurrently with other threads. These powerful capabilities are not available in C and C++. Instead, in C and C++ they have single-threaded languages.

4.2 Development Tools

After reviewing and analysis the requirement needed for the system, possible development tools are analyzed. Then the most suitable tools are decided. The development includes platform, development environment software and the programming language tools. The interoperability among the tools is considered when these tools are picked. Followings are brief explanation of the selected tools.

4.2.1 Window 98

Window 98 is used as the operating system for the whole system. It is known to be more users friendly and stable than other server based operating system. All the other developing tools will be running on this operating system.

4.2.2 Borland Jbuilder

This section describes about Borland JBuilder as the selected tool for development.

JBuilder is a group of highly productive tools for creating high-performance, platform-independent applications for Java. It is designed for all levels of development of projects, ranging from applets and applications that require networked database connectivity to client/server and enterprise-wide, distributed, mutli-tier computing solution.

The JBuilder IDE supports a variety of technologies including:

- 100% Pure Java.
- JavaBeans.
- Java 2.

- Java SDK 1.2.2.
- JFC/Swing.

The additional technologies supported by JBuilder Professional edition are:

- Servlets.
- Remote Method Invocation (RMI).
- Java Database Connectivity (JDBC).
- Open Database Connectivity (ODBC).
- All major corporate database servers.

The additional technologies supported by JBuilder Enterprise are:

- Enterprise JavaBeans (EJB).
- JavaServer Pages (JSP).
- Common Object Request Broker Architecture (CORBA).

JBuilder also provides developers with a flexible, open architecture that makes it easy to incorporate new SDKs, third-party tools, add-ins, and JavaBean components [27].

4.3 System Requirements

4.3.1 Functional Requirements

This traffic shaping simulation is aim to provided user interface environment to make the simulation model become more attractive. Therefore, the functional requirement of this system must be more users friendly.

4.3.1.1 Topology Design System

The simulation system will provide a desktop to let user design the topology of a traffic shaping simulation.

4.3.1.2 User Input System

Before the simulation started, user would give a chance to enter or select some values. For example, user can choose one of the scheduling policies provide, select the different type of service classes and initial the queues size.

4.3.1.3 Generate Output

The simulation system will generate the output traffics. A graphic user interface design will use to show how the simulation process is going on.

4.3.2 Non-Functional Requirements

In order to produce the quality of system, certain software quality factors must be conform. Therefore, the proposed simulation for the ATM simulator will follow these non-functional requirements:

4.3.2.1 Flexibility

The system must be able to incorporate new technologies in the future and in fast changing environment. These technologies includes:

- Object oriented technology
- Advance security technology

4.3.2.2 Usability

The system must be user friendly. User must be able to use the system in the shortest learning curve. They can be customizes to meet the need of changing business rule and process. Interfaces must be self explainary and consistent with other application in the environment.

4.3.2.3 Correctness

The final application must meet the objective, specification and requirement of the users.

4.3.2.4 Scalability

This system must be capable of migrate or move from machine with different specification, with minimum or no changes to the underlying component. It must be able to meet this requirement as the basic structure of hardware and software environment is changing constantly.

4.3.2.5 Reusability

This components and different part of this system must be capable to reuse. Components and parts are required to be self contain in order to archive reusability. This requirement is important as to support future redesign or expansion of current system.

4.3.2.6 Portability

This requirement will enable the application to work on various platform, hardware and operating system. Components are designed to ensure migration of component does not or only require minimum modification, recompiling, reconfiguration or redesign.

4.3.2.7 Maintainability

The application is designed so that the effort required to maintain, locate and fix an error in the program is minimum.

4.3.2.8 Manageability

Application should be capable of being manage and operate easily.

4.4 Chapter Summary

This chapter discusses the programming language and development tools chosen to create the network simulator components. Beside that, brief description about functionality requirements, non-functionality requirements and development tools for simulation is also available.

CHAPTER 5: DESIGN OF ATM NETWORK SIMULATION

5.1 THE SIMULATION MODEL

5.1.1 Overview

Mainly the design of the ATM Network Simulation is base on the Javasim ATM Simulator. The important feature of this simulator design is the use of inter-object communication for the execution of the simulation logic, i.e. innovation of methods exported by each object instead of message passing. The combination of distributed object programming architectures of Java allows user to transform the tool to a distributed simulator, by spreading the computational object across other program. The simulator is extensible to a built another simulator as its components can be easily communicate with other external components.

5.1.2 Output Ports of Switch

Congestion could happen at the output port if the switch's switching rate is too high compare to the link connected at the output port. Also, when there are too many application sending data to the same target destination. Meaning that, the queues of output ports at the switch is highly demanded and needs to transfer out the cells in a high rate. But unfortunately it is difficult to send out the cells to the link, as the link rate is slower.

Therefore, the priority scheduling algorithms are introduced to manage the output ports traffic. The priority scheduling algorithms will be implement at demultiplex part and schedule output cells part

5.1.3 Multithreading

Multithreading could be one of the most interesting and important feature of this model. Every component classes in this simulation act as a single thread and they are executed concurrently during run time process. A general clock is used to control the generation of next tick for each process to execute. This happens when not every process could complete their execution at the same time with other process. Therefore, they are designed to run concurrently and at the same time waiting for each other during execution time.

5.2 QUEUE ARCHITECTURE DESIGN

The queue architecture determined the queuing model used in this simulation and priority scheduling algorithms used for demultiplexing and scheduling cell at output port.

The algorithms use includes HQLT, Static Priority and Select Largest Queue.

5.2.1 Output Port Queuing Model

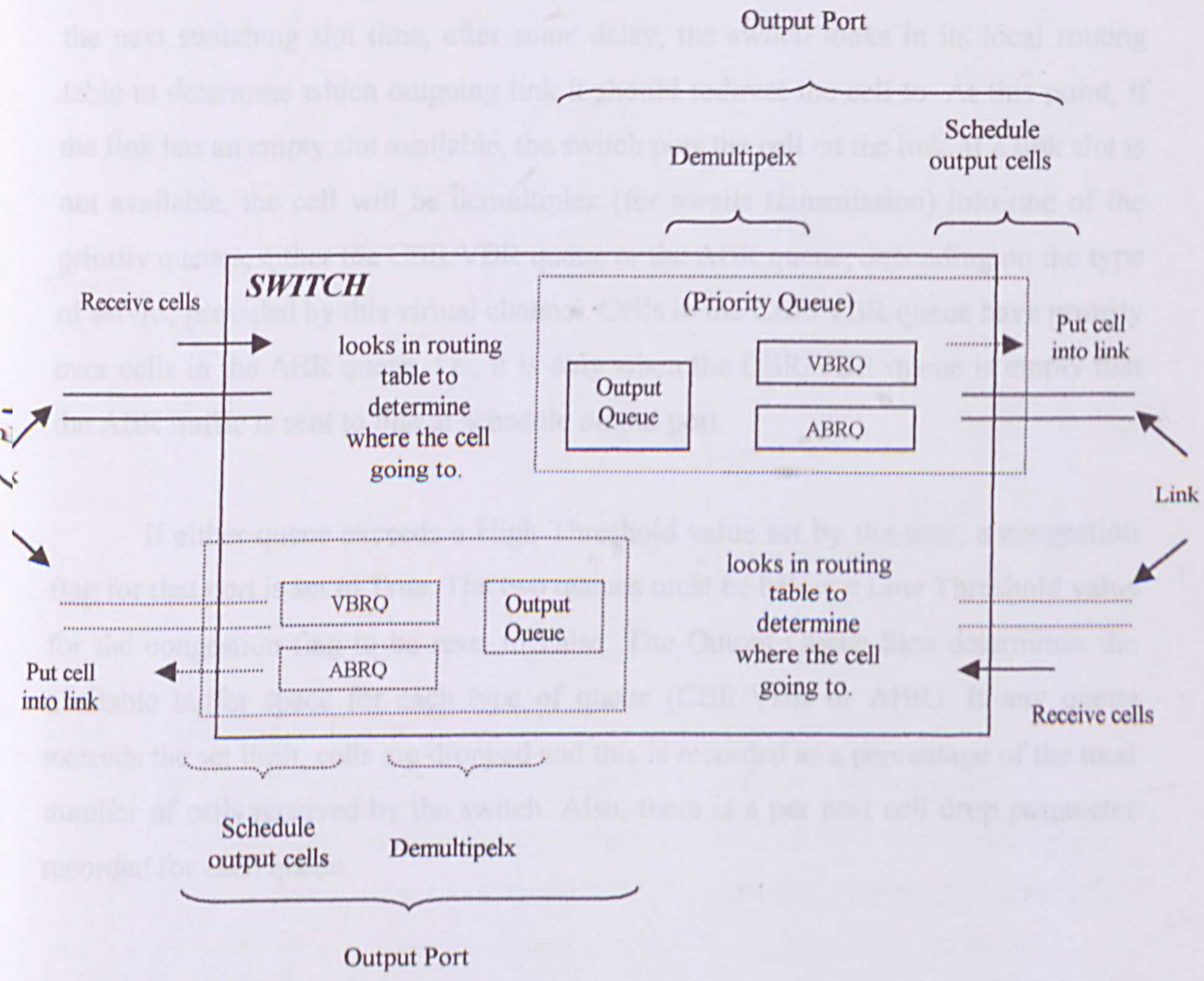


Figure5.1 Switch Architecture With Output Port Queuing Model

Switch is the component that switches or routes cells over several virtual channel links. A local routing table is provided for each switch. This table contains a route number (that is read from incoming cell structure and is the equivalent of the cell's virtual channel identifier), a next link entry, and a next switch/next B-TE entry.

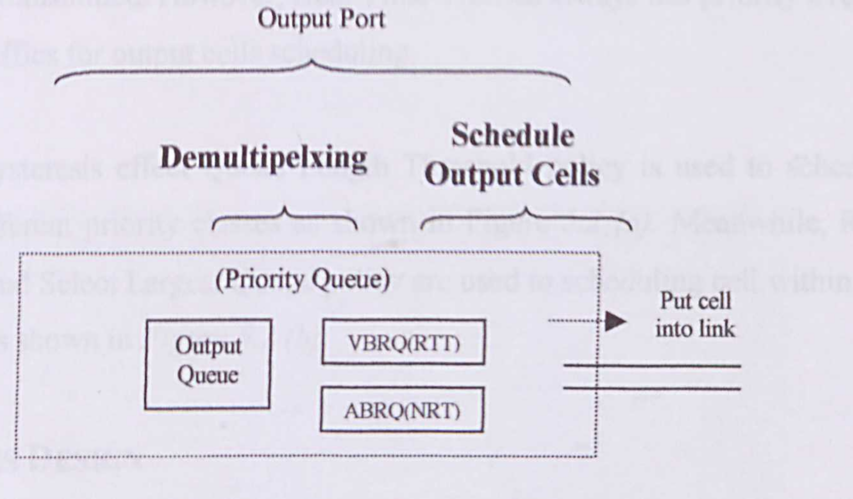
From *Figure 5.1*, output port is one of the components of a switch. Since switch is model as a thread, the processes of an output ports are controlled under a switch. A switch can have several output port attached to several links. Here, output port is built as a component where the switch can have as many output port as it wish (depend to the connection between switch and link).

