CHAPTER 2: LITERATURE REVIEW

This chapter begins with an introduction to various available simulators. A detail description is provided for the simulators that have been studied, includes the advantages and limitation. A comparison is done between the simulators. Study on simulation model is also performed. A review is conducted on both the simulators and simulation models at the end of section.

The following section explains the differences between procedural approach and object-oriented approach. It also explains why object-oriented approach is preferred in developing the ATM network simulation model. It states the reasons of implementing the simulation model using an object-oriented programming language, Java later in this section.

The final section of this chapter will discuss the ATM traffic management techniques. This includes congestion control mechanisms, queuing models, several simulation models and other factors to be considered in designing an ATM network simulator.

2.1 Computer Network Simulation

Computer simulation is a discipline of designing a model of an actual or theoretical physical system. A simulator could execute the model on a digital computer, and at the same time analyse the execution output. Simulation embodies the principle of "learning by doing" - to learn about the system with building a model and operates it.

A simulation could pictures a reality and complexity of a real network environment. Artificial objects must be built and dynamically act out roles in the simulation environment. Within the tasks of a simulation, there are three primary sub-fields, which must be performed: model design, model execution and model analysis (Figure 2.1) [7].

There are several critical characteristics that must be related to the simulation, if the simulation is to simulate the real environment's meaning. These characteristics include that the model design must be accurate, the execution must be relevant to the study and the analysis be made with understanding of the particulars and nature of the model.
The model design of the simulation plays an important role in the entire simulation. The model design must be related to the problem situation. The examples for model designs are conceptual models, functional models, constraint models, special models as well as multimodels.

The execution model handles the semantics of the simulation based on the problem domain for the simulation. Serial algorithms as well as parallel algorithms are needed to execute the simulation model. The simulator could then behave appropriately to the nature of real environment meaning.

Execution analysis could be performed to ensure the execution has been able to lead to correct results. Therefore, examples of analysis are input-output analysis, experimental design, visualization data, verification and validation on the result produced.

*Figure 2.1: Three Sub-Fields of Computer Simulation.*
2.1.1 Advantages and Disadvantages of Simulation

The advantage of a simulator is that it is portable to employ for large variety of system when a uniform execution technique is used. Besides, simulation model can be extensive towards a larger scale of tests if sufficient resources and hardware supports are available.

However, there could be problems encounter while using simulator as well. One would need to redo codes to fit in the simulation environment and problems domain. The simulation implementation and result could be differing considerably from a real environment.

Although a simulation technique could have pros and cons, simulation techniques are still widely used in modelling systems such as mathematical functions, network environments and others.

2.1.2 Types of Simulation

Essentially there are two forms of network simulation: Analytical modelling simulation and discrete event simulation. Analytical modelling is a mathematical technique that characterises a network as a set of equations. While 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 [8].

2.1.2.1 Analytical Modelling Simulation

Analytical modelling simulation is a closed-form approach to model simulation in which the network is modelled mathematically in the form of functions and equations.

The main disadvantage associated with analytical models is that it has over simplistic view of the network. Besides, it is unable to simulate the dynamic nature of a computer network.

2.1.2.2 Discrete Event Simulation

In discrete event modelling, the computer model replicates the real-world objects. Each object plays certain roles and changes its state only at discrete points during the simulation time. This approach models the real-world system by equating the times in the simulation with the time that would be taken for specific events to occur in the real world. Such events might be
the switching of packets, the traversing of data through links or the delay caused by congestion in a network environment. Although this approach is far more accurate, it requires greater computation time and may need more time in modelling the system accurately. However, since the models primary concern is accuracy, this approach is more desirable.

Discrete event simulation has many advantages and it requires more processing time. Thus, quite a considerable investment of time is needed to accurately simulate most models [8].

Therefore, discrete event simulation model is chosen as a model for the ATM network simulator. The reason is that timing is important for the ATM network simulator to perform its functions and produce a result at a period of time. Besides, an ATM network simulator could have several components and objects that perform different functions at the same time. Furthermore, accuracy is an important reason for this choice.

Survey on various simulators is performed. It is performed to observe the simulation techniques used, as well as their techniques used in describing their functionality and procedures towards building a network simulator.

2.1.3 Study of Various Simulators

Due to the emerging of speeds and dynamic of the computer networks, there are a few existing simulator available. The simulators are used to describe a number of simulation experiments performed 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 ATM network.

The general requirements of an ATM simulator are to support network performance analysis under varying traffic types, traffic loads, network capacity planning, traffic aggregation studies and ATM network protocol research. This spans a wide range of applications from production use by ATM network planners to ATM switch network and protocol design by researchers.
A study is done on simulators that includes REAL Network Simulator, UCB/LBNL/VINT Network Simulator – ns, OPNET Modeller Simulator, INSANE Simulator, DELSI 1.1 Simulator, NIST ATM/HFC Network Simulator and Libra Object-oriented ATM Network Simulator.

### 2.1.3.1 REAL Network Simulator

The REAL network simulator is a network simulator designed for testing on congestion and flow control mechanisms. The simulator takes input as a scenario, which is the description of network topology, protocols, workload and control parameters. It produces 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 two different versions: REAL 4.0 and REAL 5.0.

REAL version 5.0 provides users with a way of specifying such networks and to simulate their behaviour. It provides around 30 modules (written in C) that exactly emulate the action of several well-known flow control protocols (such as TCP). Besides, it has five research scheduling discipline (such as Fair Queuing and Hierarchical Round Robin). This version includes a graphical user interface (GUI) written in Java. This allows users to quickly build simulation scenarios with a point-and-click interface [9].