First of all, let's consider a cell arriving at the switch from a physical link. At the next switching slot time, after some delay, the switch looks in its local routing table to determine which outgoing link it should redirect the cell to. At this point, if the link has an empty slot available, the switch puts the cell on the link. If a link slot is not available, the cell will be demultiplex (for awaits transmission) into one of the priority queues, either the CBR/VBR queue or the ABR queue, depending on the type of service provided by this virtual channel. Cells in the CBR/VBR queue have priority over cells in the ABR queue, i.e., it is only when the CBR/VBR queue is empty that the ABR traffic is sent to link at schedule output port.

If either queue exceeds a High Threshold value set by the user, a congestion flag for that port is set to True. The two queues must be below a Low Threshold value for the congestion flag to be reset to False. The Output Queue Size determines the available buffer space for each type of queue (CBR/VBR or ABR). If any queue exceeds the set limit, cells are dropped and this is recorded as a percentage of the total number of cells received by the switch. Also, there is a per port cell drop parameter recorded for each queue.

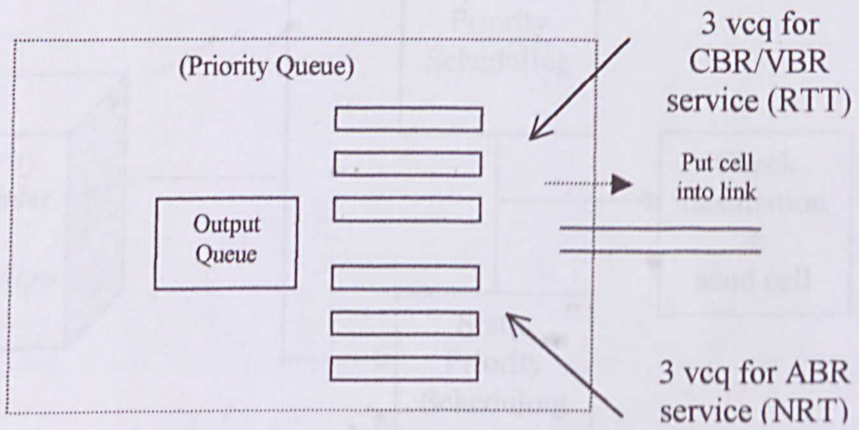
5.2.2 Priority Scheduling At Output Port

Priority scheduling can be implement by the use of cell demultiplexing and output cell scheduling as shown in *Figure 5.2*.



(a) Within Different Priority Classes

OR



(b) Within Same Priority Class

Figure5.2 Methods of Cell Demultiplexing At Output Port

5.2.2.1 Demultiplexing

There are two methods use to demultiplex the cell into different queues:

- Demultiplex into two queues only, that are ABRQ for Non Real Time Traffic service category (ABR (Available bit Rate)) and VBRQ for Real Time Traffic service category - (VBR (Variable Bit Rate) or CBR(Constant Bit Rate)).

  or

- Demultiplex into 3 VCQ (Virtual Channel Queue) for each type of service, (ABR and VBR/CBR services).

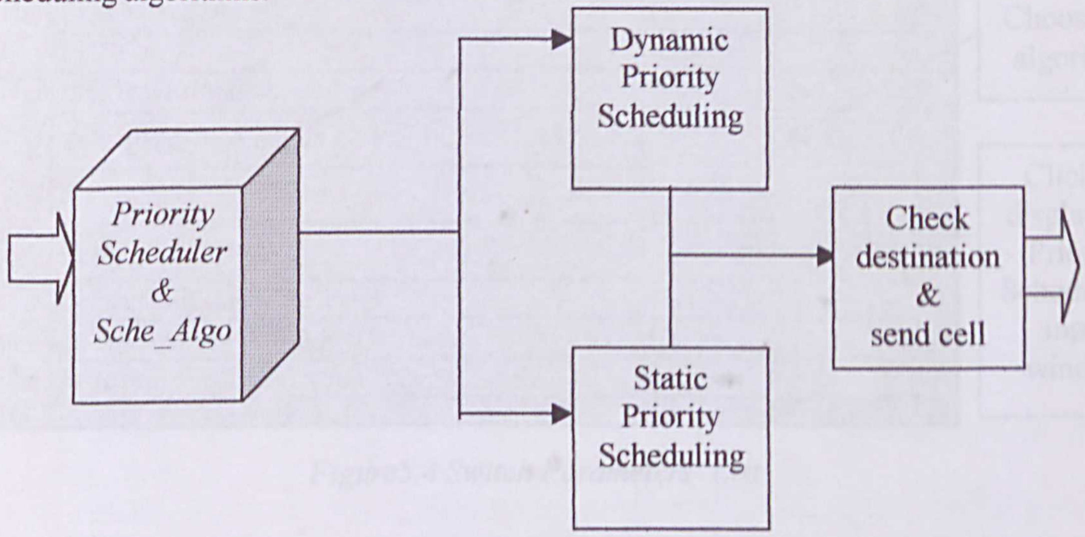
5.2.2.2 Schedule Output Cells

Cells in queues will be scheduled from output port of switch into physical links. Therefore, priority scheduling algorithms is used to decide cell from which queue will be transmitted. However, Real Time Traffics always has priority over Non Real Time Traffics for output cells scheduling.

The Hysteresis effect Queue Length Threshold policy is used to scheduling cell within different priority classes as shown in Figure *5.2 (a)*. Meanwhile, Round Robin policy and Select Largest Queue policy are used to scheduling cell within same priority class as shown in *Figure 5.2 (b)*.

5.3 Modules Design

The diagram below shows the modules design for implementation of priority scheduling algorithms.



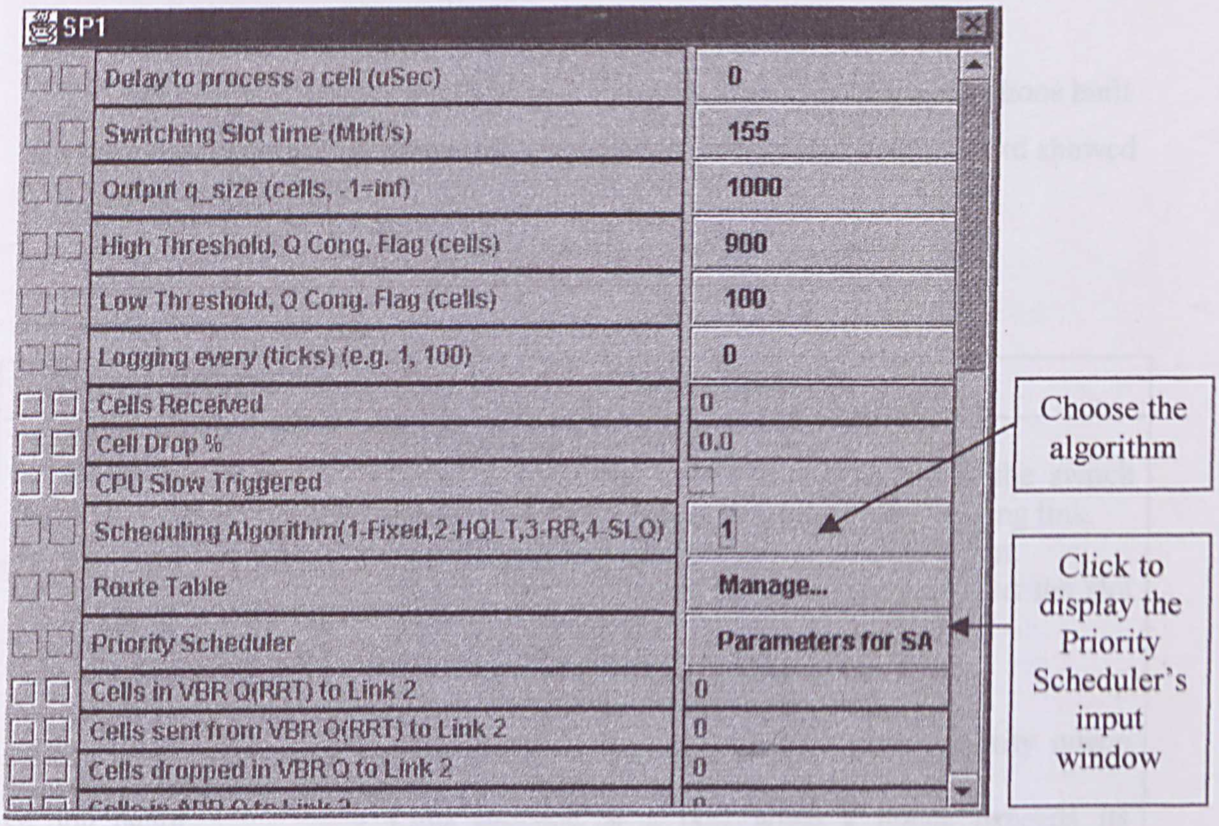
Priority Scheduler & Sche_Algo
Dynamic Priority Scheduling
Static Priority Scheduling
Check destination & send cell

Figure5.3 Modules Design

Module	Function
Priority Scheduler (see Figure5.5 for GUI design) & Sche_Algo(see Figure5.4 for Switch Parameters' List)	Gets the user input parameters need for different priority scheduling algorithms and let user choose the prefer algorithm
Dynamic Scheduling	Two dynamic scheduling policies are provided, that is ***Hysteresis effect Queue Length Threshold*** policy. Processes the outgoing cells, and administrates the queues system (number of cells and so on).