The 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 [10].

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 mode. 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 with REAL 4.0, many changes have been upgraded from version 4.0 to version 5.0. These include:

- new simulation modules
- Java-based GUI
- faster, smaller, cleaner simulation engine
- portability to FreeBSD, Solaris, and Digital Unix (OSF/3)
- minor bugs fixed in version 4.0

**Advantages**

This simulator provides a flexible test-bed 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 additional network components.

**Limitations**

Some of the REAL version's network simulator does not provide user an interactive modelling environment with a graphical user interface (GUI). But GUI is available in REAL version 5.0. This version includes a 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 experience using different platforms is a must for programmers to make changes from the source code provided because this simulator could only run in several types of platforms.
2.1.3.2 UCB/LBNL/VINT Network Simulator - ns

The LBNL Network Simulator, ns, is a simulation tool developed by the Network Research Group at the Lawrence Berkeley National Laboratory. Ns is an extensible, easily configured and programmed event-driven simulation engine, with support for several flavours of TCP (include SACK, Tahoe and Reno) and router scheduling algorithms. This work derives from S. Keshav's REAL simulator. NS is a discrete event simulator targeted at networking research and it provides substantial support for simulation of TCP, routing, and multicasting protocols [11]. This simulator has two different versions as well, namely ns-l and ns-2.

Advantages

This simulator can be used for packet-based simulations. Both the simulators: ns-l and ns-2 has the following functions:

- one-way TCP (Tahoe, Reno, Vegas, SACK) functions, which do not include SYN/FIN packets. (The Vegas implementation has only limited validation.)
- two-way Reno TCP (with SYN/FIN packets and two-way data flow) functions
- Random Early Detection (RED) queue management and class-based scheduling (CBQ)
- dynamic routing and dense-mode multicast routing
- flow manager
- lossy links
- two-way TCP
- telnet sources

ns-2 has additional functions which are listed as below [12]:

- Multi-path routing.
- Real Time Protocol (RTP)
- Scheduling algorithms such as Stochastic Fair Queuing (SFQ), Fair Queuing (FQ), and Deficit Round Robin Scheduling (DRR)
• Scalable Reliable Multicast (SRM)

• "Centralised multicast" for speeding up large-scale simulations.

• Support for mobile hosts. (Link-layer, MAC, and shared channel modules have been implemented.)

• Some random bug fixes, example in TCP transmit timer behaviour

Besides all the functionality advantages for both ns-1 and ns-2, another advantage of this simulator is that, it is modelled in an object-oriented approach. It was developed with the use of C++ programming language. The object-oriented model approach could have several advantages such as extensibility, portability, modularity and reusability.

**Limitations**

As in ns-1, there is no dynamic window advertisement segment and ACK number computations are in units of packets, and there is no SYN/FIN connection establishment/teardown in the simulator model for one-way TCP.

Besides, there is no dynamic window advertisement, no 2MSL-wait or persist states, no urgent data, and no RESET segments in the simulator model for two-way TCP. But recently, SACK, Newreno, and Tahoe functionality have been added to this model. The one-way implementation in the simulator does not has the sender check that the receiver is ECN-capable, as would be done in a real implementation, but the two-way TCP does.

Ns-1 does not run correctly with older versions of gcc and with some other C++ compilers. And it is not compatible with Tcl version older than 7.5. It should compile and run on any UNIX system that Tcl runs on. Therefore, consideration on this version should be improved.

Another disadvantage of this simulator is that, it is built to simulate TCP packets instead of ATM cells, which are using in fixed size cells.
2.1.3.3 OPNET Modeller Simulator

OPNET, or OPtimised Network Engineering Tool is a modelling and simulation tool, which was developed by MIL3 Inc. [13]. Its ATM model suite (AMS) supports many of the characteristics of ATM network [14]. It provides support for signalling, call setup and teardown, segmentation and reassembly of cells, cell transfer, traffic management and buffer management.

OPNET is a discrete event simulation package [15]. Currently, OPNET has been a commercial simulation program, which provide solutions for modelling and simulation of communication network, devices, and protocols. It used object-oriented modelling approach and graphical editors showing the structure of actual network and network components. The behaviours of each component is specified by a state transition diagram and the processes that take place in each state are described in C programming language.

Advantages

OPNET is object-oriented, where each object has a defined set of attributes. These configurable attributes result in a highly flexible development environment. Besides, OPNET employs a hierarchical approach to modelling, having three separate domains to describe any communication network. Each level of the hierarchy describes different aspects of the complete model being simulated. Models at a layer could be re-used (inherit) by other models of higher layers. This leads to a highly flexible simulation environment where generic models can be developed and used in many different scenarios.

Detailed library models provide support for existing protocols and allow researchers and developers to either modify these existing models or develop new models of their own. The set of standard models provides an insight into the workings of existing communication processes, protocols, and networks.

OPNET models can be compiled into executable code. An executable discrete-event simulation can be debugged or simply executed, resulting in output data.
The interactive debugging tool allows model developers to trace a wide range of model characteristics. For example, the contents of a packet or the creation time of a process can be easily examined.

OPNET is extremely useful for:

- Understanding the dynamic nature of networks, protocol data units and protocol interactions.
- Developing new communication protocols or optimising existing protocols.
- Analysing the performance characteristics of existing systems.
- Exploring new technologies and analysing the expected impact of introducing these new networks on existing configurations.

In short, OPNET could manage complex network topologies with unlimited sub network nesting in its hierarchical network models. It simulates arbitrary behaviour with C/C++ logic in states and transitions. It has 400 library functions to support and simplify writing protocol models.

Besides, it has geographical as well as dynamic mobility modelling that is used for mobile and satellite networks. With its total openness of API for program driven construction or inspection, it could easily integrate existing code libraries into simulation.

**Limitations**

This simulator only supported by limited platforms, such as Solaris, Windows NT, and HP-UX. Besides, the cost of using OPNET is extremely high.

### 2.1.3.4 INSANE Simulator

INSANE is a network simulator designed to test various IP-over-ATM algorithms with realistic traffic loads derived from empirical traffic measurements [16]. INSANE’s ATM protocol stack provides real-time guarantees to ATM virtual circuits by using Rate Controlled Static Priority (RCSP) queuing. ATM signalling is performed using a protocol similar to the Real-Time Channel Administration Protocol (RCAP).
Internet protocols supported include large subsets of IP, TCP, and UDP. In particular, the simulated TCP implementation performs: connection management, slow-start, flow and congestion control, retransmission, and fast retransmit.

The bulk of INSANE is written in C++. Customisation and simulation configuration is performed with Tcl script. Some version of INSANE has been tested and determined to work in the following hardware and platforms:

- DEC Alpha AXP series, Digital UNIX 3.2
- DEC DECstation 5000 series, Ultrix 4.2A
- HP PA-RISC 9000/700 series, HP-UX 9.03
- IBM RS6000 series, AIX 3.2.5
- Intel x86, BSDI BSD/OS 2.0
- Intel x86, FreeBSD 2.1.0 and 2.1.5 release
- Intel x86, NetBSD 1.0A
- SGI Indigo, Irix 5.3
- Sun Sparcstation, Solaris 2.3, 2.4, and 2.5

Advantages

Although there is no graphical user interface, an optional Tk-based graphical simulation monitor provides an easy way to check the progress of multiple running simulation processes. It is designed to run large simulations whose results are processed off-line. Besides, it works well on distributed computing clusters.

Limitations

The simulator is restricted to work in limited hardware and platforms as listed above. Furthermore, INSANE currently requires the following other software packages to run the system:

- g++ (version 2.6.3 or greater)
- libg++ (any version consistent with the installed g++)
- GNU make (pretty much any recent version)
Tcl (version 7.3 or greater) and Tk (version 4.0 or greater) are required for the simulation monitor, but is not necessary to run simulations. INSANE has been tested to use with Tcl 7.5 and 7.6, and Tk 4.1 and 4.2.

2.1.3.5 DELSI 1.1

DELSI 1.1 is also named as Delphi Simulation is a discrete-event simulation tool. Its purpose is to simulate the queuing of systems with complicated logic [17]. DELSI is implemented as a set of components for Borland Delphi 3.0 and 4.0, based on Piecewise Linear Aggregate Simulation Architecture (PLASA). DELSI's internal simulation logic has been optimised for high loading that is used with a large number of transactions in the model. The model's logic is implemented using Delphi event handling.

Advantages

The development of DELSI within the Delphi Rapid Application Development (RAD) environment allows users to combine DELSI with other components, which include GUI design, report building and databases.

The simulation system is designed by the use of object-oriented approach. The whole model and processing objects (so called blocks) are implemented as Borland Delphi components. The transactions are implemented as objects.

Limitations

DELSI is commercial simulation software, which the actual building components are not clear to the user or programmers who fragmented the source codes. Besides, The use of DELSI is only limited to the use Borland Delphi tool, which is not widely use and not portable to other platforms. Borland Delphi is a visual programming tool for Turbo Pascal programming language.

Although it is a commercial simulator, it has claimed to be a non-expansive simulation workplace for developing valuable applications [17]. Besides, a trail-version of software is available for trial purposes.
2.1.3.6 NIST ATM/HFC Network Simulator

The ATM/HFC network simulator is a tool to analyse the behaviours 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 procedural 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 component [18].

Advantages

The user could create different network topologies. Besides, the users could 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 requirements consider a large number of parameters.

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

The simulator only can run on UNIX or LINUX platform. This limits the simulator to execute in limited platforms.

2.1.3.7 Libra: An Object-oriented ATM Network Simulator

Libra is an object-oriented simulator developed for simulation and fast prototyping of ATM networks. It is designed and implemented to address the diverse issues encounter in the simulation of any large-scale ATM network, namely, modelling of source traffic, network
control, switch architecture design, and effective interaction of all these elements in an integrated fashion [19].

The simulator is built based on a well-established framework, called Ptolemy. There are several unique issues that make the simulation more suitable to the analysis of ATM networks.

**Advantages**

Its developed in an object-oriented programming environment (inherited from Ptolemy) provides the means for a modular, hierarchical, and heterogeneous description of ATM networks. Simulation of new development can be easily integrated into the environment.

Its multimedia device interface provides tools for qualitative, quantitative and analysis of ATM networks. For example, supporting the generating and sending of cells for video across a simulated ATM network and displays the result at the receiving end.

Its ability to interact directly with the real network enables first prototyping and incremental construction of an overall ATM network.