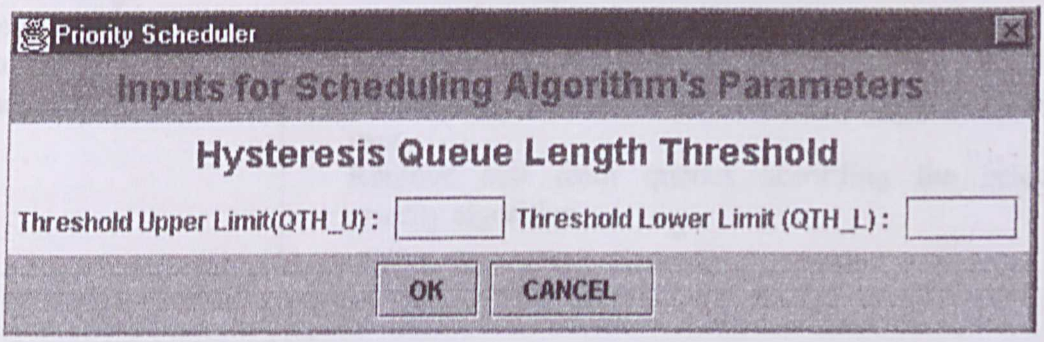
Static Scheduling	Two static scheduling policies are provided, that is ***Round Robin*** policy and ***Select Largest Queue*** policy Processes the outgoing cells, and administrates the queues system (number of cells and so on).

Table 5.1 Modules Function



Delay to process a cell (uSec)	0
Switching Slot time (Mbit/s)	155
Output q_size (cells, -1=inf)	1000
High Threshold, Q Cong. Flag (cells)	900
Low Threshold, Q Cong. Flag (cells)	100
Logging every (ticks) (e.g. 1, 100)	0
Cells Received	0
Cell Drop %	0.0
CPU Slow Triggered	☐
Scheduling Algorithm(1-Fixed,2-HQLT,3-RR,4-SLQ)	1
Route Table	Manage...
Priority Scheduler	Parameters for SA
Cells in VBR Q(RRT) to Link 2	0
Cells sent from VBR Q(RRT) to Link 2	0
Cells dropped in VBR Q to Link 2	0

Figure5.4 Switch Parameters' List



Inputs for Scheduling Algorithm's Parameters

Hysteresis Queue Length Threshold

Threshold Upper Limit(QTH_U) : Threshold Lower Limit (QTH_L) :

Figure5.5 GUI Design for Priority Scheduler's Input Window

5.4 COMPONENT CLASS DESIGN

The simulator is sub-divided into two major classes: the resources class and the executor class. The resources class is a class, which provides resources to execute the simulator, in this case, ABRApp, CBRApp and VBRApp is a resource class and it is used to generate cells flowing through the network simulator. An executor class is a class that execute and running a process using the resource classes during the simulation period.

The design of classes shows several major attributes and major functions built into the main classes of the simulator. The functions are defined in brief word showed in the tables appear below.

SPATMSwitch

Major Attributes	Major Functions
sw_delay sw_oqsize sw_ht sw_lt sw_speed sw_dropped; num_dropped,total_cell sw_speedup sw_cpucong sw_cells_received sw_log_factor sw_route_table sw_ps_window sw_ScheAlgo voports records randgen	- Receive cell from links - Increase time after the arrival of a cell at the switch before the switch places the cell on the outgoing link. - Switch cell from an input port to an output port. - Initial queues for each virtual output port start the slot time reference cycle - Set the buffer space for a queue - Select the priority scheduling algorithm - Cells awaiting transmission in a given priority queue (ABRQ or VBRQ). - Cells dropped at a port when a queue exceeds its maximum size. - For each port, the congestion flag is set when a queue exceeds its High Threshold value, cleared when both queues fall below the Low Threshold. - The outgoing cells is demultiplex to ABRQ or VBRQ or VCQ according to the sw_ScheAlgo's value input by user - Remove cell from queues according the selected priority algorithm.

SimParamPScheduler.java

Major Attributes	Major Functions
QTH_U QTH_L TitlePanelx	- Display a input window for Priority Scheduler - Prepare the input panel, Jlabel and JtextFiled for parameters of different priority scheduling algorithms

InputPanelx	such as Hysteresis Queue Length Threshold

GenericLink.java

Major Attributes	**Major Functions**
ln_speed ln_distance link_neighbors	- User may initial the link speed with different standard rates. - User also specifies the length of the link. - The output parameter reported by the simulator is link utilization in terms of bit rate (Mbits/s).

GenericBTE.java

Major Attributes	**Major Functions**

b_oqsize;
b_cellReceive
b_log_factor;
b_cell_count;
voports;
records;
app_ports;
	- If no slot is available for immediate transmission, the cell will queue in one of two queues, a VBR/CBR queue or an ABR queue.
	- The user can specify the maximum output queue size; if either queue exceeds this limit cells will be dropped.
	- The parameters that can be monitored for a B-TE are the number of cells in an output queue and the number of cells dropped at each queue. Also, the total number of cells received from the network may be monitored.

CBRApp, VBRApp and ABRApp

The CBRApp, VBRApp(rt-VBR) and ABRApp are three main components of Javasim that will generate cells for different class of service.

Major Attributes	**Major Functions**
cn_bit_rate cn_burst_length cn_int_bet_burst cn_start_time cn_trans_size cn_repeat cn_delay cn_random_size cn_random_delay cn_random_target cn_thisport cn_destnsap cn_destport cn_conattempt cn_conaccept	- Generate CBR/VBR/ABR applications - *For CBR application, the bit rate is fix* - *For VBR (rt-VBR) application, the bit rate can be fix or varies with time, that is it can be characterised as somewhat bursty* - User can send cell to other destination by specify the - destination NSAP and port - transmission size - delay between each cell OR User can let the simulator generate the above parameters randomly - Set up new connection when status of message type is NULL - Send data cells when status of message type is ACTIVE

cn_num_sent cn_status randgen	

5.5 Chapter Summary

First section of this chapter discusses the overview of the simulation model include the overview, output ports of switch and multithreading technique. Beside that, a queue architecture design and a module for implementation of priority scheduling algorithms also included. The second section includes description tables for *SPATMwitch* class, *GenericLink* class, *GenericBTE* class and *Application* class which are built for implementation of different priority scheduling algorithms.

CHAPTER 6: IMPLEMENTATION

The main purpose of this ATM network simulator is to test and compare different priority scheduling algorithms and to ensure that the queue could be managed in an appropriate manner using certain congestion control policy. Therefore in this simulation, model is built using object-oriented method and concept to ensure the flow of ATM cells in the simulator could communicate within objects.

6.1 SYSTEM IMPLEMENTATION

The switching process is started when a cell arriving at the switch from a physical link. At the next switching slot time, after some delay, the switch looks in its local routing table to determine which outgoing link it should redirect the cell to and the cell will be put into a spq. At this point, if the link has an empty slot available, the switch puts the cell on the link. If a link slot is not available, the cell will be go through a demultiplexing and schedule output process, depend the user selects priority scheduling algorithm. However, cells in the CBR/VBR queue have priority over cells in the ABR queue, i.e., it is only when the CBR/VBR queue is empty that the ABR traffic is sent to link at schedule output port. (The flow of cell is described in *Figure 6.1*).

Therefore, before the simulation start, user needs to choose the priority scheduling algorithm and initial suitable threshold values for related algorithm.

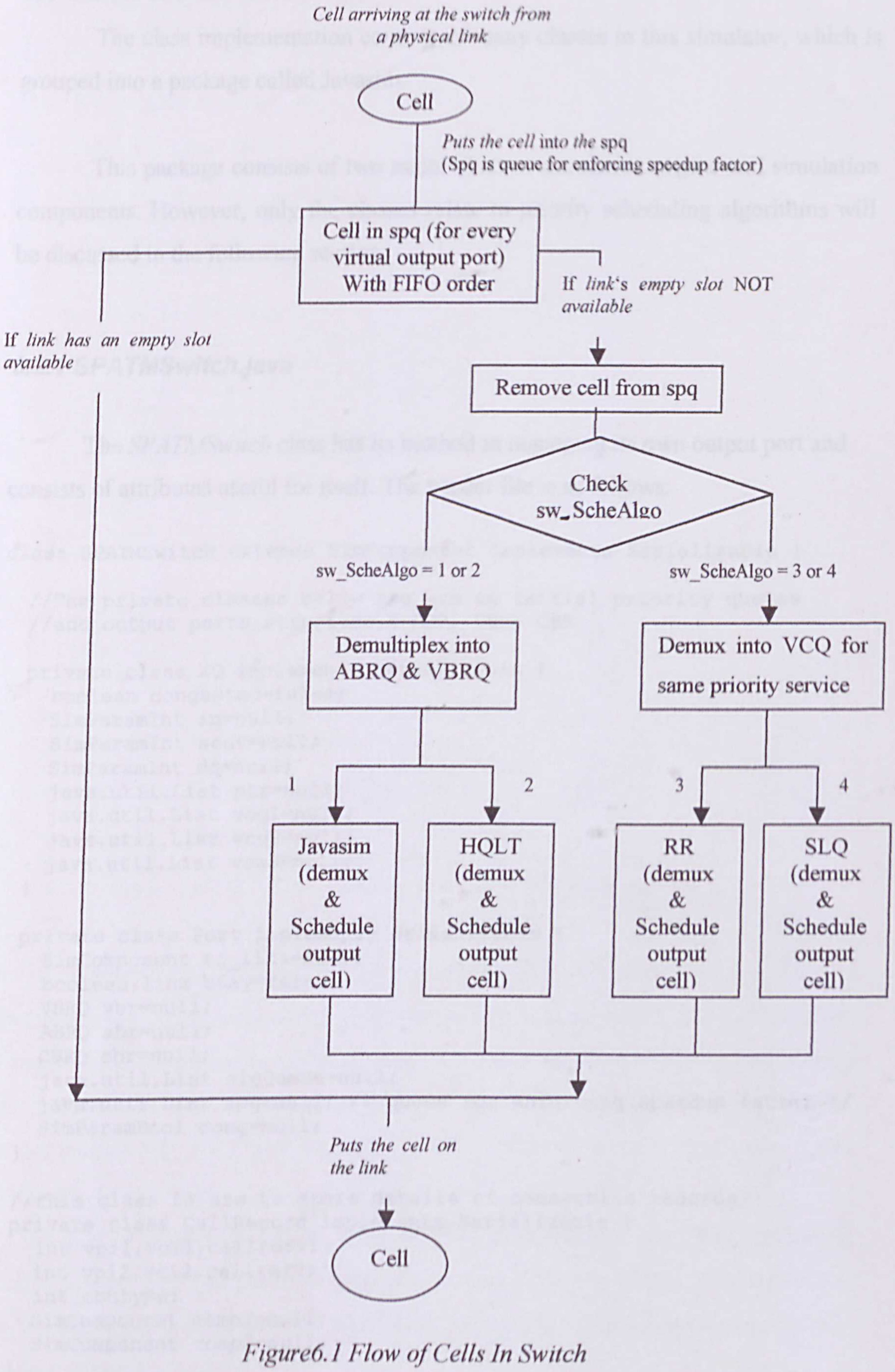

Cell arriving at the switch from a physical link
Cell
Puts the cell into the spq (Spq is queue for enforcing speedup factor)
Cell in spq (for every virtual output port) With FIFO order
If link has an empty slot available
If link's empty slot NOT available
Remove cell from spq
Check sw_ScheAlgo
sw_ScheAlgo = 1 or 2
sw_ScheAlgo = 3 or 4
Demultiplex into ABRQ & VBRQ
Demux into VCQ for same priority service
1
2
3
4
Javasim (demux & Schedule output cell)
HQLT (demux & Schedule output cell)
RR (demux & Schedule output cell)
SLQ (demux & Schedule output cell)
Puts the cell on the link
Cell

Figure6.1 Flow of Cells In Switch

6.2 Class Implementation

The class implementation consists of many classes in this simulator, which is grouped into a package called Javasim.