**Limitations**

Since this simulator emphasises on object-oriented concepts, therefore it has all the object-oriented advantages, which includes modular, extensible, portable as well as reusable. The only limitation is that the simulator might not suites a conventional procedural programmer to understand the model. Besides, it does not include design model for multithreaded timing operation as well as web-enable feature.

**2.1.3.8 Summary of Existing Simulators**

The following table summarises the various simulators studied in the previous section:
<table>
<thead>
<tr>
<th>Simulator</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>
| REAL Network Simulator       | • A test-bed for packet flow and congestion control  
• Flexible  
• Behaviour can be modified for various components | • No interactive modelling environment  
• No GUI (except REAL 5.0)  
• Programmers must know programming language |
| UCB/LBNL/VINT Network Simulator - *ns* | • Test on TCP packets  
• Has queuing algorithm  
• Has Random Early Detection method  
• Object-oriented | • Only supports TCP packet simulation  
• Programmer must know C++ programming language |
| OPNET Modeller Simulator     | • Object-oriented  
• Highly configurable  
• Hierarchical approach  
• Simulation of existing and future protocols  
• Rich in features | • Limited to specific platforms  
• Very costly |
| INSANE Simulator             | • Simulation monitor allows checking of multiple processes  
• Able to handle large simulations offline | • Limited to specific platforms |
| DELSI 1.1                    | • Able to handle large transactions  
• Able to combine with other components  
• Object-oriented using Delphi components | • Core components may be unclear  
• Limited to Borland Delphi, which is not popularly used |
| NIST ATM/HFC                 | • Able to handle and configure many topology and component parameters  
• Has GUI | • Uses procedural approach, which results in overlapping functions  
• Limited to UNIX and LINUX platform |
| Libra Object-oriented ATM Network Simulator | • Object-oriented  
• Modular, hierarchical and easy to integrate | • Does not suit procedural programmers  
• Does not have multithreaded design for simulator  
• Not web-enable |

From the analysis of the existing simulators, only *ns*, OPNET, DELSI and LIBRA are object-oriented modelled simulator. While the rests are written in procedural programming approach. Hence, they still do not match this project objective. However the algorithm for packet
flowing and congestion control scheme can be taken as references for this project. OPNET simulator is a very powerful tool for simulation on ATM networks. It is rich in features but it could be an expansive tool for small-scaled network simulation.

Throughout the survey on network simulators, Libra simulator has been the most suitable simulation tool constructed in object-oriented approach that suits the main objectives of this project. But it does not employ multithreading features in the model. Multithreading could enable the simulator to “tie a close node” with the global simulation clock. The clock is an important object of a network simulator. The reason is that processes could be executed in a parallel and concurrent manner, each processing objects could differ in each timing process.

Although Libra can be easily integrated with other objects, however its ability to integrate its components into multi-platforms is difficult.

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Discrete event simulation</th>
<th>Object-oriented</th>
<th>GUI</th>
<th>Multi-threading</th>
<th>Web-enable</th>
<th>Platform independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL</td>
<td>√</td>
<td>X</td>
<td>√</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Ns</td>
<td>√</td>
<td>√</td>
<td>Poor</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>OPNET</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>INSANE</td>
<td>√</td>
<td>X</td>
<td>Poor</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DELSI 1.1</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>NIST</td>
<td>√</td>
<td>X</td>
<td>√</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ATM/HFC</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LIBRA</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

From the comparative analysis above, all of the simulators does not implement multithreading operation, not web-enable as well as not platform independent. They are highly reliable on specific platforms.
Therefore in this project, one of the objectives is to model an ATM simulator with multithreading feature. Besides, the simulator shall be multi-platform and can be ported into the world-wide-web in the Internet environment.

Besides comparing existing simulators, a study of the designs of other developed simulation models is made as well. This is to acquire a better idea in modelling techniques for this ATM network simulator.

### 2.1.4 Study on Several Network Simulation Models

In the previous section, various simulators are reviewed. This section continues to review the simulation and management models done by most researchers. Those models are available in most journals and papers presented by researchers. In the study, the simulation models to be discussed are as follows:

- An object-oriented network model for development of integrated management system
- An spiral network architecture
- An object-based network simulator
- An ATM switch simulation tool based on the C++ object-oriented programming language.

The models are based on object-oriented approach.

#### 2.1.4.1 Object-oriented Network Model for Development of Integrated Management System

Object-oriented modelling has been proposed to be an effective means of dealing with the complexities of modelling resources and management functions in a network. Research has been done by identification and classification of objects to be managed in the network [20].

In this network management system, several models were built. The basic structure starts with a generic network model, which forms the framework for the specialised objects representing specific network modelled. All objects are derived from the Top managed object, which is an OSI defined managed objects. Under the Top managed objects are high level abstract objects
that represent the different main views of the network. The NetResource object represents the network objects essential for the operation of the network. The Control and Administration objects would handle the management and service aspects of the network.

2.1.4.2 Spiral Network Architecture

In [21], the network architecture is re-designed using object-oriented principles. The non-traditional Spiral architecture was chosen because it features strengths that does not included in the traditional computer network architectures. The Spiral computer network architecture is a highly fault-tolerant, easily expandable, self-routing network topology developed in the 1980’s.

The minimum Spiral module contains four modules. Each module is composed of four fully connected nodes. Also present are spiral links attached to each node, which form the connection between modules.

![Figure 2.2: A single Spiral module.](image)

2.1.4.3 Object-based Network Simulator

Many simulation systems developed in the past are influenced by some particular computer platforms. In [22], investigation and simulation study of network problems was done by the design of an object-based model so that it could come out with a flexible simulation environment for computer network studies. A flexible network simulation system should featuring extensibility, reusability and portability.
The system was designed using Java programming language with the package approach, which greatly increased the reusability and extensibility of the development simulation. Core of this system is the object-oriented discrete event simulation engine, which includes the scheduler, definitions for the attributes and behaviour of basic simulation entities and data structures for the implementation needs such as binary trees and queues etc.

The simulation system is open-ended and allows other packages to develop and incorporated later. The simulation overviews a centralized simulation, which interacts with the components of configuration on layout, test editing, object-oriented discrete and graph plotter.

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

A software package to simulate ATM switches written in C++ programming language is presented and its main modules and objects are described [23]. The package consists of several modules: General purpose 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, a graphical user interface is implemented for configuration of simulation.

2.1.4.5 Summary on Network Simulation Models

From the review of four different types of simulation models, a general idea for object-oriented model can be observed.

From the first model of section 2.1.4.1, its basic structure is divided into two main parts. The first part is the resource that essential for network operation. The second part is the control object that could handle management controls.
In the second model of section 2.1.4.2, the Spiral model showed interconnected objects. Therefore, its modules are represented as objects. The modules are linked up as a network. In section 2.1.4.3, an object-based network simulator is developed using Java and object-oriented approach. This simulation model has matches part of the objectives for this project. But it does not include multithreading operation and the web-based enable model. Lastly, in section 2.1.4.4, the ATM switch simulation tool has fulfilled the criteria for buffer management for switch in this project. Unfortunately, it does not include link management features, multithreading as well as web-based criteria in the switch simulation tool.

From an overall view, the four discussed simulation models have set strong idea in developing an object-oriented model. Unfortunately, they do not emphasis on congestion control mechanism, multithreading and web-enable features in the models.

2.1.5 Conclusion on Computer Network Simulation Review

The review of the various simulator approaches as well as the simulation modelling methods has provided valuable ideas that are useful in producing and developing of a simulation model for this project.

Firstly, the review of different simulators available has provided an idea of developing a network simulator. This includes the simulator advantages that can be integrated as well as limitations that can be overcome. Based on the summary of existing simulators in section 2.1.3.8, their major strengths can be referred to and incorporated into the object-oriented modelled simulators in order to have the advantages of portability, extensible, reusable, flexible and configurable components. However, those simulators do have built-in congestion control methods that are useful in this project.

Besides their advantages as stated in section 2.1.3.8, there are also limitations that must be overcome. Most of the simulators are not object-oriented. Therefore, the development of ATM network simulator for this project should overcome these limitations. Most of the simulators are limited to only several types of platforms. This limitation can be overcome if this project is built on a cross-platform portable programming language. One of the solutions for developing a cross-platform simulator is to build a web-based simulator, which could be
executed across the Internet. Concurrency is not mentioned as an important feature in most of the simulators. Processes of a network simulator have to communicate concurrently with the global timing instead of sequentially. Therefore, a programming language that could provide communication of objects in a concurrent manner is needed.

A review that covers simulation models is also discussed. From its summary in section 2.1.4.5, the simulation models are of the same design – using object-oriented approach. The models possess their own definition on objects as well as in the communication between their objects.

From the review, an idea of resource-executor object model is gained. Besides defining the objects used in the model, they are categorised as a resource or an executor object. The purposed of this model is to keep the objects in a more planned way. The model would not be too complex for implementation. Furthermore, the resource and executor objects could be extensible and configurable when necessary. Lastly, the model could implement a parallel execution among the executor objects that processes multithreading operation.

From the review of the various simulators and simulation models, a simulation model for the project has been agreed upon – the resource-executor model. In the following sections, review on programming approaches will be discussed.

2.2 Programming Approaches

The selection of programming language is very important in developing a simulator or any application programs. Therefore, the advantages and disadvantages of several programming languages need to be taken into consideration. Hence, comparison between procedural programming and object-oriented programming is made.

2.2.1 Procedural Programming

In the early days of computing, programming was an extremely procedural process. The procedural languages placed code into blocks called procedures or function. 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. 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. The computer processors own language - machine code, is procedural. So the translation of the procedural high-level language into machine code is straightforward and efficient. In addition, procedural programming has a built-in way of splitting big lists of instructions into smaller lists - the functions [24].

In a procedural-based programming language, a programmer writes out instructions that are followed by a computer from start to finish. For example, a procedural-oriented program might work like this [25]:

![Diagram](image)

*Figure 2.3: Simple procedural program*

A more complex procedural-based program might introduce logical branches such as in Figure 2.4.
Figure 2.4: Complex procedural program

The goal of each of these blocks act like a black box, which completed one task or another. The strength of this type of programming believed that one could always write these functions without modifying external data. On the other hand, a difficult problem with this language method is to write all functions that do not modify data outside their boundary. So, when functions began changing data outside their boundary (in C this is done by passing a pointer), a problem called coupling began to surface. The presence of coupling meant that when each method cannot just be tested individually. It has to make sure that the method did not corrupt external data. This increases the difficulty in testing.

As the programs become increasingly complex and large, object-oriented programming language is favoured in order to overcome the drawbacks of procedural programming.

2.2.2 Object-oriented Programming

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. It absorbed the existing classes’ attributes and behaviours and embellishing them with capabilities require by new classes.
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.