This package consists of two major classes: simulation engine and simulation components. However, only the classes relate to priority scheduling algorithms will be discussed in the following section.

6.2.1 *SPATMSwitch.java*

The *SPATMSwitch* class has its method in managing its own output port and consists of attributes useful for itself. The header file is as follows:

```
class SPATMSwitch extends SimComponent implements Serializable {

  //The private classes below are use to initial priority queues
  //and output ports structure.X-ABR, VBR, CBR

  private class XQ implements Serializable {
    boolean congested=false;
    SimParamInt iq=null;
    SimParamInt sent=null;
    SimParamInt dq=null;
    java.util.List ptr=null;
    java.util.List vcq1=null;
    java.util.List vcq2=null;
    java.util.List vcq3=null;
  }

  private class Port implements Serializable {
    SimComponent to_link=null;
    boolean link_busy=false;
    VBRQ vbr=null;
    ABRQ abr=null;
    CBRQ cbr=null;
    java.util.List sigQueue=null;
    java.util.List spq=null; /* queue for enforcing speedup factor */
    SimParamBool cong=null;
  }

  //this class is use to store details of connection records
  private class CallRecord implements Serializable {
    int vpi1,vci1,callref1;
    int vpi2,vci2,callref2;
    int contype;
    SimComponent comp1=null;
    SimComponent comp2=null;
  }
  //Input and control attributes
  private SimParamInt sw_delay;
  private SimParamInt sw_oqsize;
  private SimParamInt sw_ht;
  private SimParamInt sw_lt;
  private SimParamInt sw_speed;

  //call a route table
  private SimParamRTable sw_route_table=null;
  //call a input window for parameters of any algorithms
  private SimParamPScheduler sw_ps_window=null;
  //initial what user have choose for the scheduling priority
  private SimParamInt sw_ScheAlgo;

  //Display attributes
  private SimParamInt sw_cells_received;
  private int num_dropped,total_cell;
  private SimParamDouble sw_dropped;
  private int sw_speedup;
  private SimParamBool sw_cpucong;
  private SimParamInt sw_log_factor;

  /*for HQLT algorithm*/
  private int hqlt_QTH_U;
  private int hqlt_QTH_L;

  /*for RR algorithm*/
  private SimParamInt vbr_turn;
  private SimParamInt abr_turn;
  private SimParamInt turn;

  /*virtual output ports*/
  private java.util.Map voports;
  private java.util.List records;
  private java.util.Random randgen;

  /*private events*/
  static final int MY_RECEIVE = SimProvider.EV_PRIVATE + 1;
  static final int MY_SLOT_TIME = SimProvider.EV_PRIVATE + 2;
}
```

The different types of attributes declared in the header file had different functions. The input and control attributes is user define values that use to create different conditions for a simulation topology. Display attributes include total number of received cells and percentage of drop cells when simulation is running. Private events include MY_RECEIVE and MY_SLOT_TIME which are needed in the action routine.

Some Private methods are built in the class to implement actions routine for receive, demultiplex and schedule cells in a process slot time. Beside, public methods are built too for start and reset the simulation, and display output ports structure when a link is connected to switch.

6.2.2 SimParamPScheduler.java

The *SimParamPScheduler* class is an information window included functions to return input parameters for HQLT scheduling algorithm when the function is called in *SPATMSwitch class* and some GUI members. This class is inherited from *SimParameter class*. The header of the file is as below:

```
class SimParamPScheduler extends SimParameter implements
ActionListener,java.io.Serializable {

    // the attributes below will be return to SPATMSwitch.java
    int QTH_U;    //upper limit threshold for HQLT
    int QTH_L;    //lower limit threshold for HQLT

    //GUI members (a Swing component)
    private transient JComponent jcomp=null; /* return this
    information window when the related button is press by user */

    JPanel TitlePanel = new JPanel();
    JLabel Title = new Jlabel();

    JPanel InputPanel1=new JPanel();
    JLabel SubTitle1 = new Jlabel();
    JTextField hqu=new JTextField(5);
    JTextField hql=new JTextField(5);

    JPanel compPanel=new JPanel();
    JButton btnSave=new JButton("OK");
    JButton btnClose=new JButton("CANCEL");
}
```

Attributes used includes in the *SimParamPScheduler* class are returnable parameters and GUI members. The GUI members used in this class are JComponent, JPanel, Jlabel, JtextField and JButton. This attributes are used to get input for returnable parameters which is use to implement related priority scheduling algorithms.

The method perform in this class is to get user input values and transfer the value of JTextFiled into returnable parameters.

6.2.3 CBRApp.java

The *CBRApp* class generate CBR cells at a constant rate for the duration of the simulation. The header of the file is as below:

```
class CBRApp extends SimComponent implements java.io.Serializable {
  //user initial values
  private SimParamDouble cn_bit_rate;
  private SimParamInt cn_start_time;
  private SimParamDouble cn_trans_size;
  private SimParamInt cn_repeat;
  private SimParamInt cn_delay;
  private SimParamBool cn_random_size;
  private SimParamBool cn_random_delay;
  private SimParamBool cn_random_target;
  private SimParamInt cn_thisport=null;
  private SimParamNSAP cn_destnsap;
  private SimParamInt cn_destport;
  private SimParamInt cn_conattempt;
  private SimParamInt cn_conaccept;

  private int cn_status;
  private int cn_vpi,cn_vci;
  private long cn_cur_trans_size;
  private int cn_con_done;
  private long cn_num_sent;

  private java.util.Random randgen;

  //connection status constants
  private static final int UNI_NULL = 0;
  private static final int UNI_CALL_INIT = 1;
  private static final int UNI_CALL_PROC = 2;
  private static final int UNI_ACTIVE = 3;

  //private events
  static final int MY_SENDCELL = SimProvider.EV_PRIVATE + 1;
  static final int MY_START = SimProvider.EV_PRIVATE + 2;
}
```

Some attributes used in the *CBRApp* class are user input parameters. This attributes is used to specify the type of CBR traffic will be generate for simulation.

The method perform in this class is to setup connections when the connection status is null and send cells when connection status is active.

6.2.4 VBRApp.java

The *VBRApp* class generate VBR traffics as an ON - OFF source for the duration of the simulation. Cells are generated at the specified bit rate during a burst.

Mean burst length and mean interval between bursts are user specified, but the actual periods of both are drawn from an exponential distribution. The header of the file is as below:

```
class VBRApp extends SimComponent implements java.io.Serializable {
  //user initial values
  private SimParamDouble cn_bit_rate;
  private SimParamDouble cn_burst_length;
  private SimParamDouble cn_int_bet_burst;
  private SimParamInt cn_start_time;
  private SimParamDouble cn_trans_size;
  private SimParamInt cn_repeat;
  private SimParamInt cn_delay;
  private SimParamBool cn_random_size;
  private SimParamBool cn_random_delay;
  private SimParamBool cn_random_target;

  private SimParamInt cn_thisport=null;
  private SimParamNSAP cn_destnsap;
  private SimParamInt cn_destport;
  private SimParamInt cn_conattempt;
  private SimParamInt cn_conaccept;
  private SimParamInt cn_curincome;
  private SimParamInt cn_totalincome;

  private int cn_status;
  private int cn_vpi,cn_vci;
  private long cn_cur_trans_size;
  private int cn_con_done;
  private long cn_num_sent;

  private java.util.Random randgen;

  //connection status constants
  private static final int UNI_NULL = 0;
  private static final int UNI_CALL_INIT = 1;
  private static final int UNI_CALL_PROC = 2;
  private static final int UNI_ACTIVE = 3;

  //private events
  static final int MY_SENDCELL = SimProvider.EV_PRIVATE + 1;
  static final int MY_START = SimProvider.EV_PRIVATE + 2;
  static final int MY_BURST = SimProvider.EV_PRIVATE + 3;
  //Note: MY_BURST is just a delayed send cell
}
```

Some of the attributes used in the *VBRApp* class are user input parameters. This attributes is used to specify what type of real time VBR traffic will be generate for simulation.

Basically, the method perform in this class is similar to *CBRApp* class except that cells can also be generate within an ON (`cn_burst_length)` and OFF (`cn_int_bet_burst)` period.

6.3 Priority Scheduling Algorithms Implementation

There are two main functions in SPATMswitch.java implement the user select priority scheduling algorithms. There are *sw_demux_spq* and *sw_schedule_output*.

Below is the detail description for each algorithm.

6.3.1 Javasim (FCFS and Fixed Priority)

```
 sw_demux_spq (){
 if (sw_ScheAlgo= 1 or sw_ScheAlgo = 2)       {
     if(connection_type=ABR ) {
      if(( total no of cells in ABRQ < sw_oqsize) or  (sw_oqsize=-1)) {
        add cell into ABRQ;
      }
      else {
        drop the cell;
      }
     }
     else if(connection_type=CBR or connection_type=VBR){
      if( (total no of cells in VBRQ  < sw_oqsize) or  (sw_oqsize=-1))  {
        add cell into VBRQ;
      }
      else {
        drop the cell;
      }
     }
 } //end for sw_ScheAlgo = 1 or 2
}

sw_schedule_output() {
  if (sw_ScheAlgo=1 ) {
  //Always remove cell from VBRQ fisrt
  if( VBRQ is not Empty) {
    remove cell from VBRQ;
  }
  //If VBRQ is Empty, remove cell from ABRQ
  else  if( ABRQ is not Empty) {
    remove cell from ABRQ;
  }
}
}
```

6.3.2 Hysteresis effect Queue Length Threshold

```
sw_schedule_output() {
   if (sw_ScheAlgo=2 ) {
QTH_U=upper limit threshold for HQLT algorithm
QTH_L=lower limit threshold for HQLT algorithm

           while (number of cell in ABRQ >= QTH_U) {
                  remove cell from ABRQ;        //Non-Real Time Traffic
           }
           if (number of cell in ABRQ > QTH_L) {
                  remove cell from ABRQ;
           }
           else {
                  // remove cell from VBRQ fisrt
                  if( VBRQ is not Empty) {
                   remove cell from VBRQ;         //Real Time Traffic
                  }
                  //If VBRQ is Empty, remove cell from ABRQ
                  else  if( ABRQ is not Empty) {
                    remove cell from ABRQ;
                  }
           }
}
}
```

6.3.3 Round Robin

```
sw_demux_spq(){
 if (sw_ScheAlgo =3 or 4)
     {//NRT traffic
     if(connection_type=ABR ) {
       if(( (total no of cell for vcq1,vcq2 & vcq3) < sw_oqsize) or (sw_oqsize= -1)) {
        randomly add cell into vcq1, vcq2 or vcq3;
       }
      else {
        drop cell;
      }
     }
    //RTT
    else if(connection_type=VBR || connection_type=CBR ){
      if(( (total no of cell for vcq1,vcq2 & vcq3) < sw_oqsize) or (sw_oqsize= -1)) {
        randomly add cell into vcq1, vcq2 or vcq3;
      }
      else {
        drop cell;
      }
     } //end else for 3
} end for demux

sw_schedule_output() {
  if (sw_ScheAlgo=3 ) {
  get values of SP_VCQ1,SP_VCQ2 & SP_VCQ3;
   if (VBR's vcq1,vcq2 or vcq3 is not Empty) {
          if (vbr_turn.getValue() == 0)  {
           if VCQ1 not empty then remove cell from VCQ1 and set vbr_turn=1;
           else if VCQ2 not empty then remove cell from VCQ2 and set vbr_turn=2;
           else if VCQ3 not empty then remove cell from VCQ3 and set vbr_turn=0;
          }
          else if (vbr_turn.getValue() == 1){
           if VCQ2 not empty then remove cell from VCQ2 set vbr_turn=2;
           else if VCQ3 not empty then remove cell from VCQ3 and set vbr_turn=0;
           else if VCQ1 not empty then remove cell from VCQ1 and set vbr_turn=1;
          }
          else if (vbr_turn.getValue() ==2){
           if VCQ3 not empty then remove cell from VCQ3 and set vbr_turn=0;
           else if VCQ1 not empty then remove cell from VCQ1 and set vbr_turn=1;
           else if VCQ2 not empty then remove cell from VCQ2 and set vbr_turn=2;
          }
}//close vbr
else if(ABR's vcq1,vcq2 or vcq3 is not Empty) {

          if (abr_turn.getValue() ==0)  {
          if VCQ1 not empty then remove cell from VCQ1 set abr_turn=1;
          else if VCQ2 not empty then remove cell from VCQ2 set abr_turn=2;
          else if VCQ3 not empty then remove cell from VCQ3 set abr_turn=0;
          }
          else if (abr_turn.getValue() ==1){
          if VCQ2 not empty then remove cell from VCQ2 set abr_turn=2;
          else if VCQ3 not empty then remove cell from VCQ3 set abr_turn=0;
          else if VCQ1 not empty then remove cell from VCQ1   set abr_turn=1;
          }
          else if (abr_turn.getValue() ==2){
          if VCQ3 not empty then remove cell from VCQ3 set abr_turn=0;
          else if VCQ1 not empty then remove cell from VCQ1 set abr_turn=0;
          else if VCQ2 not empty then remove cell from VCQ2 set abr_turn=0;
          }
}//end abr
}
}
```

6.3.4 Select Largest Queue

```
sw_schedule_output() {
  if (sw_ScheAlgo=4 ) {
```

```
  if (VBR's vcq1,vcq2 or vcq3 is not Empty)  {
   get number of cells for every vcq for VBR/CBR service;
   do bubble sort then find out which vcq is the large queue;

   the vcq1 is the large queue
   if (vcq1 is not Empty)  {
        remove cell from vcq1;
   }
   the vcq2 is the large queue
   if (vcq2 is not Empty)  {
        remove cell from vcq2;
   }
   the vcq3 is the large queue;
   if (vcq3 is not Empty)  {
        remove cell from vcq3;
   }
   }//end for VBRQ

   else if(ABR's vcq1,vcq2 or vcq3 is not Empty) {
   get number of cells for every vcq for ABR service;
   do bubble sort then find out which vcq is the large queue;

   the vcq1 is the large queue;
   if (vcq1 is not Empty)  {
        remove cell from vcq1;
   }

   the vcq2 is the large queue;
   if (vcq2 is not Empty)  {
        remove cell from vcq2;
   }

   the vcq3 is the large queue;
   if (vcq3 is not Empty)  {
        remove cell from vcq3;
   }
   }//end ABRQ
}
}
```

6.4 Chapter Summary

This chapter discusses implementation of priority scheduling algorithms at output ports of the switch for the simulator. Classes with important data items and its associate data are discussed further in the implementation section. Further more, the details implementation for the proposed algorithms also included.

CHAPTER 7: TESTING

Testing is done step by step to compare the simulation result by running the simulator with different priority scheduling algorithms. The priority scheduling algorithms could be test with various conditions, such as by implementing several assumptions of the switching rates for the switch and application.

7.1 CLASS AND UNIT TESTING

The purpose of the class and unit testing is to ensure that parameters, attributes and methods perform in every class is running without error during simulation. Most of the testing is done by observing the output parameters which generated during simulation.

7.1.1 SimParamPScheduler Class

SimParamPScheduler class was tested by input the values for different parameters at the JtextFields in the priority scheduling algorithms' parameters input window. A input window will appear when user click the "Parameters for SA" button at the *SPATMSwitch class*'s parameter list

The input window for different priority scheduling algorithms is shown in *Figure 7.1*. Every time when the input window is called ("Parameters for SA" button is clicked), the previous input parameters will be reset.

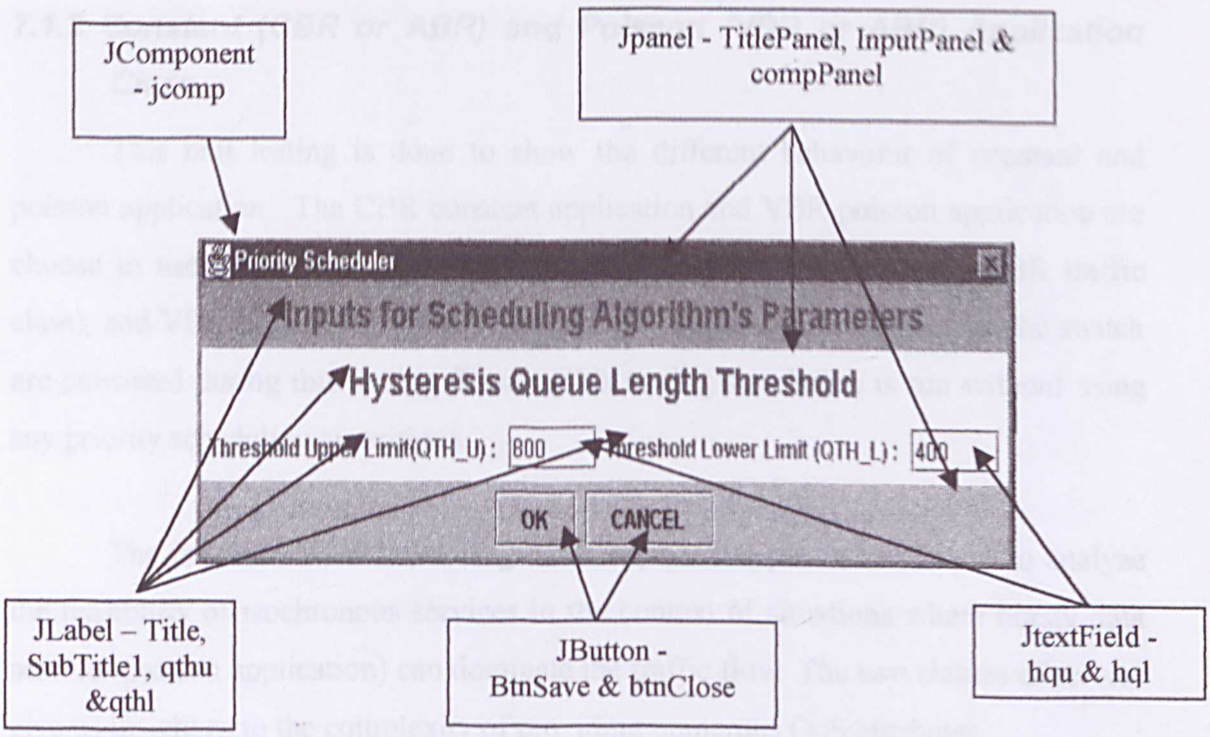

JComponent - jcomp
Jpanel - TitlePanel, InputPanel & compPanel
Priority Scheduler
Inputs for Scheduling Algorithm's Parameters
Hysteresis Queue Length Threshold
Threshold Upper Limit(QTH_U) : 800
Threshold Lower Limit (QTH_L) : 400
OK
CANCEL
JLabel – Title, SubTitle1, qthu &qthl
JButton - BtnSave & btnClose
JtextField - hqu & hql

Figure 7.1 Input Window for Priority Scheduling Algorithms' Parameters

For example, after user key in the value of QTH_U and QTH_L in the two JtextFields, methods in *SimParamPScheduler class* will convert the above values into integer format after the button OK is pressed.

```
Value for QTH_U and QTH_L in SimParamPScheduler class is 800
and 400
```

A new object called sw_ps_window is create in the *SPATMSwitch class* as initial below:

SimParamPScheduler sw_ps_window;.

After the input window in *Figure 7.1* is closed, the correct values of QTH_U and QTH_L from *SimParamPScheduler class* will be called and return in *SPATMSwitch class* by the follow statements:

int hqlt_QTH_U=sw_ps_window.getQTHU();

int hqlt_QTH_L=sw_ps_window.getQTHL();

```
HQLT algorithm initial by 2 Value for QTH_U and QTH_L in
SPATMSwitch class is 800 and 400
```

7.1.2 Constant (CBR or ABR) and Poisson (VBR or ABR) Application Class

This unit testing is done to show the different behaviour of constant and poisson application. The CBR constant application and VBR poisson application are choose to use in this testing. Therefore only CBRQ (reserved for the CBR traffic class), and VBRQ (reserved for the VBR traffic class) in an output port of the switch are consisted during this testing. Beside, this testing simulation is run without using any priority scheduling algorithms.