Objects are the central idea behind OOP. The idea is quite simple. A method is similar to a procedure. The basic idea behind an object is its methods simulation. Most programs are written with very little reference to the other objects of the program. In object-oriented methodology, a program should be written to simulate the states and activities of the objects. This means, apart from looking at data structures while modelling an object, methods associated with that object should be considered as well. In another words, the methods are functions that modify the objects attributes.

The concept where objects contain both data and methods is referred to as encapsulation. Encapsulation could hide unimportant implementation details from other objects. This could provide modularity as the source code for an object can be written and maintained independently. Similarly, one does not need to know how a class is implemented, but just knowing which methods to invoke.

OOP approach has several key benefits. These are as follows:

- **Extensibility** – By modification of existing objects, new features can be added to the system where changes on new objects can be done.
- **Maintainability** – Maintenance and modification of objects can be done individually.
- **Reusability** – Objects which are used in a system can also be used in another newly built system with little or no changes.
- **Simplicity** – It is simple and less complex using the OOP approach while building programs, which attempts to model the objects interactions of the real world. Any changes are easy to modify with no much affect within the entire system.
- **Modularity** – Objects within the program are individual separate entities. The internal functions are isolated and de-coupled from other objects in the system. This solves the problem of coupling in procedural programming approach.
2.2.3 Conclusion from the Programming Language Approaches

The programming approach chosen must fit the simulation model that has been decided upon in section 2.1.5. The programming language must provide the features of an object-oriented programming approach, which had been decided earlier in section 2.1.5.

Since the procedural programming approach suffers from the problem of coupling, this programming language could not provide extensibility and flexibility for the model decided. The coupling problem could make the program more complex and troublesome to implement.

Thus, the object-oriented approach should be the choice to fit the simulation model. This approach could overcome the limitations of the procedural programming approaches. The object-oriented approach creates code that is highly reusable and extensible. More extensive features can be added for more capabilities in the methods of object-oriented approach. Beside, modification can be done on objects when needed, making the program code highly reusable. Finally, changes in the implementation of objects can be done. These changes not affect the operation of other objects. This made the programming codes more flexible to changes.

Therefore, the object-oriented approach is an ideal approach for use in this project. The use of the object-oriented approach provides extensible features to the simulator. It is simple, modular, modifiable and easily maintainable and reusable.

2.3 Programming Language

The programming language used in developing the simulation model for ATM network simulator depends largely on the features provided by the programming language. Moreover the programming language must provide sufficient features to meet the needs of the simulator to be developed. Since object-oriented programming approach is the decided choice, the programming language considered should meet the requirement of an object-oriented programming approach. Additionally, other features of the language should be able to meet the other additional requirements for this project.
Therefore, after taking into considerations the features and requirements needed for the simulation model and architecture, the Java programming language is found to be most suitable for this project.

### 2.3.1 Java Programming Language

Java is just a small, simple, object-oriented, interpreted or dynamically optimised, byte-coded, architecture-neutral, garbage-collected, multithreaded programming language. Besides, it has a strongly typed exception-handling mechanism for writing distributed, dynamically extensible programs. Java is developed by Sun Microsystems. It is a powerful programming language built to be secure, cross-platform and international [22].

#### 2.3.1.1 Java is Object-oriented

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 trend. The object-oriented paradigm meshes well with the needs of client-server and distributed software. Benefits of object technology are rapidly becoming realised 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 greater lifetime than 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.

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 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.
2.3.1.2 Java with Exception Handling
In Java, only single inheritance is supported but multiple implementations of interface class is allowed [22]. 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. The environment of the component that encounters difficulties is programmed to "catch" that exception and throw an "exception" to deal with it.

2.3.1.3 Java Strings and Characters
Java eliminated the use of pointers in C and C++. Then replaced it with references, which prevents program from accessing illegal areas of the system's memory. This reduces the chances of making mistakes with the dangerous pointers.

2.3.1.4 Java with Multithreading
Concurrency is very important in a simulation model as there might be many objects doing their own processes at the same time. It is impossible to allow them execute processes in a sequential manner, 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 [26].

In Java, programmer specifies that application contains threads of execution and the program may execute concurrently with other threads. This operation is called multithreading. This powerful capability is not available in C and C++. Instead, in C and C++ they are single-thread-of control languages, which is also referred as sequential programming language.

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 simulation process. A general clock is used to control and synchronised the execution of processes. They are designed to run concurrently, at the same time waiting for each other to complete their tasks.
2.3.1.5 Why using Java

Java programming language was chosen for the development of this ATM network simulator due to of several features available. The most important reason is that it is an object-oriented approach programming language. It is important to have a programming language that fully supports the concept of object-oriented.

Besides, an important feature of Java programming language is the support of concurrency simulation model - multithreading. As one of the objectives for this project is to have parallel processing objects within the simulator, the programming language chosen must support this feature. In C++ programming language, it can only have single threading in marinating operation processes. Therefore, Java programming language is the choice because it supports multithreading operation.

As compared with C++ programming language, the used of dangerous pointers is eliminated in Java programming language. Java programming language uses references in referencing other attributes and methods in its code.

Another advantage of using Java programming language is that it is a web-enable application. The simulator built can be executable across the Internet by extending the application to a built in objects for Java programming language named Applet. The deployment of application over the world-wide-web is important as it enable multi-platforms execution, instead of running on limited platforms.

Since Java has more advantages as compared with other object-oriented languages, such as C++, therefore, it is the only choice made for developing this proposed ATM network simulator. To develop the simulator, Java still requires various tools to built, debug and compile the programming codes. After a lengthy discussion, Jbuilder is chosen as a companion for Java.

2.3.2 Programming Language Tool

The ability to make use of programming language tools is important in order to make full use of the Java programming language. These tools should include those essential tools such as
the language compiler itself and in addition programming aids in Java such as debuggers, graphical user interface and language libraries.

**2.3.2.1 Borland’s JBuilder**

JBuilder has the ability to specify a particular Java Development Kit (JDK) to use within a project by choosing a particular JDK switches the version of AppletViewer.

The newest release of JBuilder Enterprise provides the following innovations for distributed development:

- Complete Java 2 support with the ability to switch JDK’s
- Visual development of distributed applications
- Enterprise Java Bean support
- Distributed debugging capabilities
- Java servlet support

Enterprise developers are successful with JBuilder because it allows them to focus on building business logic rather than system and infrastructure code [27].

The features and benefits for Jbuilder 3.0 are shown as below:

*Table 2.3: Features and Benefit for Jbuilder 3.0*

<table>
<thead>
<tr>
<th>Feature</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java 2 Platform</td>
<td>This is an extremely important consideration for both server side and client side development. The Java 2 platform brings a complete set of functionality to enterprise developers.</td>
</tr>
<tr>
<td>Support</td>
<td></td>
</tr>
<tr>
<td>Support for 100% pure Java</td>
<td>It generates pure Java code without proprietary extensions and class libraries.</td>
</tr>
<tr>
<td>JDK Switching</td>
<td>If the Java platform evolves to a new version, it is important to deploy to new environment. The switching can be done in Jbuilder 3.0.</td>
</tr>
<tr>
<td>Distributed Debugging</td>
<td>It is capable to step line by line through a remote call to a virtual machine on perhaps another machine and platform.</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Servlet Support</td>
<td>It has full support for Servlet development, and include tools to simplify the development and testing of these extensions.</td>
</tr>
<tr>
<td>JFC/Swing Support</td>
<td>On the client side of an enterprise application it allows to get the most out of the JFC/Swing component library. The ability to work with these components visually can save the client side development effort.</td>
</tr>
<tr>
<td>Component Development</td>
<td>One of the most important concepts in enterprise Java development is the need to hide application complexity in reusable components. By building components that perform activities, developers can &quot;divide and conquer&quot; the development effort on the client side.</td>
</tr>
<tr>
<td>JDBC 2.0 Support</td>
<td>All major database server vendors have embraced the all Java JDBC API for connectivity to their products.</td>
</tr>
<tr>
<td>Extensibility</td>
<td>It is likely to have the ability to customise the programs, to create custom wizards and integrate external tools. Having the ability to modify the programs for particular needs will likely be an important requirement for any development team.</td>
</tr>
<tr>
<td>Cross-platform support</td>
<td>In a heterogeneous platform environment it is important that the program can bring the same power to developers on many platforms.</td>
</tr>
</tbody>
</table>

### 2.3.2.2 Why using Borland JBuilder

Borland's Jbuilder Enterprise version 3.0 is chosen as a programming tool for this project mainly because it supports Java programming language.

With the inclusion of a graphical user interface and a powerful editor in Jbuilder, the development environment provides the complete set of tools that meets all the development needs of this project. It covers a straightforward use of the features enabled in the tool.
Besides, JBuilder can be used in Windows platform as well as Linux where a freeware version is available. Therefore, it is possible to switch platform while implementing the simulator. JBuilder 3.0 could support Java 2 platform and perform new additional features such as swing component. And the last but not least, JBuilder 3.0 could have choices in switching Java Development Kits to develop and debug the network simulator.

Borland’s JBuilder Enterprise 3.0 is the choice to develop the ATM network simulator to enable cross-platform development and enabling web-based deployment. Ever since the simulation model, programming approach and programming language is decided, the methods of implementing congestion control will be discussed in the following section. This could lead to the decision on the use of ATM traffic management approaches.

2.4 Traffic Management in ATM Networks

The types of traffic patterns imposed on ATM network as well as its transmission characteristics are different from those of others network. As most packet switching and frame relay network carry only non-real time data traffic, their connections are bursty in nature and the receiver receives incoming traffic in a bursty fashion as well. In addition, tools for congestion control in packet switching and frame relay networks are inadequate for ATM networks.

Traffic management is a resource management problem, which deals exclusively with the mechanisms required to control traffic on the network. Congestion happens when the demand of resource exceeds the availability. There are two types of mechanism to handle congestion: Congestion control, which is done when the network is overloaded or detected, and Congestion avoidance, which is done before the network is overloaded or when congestion is predictable. Congestion management deals with problem matching the demand and capacity of a single network traffic. Traffic management ensure the network bandwidth, buffer and computational resources are efficiently utilized while meeting the QoS.
Besides, there are differences between flow control and congestion control. Flow control deals with the control of a particular flow, whereas congestion control deals with the control of a group of flows sharing a group network resources.

Before dealing with any controls on the ATM networks, the behaviour of the networks should be characterised. Determination of ATM service classes in the network is important to allocate the required bandwidth for each application that will determine the QoS in the ATM network.

2.4.1 Behaviour of ATM Traffic within a Network

The characterisation of the ATM traffic is relevant in making appropriate architecture choices in resource allocation and control mechanism of ATM networks. The performance evaluation of ATM networks was performed under the hypothesis of a uniform and uncorrelated traffic model. But the ATM traffic cannot always be described in the terms of Bernoulli or Poisson source and uncorrelated cells uniformly distributed in the networks as sometimes accepted [28].