The intention behind choosing these important types of services is to analyze the feasibility of isochronous services in the context of situations where bursty data service (poisson application) can dominate the traffic flow. The two classes of service give us insight as to the complexity of providing numerous QoS attributes.

Below are the assumption make for this testing:

- The output queue length for the queues of switch was limited to 1000 entries
- The traffic sources used in this testing consisted of CBR sources and VBR sources with a single output link at 155 Mb/s (*refer Figure 7.2*).
- The CBR sources were transmitting data at a constant rate of 155 Mb/s.
- The VBR sources were transmitting "bursty" traffic at a rate of 155 Mb/s for 1 milliseconds (1000 usecs) followed by an OFF period where no data was transmitted for 1 milliseconds (1000 usecs) (*refer Figure 7.3*).
- Incoming traffic may be at rates very different from the two incoming links considered in this testing.

The congestion happened and cells will drop when all the queues are full and the aggregate incoming rate to the switch exceeds the output flow rate to the output link.

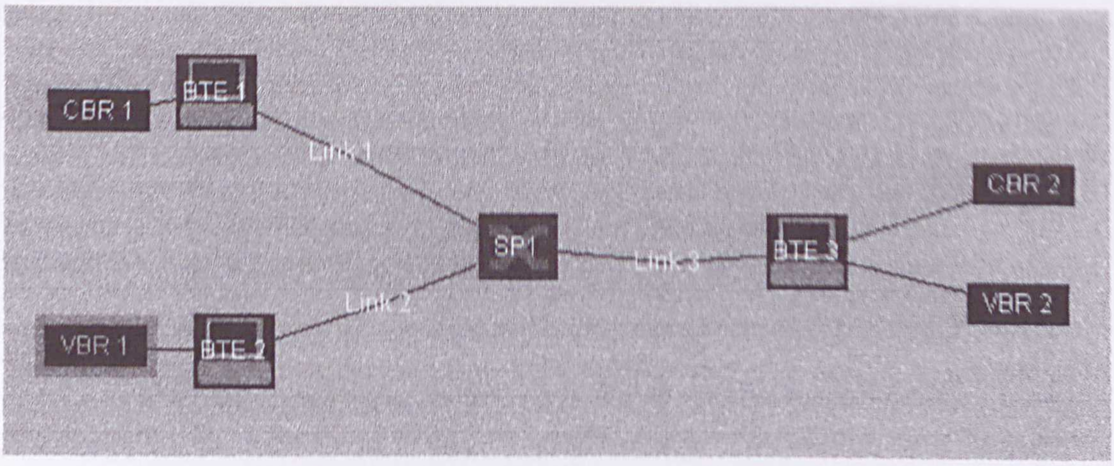
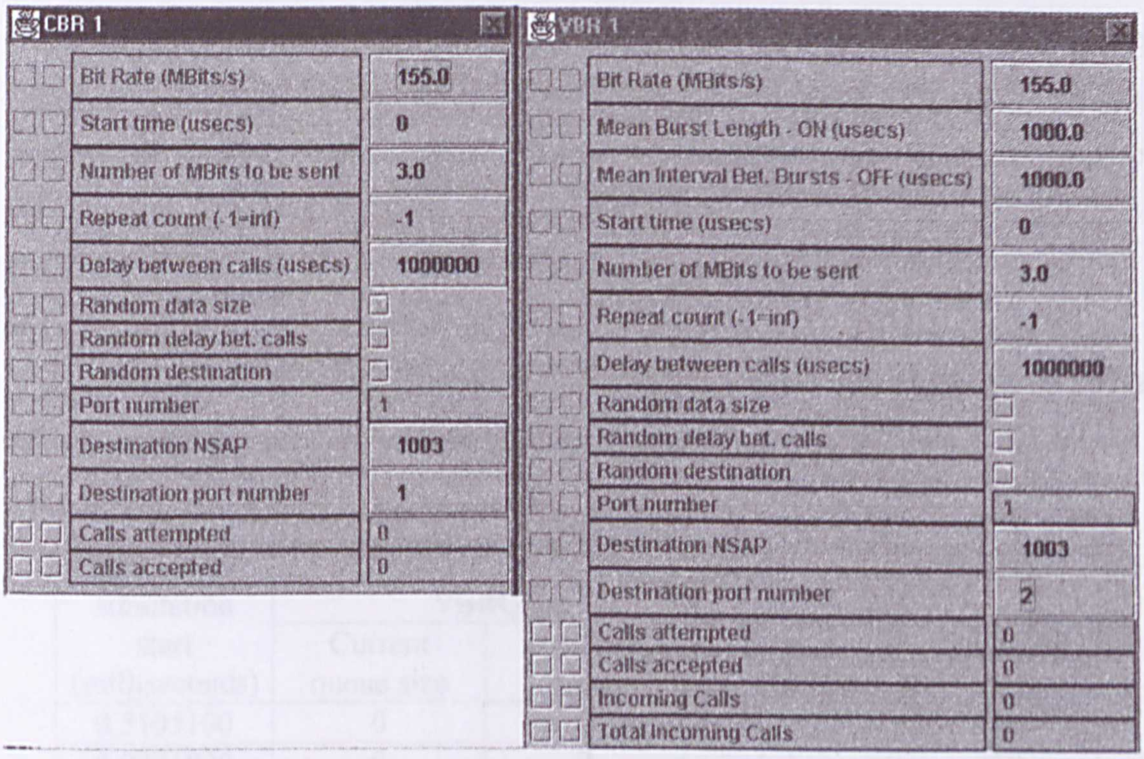


Figure 7.2 Network Topology for Testing of Constant and Poisson Applications



CBR 1

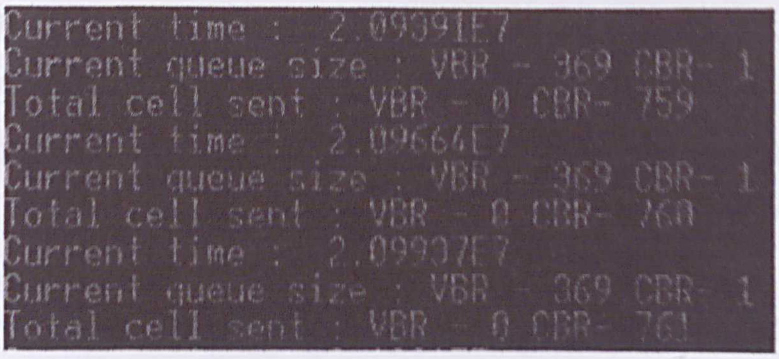
Bit Rate (MBits/s)	155.0
Start time (usecs)	0
Number of MBits to be sent	3.0
Repeat count (-1=inf)	-1
Delay between calls (usecs)	1000000
Random data size	☐
Random delay bet. calls	☐
Random destination	☐
Port number	1
Destination NSAP	1003
Destination port number	1
Calls attempted	0
Calls accepted	0

VBR 1

Bit Rate (MBits/s)	155.0
Mean Burst Length - ON (usecs)	1000.0
Mean Interval Bet. Bursts - OFF (usecs)	1000.0
Start time (usecs)	0
Number of MBits to be sent	3.0
Repeat count (-1=inf)	-1
Delay between calls (usecs)	1000000
Random data size	☐
Random delay bet. calls	☐
Random destination	☐
Port number	1
Destination NSAP	1003
Destination port number	2
Calls attempted	0
Calls accepted	0
Incoming Calls	0
Total Incoming Calls	0

Figure 7.3 Parameters Input Window for Constant and Poisson Applications

Every time when a cell is scheduled to output link during the simulation, a screen with information about current queue size and total number of cell sent will be print as *Figure 7.4* below. Some of the data are copy into *Table 7.1*.



```
Current time :  2.09391E7
Current queue size : VBR - 369 CBR- 1
Total cell sent : VBR - 0 CBR- 759
Current time :  2.09664E7
Current queue size : VBR - 369 CBR- 1
Total cell sent : VBR - 0 CBR- 760
Current time :  2.09937E7
Current queue size : VBR - 369 CBR- 1
Total cell sent : VBR - 0 CBR- 761
```

Figure 7.4 Sample Output Data Generate During Simulation

The *Table 7.1* show that the *VBRApp class* is generated the bursty cells in ON-OFF state as initial Mean Burst Length (1000 usecs or 1 milliseconds) and Mean Interval Between Burst (1000 usecs or 1 milliseconds). That is, the VBRQ at the output port in switch only start to receive cell at 1 milliseconds after the simulation start. At the time 2.09 milliseconds after the simulation start, the *VBRApp class* stop sending cell, therefore, the number of cells in VBRQ never increase during 2 to 3 milliseconds.

Beside that, the table also show that *CBRApp class* is generate cells as initial bit rate and cells is sent to switch constant and continuously.

Time after simulation start (milliseconds)	VBRQ		CBRQ	
	Current queue size	Total cell sent	Current queue size	Total cell sent
0.5105100	0	0	1	179
0.9991800	0	0	1	358
1.0483200	11	0	1	376
1.0920000	27	0	1	392
1.5015000	177	0	1	542
1.9983600	359	0	1	724
2.0939100	369	0	1	759
2.5443600	369	0	1	924

Table 7.1 Simulation Results for Testing of CBR and VBR Applications

7.1.3 HQLT With Different Values of Upper Limit And Lower Limit Threshold

The HQLT algorithms can be tested with different value of Upper Limit and Lower Limit Threshold by running a simulation which use a network topology shown in *Figure 7.5*. Below are the two conditions set for this testing:

- The output queues' size in this simulation is set to 1000 for every switch.
- Every simulation will be pause after 0.010 seconds from the beginning of simulation.

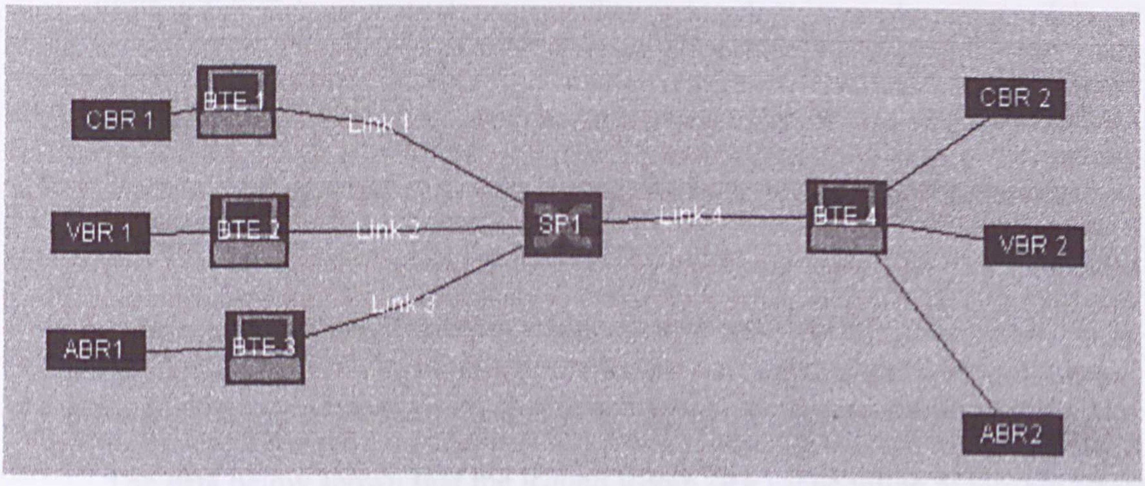


Figure7.5 Network Topology for System Testing of HQLT

Every time the simulation is running with different pair of Upper Limit and Lower Limit Threshold as listed in *Table 7.2*. The simulation results show that when the values of Upper Limit (QTH_U) is increase and Lower Limit Threshold (QTH_L) is decrease, the total percentage of cells drop will slightly increase even the ABR cells sent to destination is increase. It is because the dropped cells are contributed by CBR and VBR cells which have total number of cells is greater then the total number of ABR cells.

Therefore, to decrease the total percentage of cells drop, the values of Lower Limit Threshold (QTH_L) should be increase when Upper Limit (QTH_U) is increase as data shown in the fifth column in the below table.

QTH_U	100	120	140	800
QTH_L	60	40	20	400
Total Cells Receive	4640	4631	4622	4766
Total Cells Drop %	55.76	56.16	56.56	49.02
Cell in VBRQ (RTT) to link 4	1000	1000	1000	1000
Cell sent to destination from VBRQ	64	44	24	404
Cell drop from VBRQ to link 4	2586	2600	2613	2336
Cell in ABR (NRT) to link 4 to link 4	61	40	21	401
Cell sent to destination from ABRQ	922	940	957	618
Cell drop from ABRQ to link 4	0	0	0	0
Total Cell Receive at same BTE	983	982	980	1021

Table 7.2 Values of Output Parameters Generated During Simulation

7.1.4 Performance of Round Robin (RR) Scheduling Algorithm

The data below shown that the Round Robin (RR) scheduling algorithm treats all queues equally. The Round Robin (RR) scheduling algorithm is implement by transmit cell from different queues to output link in a sequence of one queue by one queue for the same priority class.

The turn number in the data below is to represent which queue is served currently. There is, after transmitting a cells from the first queue (turn = 0), the scheduler will start checking and transmitting a cells from the second queue (turn = 0) at the next time it return to the same priority class of service (*VBRApp class*).

```
Current time :  1.96742E7
Current turn :  0
Current queue size (VBR): VCQ1 - 166 VCQ2 - 197 VCQ3 - 128
After scheduling (VBR): VCQ1 - 165 VCQ2 - 197 VCQ3 - 128

Current time :  1.97802E7
Current turn :  1
Current queue size (VBR): VCQ1 - 168 VCQ2 - 197 VCQ3 - 131
After scheduling (VBR): VCQ1 - 168 VCQ2 - 196 VCQ3 - 131

Current time :  1.98862E7
Current turn : 2
Current queue size (VBR): VCQ1 - 168 VCQ2 - 196 VCQ3 - 136
After scheduling (VBR): VCQ1 - 168 VCQ2 - 196 VCQ3 - 135

Current time :  1.99922E7
Current turn :  0
Current queue size (VBR): VCQ1 - 168 VCQ2 - 196 VCQ3 - 140
After scheduling (VBR): VCQ1 - 167 VCQ2 - 196 VCQ3 - 140
```

7.1.5 Performance of Select Largest Queue Scheduling (SLQ) Algorithm

The data below show the implementation of Select Largest Queue (SLQ) scheduling algorithm. The Select Largest Queue (SLQ) scheduling algorithm is implement by transmit cell from a queue with largest size (highest number of total cells in queue) to output link for the same priority class.

At same priority class of service (*VBRApp class*) there exist three non-empty queues (VCQ1, VCQ2 and VCQ3), and the largest queue in the priority class will be allowed to send cell to its destination. However, before sending the cell, the simulator will compare the current total number of cells in every queue with sorting method to find out which queue is the largest queue.

```
Current time :  3.21822E7
Before sorting and scheduling (VBR):
VCQ1(a[0]) - 291 VCQ2(a[1]) - 161 VCQ3 (a[2])- 276
After sorting (VBR):
a[0] - 161 a[1] - 276 a[2] - 291
After scheduling (VBR):
VCQ1(a[0]) - 290 VCQ2(a[1]) - 161 VCQ3 (a[2])- 276

Current time :  3.22882E7
Before sorting and scheduling (VBR):
VCQ1(a[0]) -296 VCQ2(a[1]) - 161 VCQ3 (a[2])- 276
After sorting (VBR):
a[0] - 161 a[1] - 276 a[2] - 296
After scheduling (VBR):
VCQ1(a[0]) - 295 VCQ2(a[1]) - 161 VCQ3 (a[2])- 276
```

7.2 SYSTEM TESTING

The system testing is implemented by compare the simulation result for two different algorithms with same topology. This step is taken to get more details for the difference between compared algorithms.

Two different types of systems testing is done, there are:

- Compare two simulations with and without Hysteresis effect Queue Length Threshold (HQLT) scheduling algorithm for service category within different priority class when congestion is happened.
- Compare Round Robin (RR) and Select Largest Queue (SLQ) scheduling algorithm for service category with same priority class when congestion is happened.

The major network and traffic scenario for System Testing is set as below:

- Three applications (CBR application, VBR application and ABR application) are sending data to the same target destination. Therefore, the queues at an output port of the switch are highly demanded and needs to transfer out the cells in a high rate. But unfortunately it is difficult to send out the cells to the link, as the link rate is slower. So, congestion will happen between at the output port and outgoing link and some cells will be drop.
- Initial the value of the speed of link connected at the output queue 25% smaller than switch's switching rate. This means that input link speed (155 km/s) is smaller than output link speed (40 km/s) and the distance of link in every between BTE and switch is 0.1 km.
- All applications (ABR and CBR) will be given the same bit rate (100 Mbits/s), number of bits to send (30 Mbits) and cell delay (1000,000 usecs), except VBR application will be given an ON-OFF state by initial the values of Mean Burst Length (1000 usecs) and Mean Interval bet. Burst (1000 usecs).
- The output queues' size in this simulation is set to 1000 for every switch.
- Every simulation will be pause after 0.010 seconds from the beginning of simulation.

7.2.1 Javasim(FCFS & Fixed Priority) vs Hysteresis effect Queue Length Threshold (HQLT)

Even there are three different applications (CBR application, VBR application and ABR application) sent different types of traffic to a same target destination, but these applications can be category into Real Time Traffics (CBRApp and VBRApp) and Non Real Time Traffic (ABRApp) (refer *Figure 7.6*).

Two simulations with different *Sche_Algo* – 1 and 2 (parameter for running priority scheduling algorithm) are tested. There first test is run without any scheduling algorithm, there mean that the Real Time Traffics (VBRQ) always be serve first, and only if Real Time Traffics is finish to serve (VBRQ is empty), then switch will schedule cell of Non Real Time Traffic (from ABRQ) into output link. During the second test (simulation for HQLT), when the number of cells in ABRQ exceeds QTH_U (upper limit threshold) and QTH_L (lower limit threshold), cell in ABR queue will be removed first.

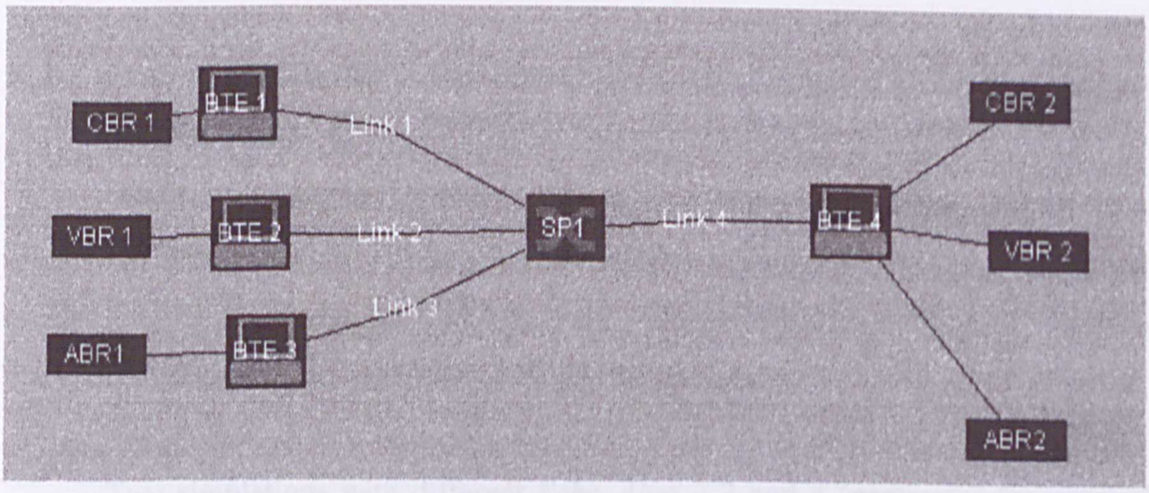


Figure7.6 Network Topology for System Testing of Javasim vs HQLT

Performance Evaluation

The results of simulation for this testing are shown at the *Table7.3.*

Even the number of cell in ABR queue always be less then value of QTH_U threshold, but the cell dropped percentage for HQLT shown in the both table is more then Javasim. Therefore, HQLT may not be a better method to control the non real time traffic service category when congestion happens at output port of a switch.

Beside that, the total number of cells for Real Time Traffic queue (VBRQ) is more than Non Real Time Traffic queue (ABRQ), it is because both CBR and VBR application is sharing the same Real Time Traffic queue during this simulation.

Priority Scheduling Algorithm	Javasim	HQLT
sw_ScheAlgo:	1	2
QTH_U	-	800
QTH_L	-	400
Cells Receive	4757	4766
Cells Drop %	36.40	49.02
Cell in VBRQ (RTT) to link 4	999	1000
Cell sent to destination from VBRQ	1020	404
Cell drop from VBRQ to link 4	1715	2336
Cell in ABR (NRT) to link 4 to link 4	1000	401
Cell sent to destination from ABRQ	0	618
Cell drop from ABRQ to link 4	16	0
Total Cell Receive at same BTE	1018	1021

Table7.3 Data Generated for System Testing of Javasim vs HQLT

7.2.2 Round Robin (RR) vs Select Largest Queue (SLQ) For Same Priority Class

The traffics sources for this testing are the cells generate from the *VBRApp class* and *CBRApp class*, it is because VBR poisson service and CBR constant service are assume that in a same priority class (Real Time Traffic) (refer *Figure 7.6*). There are three Virtual Channel Queue (VCQ) used in this testing.

Two simulations with different *Sche_Algo* – 3 and 4 (parameter for running priority scheduling algorithm) are tested. The first test is run with Round Robin (RR) scheduling algorithm, there mean that after schedule a cell from Q_i, the switch will moves to serve queue Q_{i+1} at next slot time. During the second test (simulation for SLQ), a sorting method is use to rearrange the VCQs in increasing order of total number of cells in queue, cell in the queue with largest size will be removed first.

The results of simulation testing for the two algorithms are shown at the *Table7.3.*

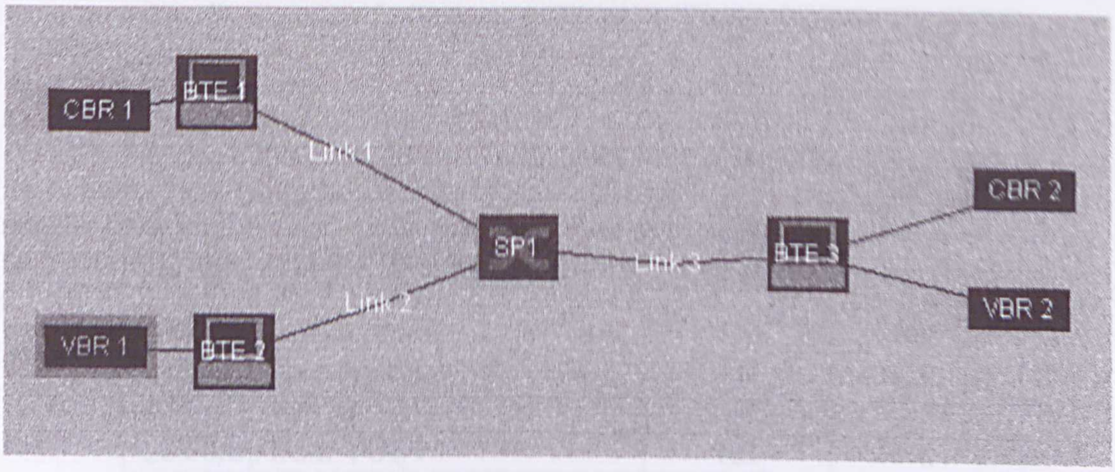


Figure7.7 Network Topology for System Testing of RR vs SLQ

Performance Evaluation

The result in *Table 7.4* show that Select Largest Queue scheduling algorithm has better performance then Round Robin scheduling algorithm. It is because the Select Largest Queue scheduling algorithm can maintain the fairness for every VCQ in same priority class more effectively compare with Round Robin scheduling algorithm.

Consider that the number of cells drop in VCQ3 for Round Robin scheduling algorithm is 122, but number of cells drop in VCQ3 for Select Largest Queue scheduling algorithm is 0. This mean that the VCQ3 for Round Robin scheduling algorithm access its optimum buffer space more quickly then VCQ3 for Select Largest Queue scheduling algorithm.

Beside that, the total number of cells in every VCQ for Round Robin scheduling algorithm has more significant different compare with total number of cells in every VCQ for Select Largest Queue scheduling algorithm with less significant different. Therefore, it can be concluded that Select Largest Queue scheduling algorithm threats every VCQ with more fairness then Round Robin scheduling algorithm.

Priority Scheduling Algorithm	RR	SLQ
sw_ScheAlgo:	3	4
Cells Receive	3638	3628
Cells Drop %	3.35	0
Total Cell Sent	979	976
Cell in VCQ1 to link 4	718	892
Cell drop from VCQ1	0	0
Cell in VCQ2 to link 4	814	892
Cell drop from VCQ2	0	0
Cell in VCQ3 to link 4	1000	865
Cell drop from VCQ3	122	0
Total Cell Receive at same BTE	978	974

Table7.4 Data Generated for System Testing of RR vs SLQ

7.3 CHAPTER SUMMARY

This chapter discusses class, unit and system testing that had been done for the simulator. Beside that, comparison between different priority scheduling algorithms also described base on the simulation results.

CHAPTER 8: CONCLUSION

From this project, the most valuable experience is to study and understand into more detail for the existing ATM traffic shaping techniques and different types of priority scheduling policy. It is valuable to have experience at the same time building object-oriented model for the ATM network simulator, which mainly emphasizing on implementation of different priority scheduling policies.

From the two testing result at chapter 7, can be concluded that Hysteresis effect Queue Length Threshold (HQLT) scheduling algorithm not able to reduce the percentage of cell dropped when given priority to Non Real Time Traffic. Beside, also can be concluded that Select Largest Queue scheduling algorithm have a better performance then Round Robin scheduling algorithm for served the queues in same priority class.

An object-oriented ATM network simulator environment emphasizing on priority scheduling policies is designed and implemented to fulfill the purpose of improving the efficiency in performing computer network simulation studies. This simulation project is designed to be portable, which allows component developed by different modeler to be shared and thus achieve reusability and flexibility. It is possible for different modeler to add or remove any methods currently designed since the simulator is built by an object-oriented approach using Java programming language. Moreover, applying multithreading into a simulator design can be used to closely model the real ATM network simulator.

With the benefit of using Java programming language, one of the major advantages is the probability of Java applets into the World Wide Web and Internet server. From the web browser, the simulator could be executed on any computer platform.

REFERENCES

[1] Jain, Raj. And Siu, Kai-Yeung. A Brief Overview of ATM: Protocol Layers, LAN Emulation, and Traffic Management.

[2] J. Kenny, 1999. The ATM Forum Traffic Management Specification Version 4.1, *AF-TM0121.000.*

[3] Diaz, J.C.; Plaza, P.; Crespo, J.ATM traffic shaper: ATS. *Design, Automation and Test in Europe, 1998., Proceedings* ,1998 , Page(s): 96 –101

[4] Huang, Alan., Moreland, Chuck. And Wright, Ian. 1998. Advanced Traffic Management for Multiservice ATM Networks. Network Equipment Technologies, Inc

[5] Lizambri, T.; Duran, F.; Wakid, S. Priority scheduling and buffer management for ATM traffic shaping. *Distributed Computing Systems, 1999. Proceedings. 7th IEEE Workshop on Future Trends of* , 1999 , Page(s): 36 –43

[6] Chaudhry, S.; Choudhary, A. Time dependent priority scheduling for guaranteed QoS systems. *Computer Communications and Networks, 1997. Proceedings., Sixth International Conference on* , 1997 , Page(s): 236 –241

[7] Meierhofer, J.; Bernhard, U.P.; Balmelli, P.; Bernasconi, D. Priority scheduling algorithm for ATM wireless network access. *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on* , 1997, Page(s): 289 -294 vol.1

[8] What is Simulation? http://www.cis.ufl.edu/~fishwick/introsim/node1.html Last Updated: 19 October,1995. Paul Fishwick.

[9] The REAL Network Simulator. *http://minnie.cs.adfa.edu.au/REAL/index.html* Last updated: June, 1995. Warren Toomey.

[10] Computer Network Simulation. http://oak.ece.ul.ie/~dalyf/thesis/aa.htm Last updated: October, 1997. Fergal Daly.

[11] Nada G., el. 1998. The NIST ATM/HFC Network Simulator Operation and Programming Guide Version 4.0. National Institute of Standards and Technology.

[12] REAL 5.0 Overview.

http://www.cs.cornell.edu/skeshav/real/overview.html
Last updated: August 13, 1997. S. Keshav, Cornell University.

[13] Goyal, Rohit., Jain, Raj., Fahmy, Sonia., and Narayanaswamy, Shobana. Modeling Traffic Management In ATM Network With OPNET.

[14] Malgosa-Sanahuja, M; Castells-Cuscullola, J; Garcia-Haro.J. 1997. 10 Years PACRIM 1987-97 Networking the Pacific Rim 1997. *IEEE Pacific Rim Conference on Communications, Computer and Signal Processing.* Pp. 972-976.

[15] Raj Jain. 1996. Congestion Control and Traffic Management in ATM Networks: Recent Advances and A Survey. *Computer Network and ISDN Systems*. Vol. 28. 1996. pp. 1723-1738.

[16] C.Lehr, Robert. And W.Mark, Jon. On Traffic Shaping in ATM Networks

[17] *Jong, S.G.; Chin, Y.O.* Algorithms on dynamic priority scheduling for heterogeneous traffic in ATM. *Local Computer Networks*, 1993., Proceedings., 18th Conference on , 1993 , Page(s): 380 –387

[18] *Raha, A.; Malcolm, N.; Wei Zhao.* Performance evaluation of admission policies in ATM based embedded real-time systems. *Local Computer Networks*, 1994. Proceedings., 19th Conference on , 1994 , Page(s): 129 -138 CNF

[19] Seok-Kyu Kweon; Shin, K.G., Traffic-controlled rate monotonic priority scheduling of ATM cells. *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation., Proceedings IEEE* Volume: 2 , 1996 , Page(s): 655 -662 vol.2

[20] Guo, M., Hoang, D.B. 1998 An Object-based Network Simulator. *Global Telecommunication Conference 1998, GLOBECOM 1998.* Vol.3, pp. 1562-1567. November 1998.

[21] Deitel & Deitel. 1999. *JAVA How to Program, Third edition.* Prentice Hall Inc. New Jersey.

[22] Deitel & Deitel. 1999. *C++ How to Program, Third edition.* Prentice Hall Inc. New Jersey.

[23] M. Melly., Object Oriented Basic Concepts and Advantages. *http://www.mmrg.ecs.soton.ac.uk/publications/archive/melly1995a/html/node3.html.*

[24] Java. *http://www.whatis.com.* TechTarget.com.

[25] The Java^{TM} Language: An Overview. *http://java.sun.com/docs/overviews/java/java-overview-1.html*. Last updated: May 12, 2000. Sun Microsystems, Inc.

[26] Lim, Shiau Hong., Java Network Simulator - Specification & Development Guide (version 0.31, 0.4)

[27] Borland JBuilder. *http://www.borland.com/techpubs/jbuilder/jbuilder3-5/qs/intro.html*. Last updated: April 7, 2000. Borland Inprise.

[MONET] Jane Li., Procedural Programming Language. http://monet.uwaterloo.ca/janeli/pp.htm.