ATM has several classes of service using its network. Based on the classification into service classes, the traffic and congestion control management could differentiate the connections in the network and perform necessary control on them. Example is constant bit rate (CBR) that should have highest priority and a uniform flow in the network. Its cell could not be discarded while crossing a congestion-controlled network. Variable bit rate (VBR) connection could have bound with delay variation and cell loss ratio, depending on its real time or non-real time type of connection. Unspecified bit rate (UBR) has best afford connection where no amount of capacity is guaranteed, and any cell may be discarded. Available bit rate (ABR) uses the left over bandwidth in a network, and divided among its type of applications in a fair sharing basis. Besides, it has several mechanisms to guarantee its minimum capacity during its cells transfer. Therefore, the burst of its connection is kept at a minimum rate with minimum risks of cell loss for its application.

Due to the interesting characteristics of an ABR application connection, a review on its traffic management as well as congestion control mechanisms is performed.
2.4.2 ABR Traffic Management

ABR uses the closed-loop control, which in this way it shares the capacity not used by CBR and VBR among its sources. ABR connection provides feedback to sources to adjust the load dynamically and thus avoid cell loss and share the capacity fairly.

In ABR service, the source adapts its rate to a changing network conditions. Information about the state of the network, for example the bandwidth availability, state of congestion, and impending congestion, is conveyed to the source through special control cells called Resource Management Cells (RM cells). The transmission rate is tough important for ABR connection as it uses its specific rates to controls its traffic, as well as its capacity allocation to perform congestion control in the network [1].

2.4.2.1 ABR Rate Control

There are several mechanisms in controlling the cells flow rate in the network. For ABR sources that adapt their transmission rate to the provided feedback, a lower cell loss ratio is guaranteed and it becomes the major distinction between ABR and UBR. Also, the ABR service is appropriate for applications that can tolerate adjustments to their transmission rates and unpredictable cell delays.

However, once this rate-based congestion control algorithm fails, this control would unexpectedly exhibit unstable operation under some parameter setting, which leads to dramatic degradation of the performance [29]. It is happened when the propagation delays of connections is very large.

2.4.2.1.1 Feedback Mechanism

The transmission rate of a cell transfer from a source on an ABR connection is characterized by four parameters:

- **Allowed cell rate (ACR):** The current rate at which the source is permitted to transmit cells. The source may transmit at any rate between zero and ACR.
- **Minimum cell rate (MCR):** The minimum value that ACR may take
- **Peak cell rate (PCR):** The maximum value that ACR may take
- **Initial cell rate (ICR):** The initial value assigned to ACR
A source would start with ACR = ICR and dynamically adjusts ACR based on feedback from the network. Feedback is produced in the sequence of Resource Management (RM) cells. Each cell contains three fields, which are:

- A congestion indication (BI) bit
- A no increase (NI) bit
- An explicit cell rate (ER) field

### 2.4.2.1.2 Cell Flow Control

There are two types of ATM cells flow on ABR connection: data cells and resource management cells. A source receives a regular sequence of RM cells that indicate the feedback to enable adjustment on its rate of cell transmission. The bulk of RM cells are initiated by the source, which transmits one forward RM (FRM) cell for every (Nrm-1) data cells. Nrm is a preset parameter, which usually sets to 32. As the FRM cell is received at the destination, a backward RM (BRM) cell is transmit back to the source, and it follows continuously until the transmission ends. Note that when there is bi-directional traffic, there are FRMs and BRMs in both directions on the Virtual Channel (VC). The ATM switch or destination may change the fields in the BRM in return.

Each FRM cell contains fields as follows:

- CI which usually sets CI = 0 by the source
- NI which usually sets NI = 0 or NI = 1 by the source
- ER which is a desire transmission where ICR ≤ ER ≤ PCR.

It required that each switch have to control congestion by implementing one of the following congestion control mechanisms: explicit forward congestion indication (EFCI) marking, relative rate marking, and explicit rate marking [2].

The ATM switch has a few ways to provide feedback control to the source, there are:
Figure 2.5: EFCI marking

- **EFCI marking**: The EFCI (explicit forward congestion indication) condition in an ATM cell header may be set by a switch as it passes in the forward condition. This would cause the destination to set the CI bit in a BRM cell (Figure 2.5). The CI bit in BRM cell would be set to 1 when congestion occurs.

- **Relative rate marking**: The CI or NI bit of a passing RM cell may be set directly by the switch. In both the FRM and BRM bit would remain the same as it turns around. To get rapid result, instead of waiting for BRM cells from the destination, the switch can generate a BRM with the CI or NI set.

- **Explicit rate marking**: The switch may reduce the value of the ER field of an FRM and BRM.

This means that both the switch and the destination could also signal congestion. ABR technique on controlling congestion is referred to as rate control.

**2.4.2.2 ABR Capacity Allocation**

Besides performing congestion control function, the switch must also provide fair capacity allocation function for the ABR service. An ATM switch should allocate a fair share of its capacity among all connections that pass through this switch point. Therefore, when congestion happens, the switch should throttle back on those connections that are using more than their fair share of the capacity.
The switch algorithm for congestion control and fair capacity allocation fall into two broad categories: Binary feedback schemes that use only the EFCI, CI and NI bits, and explicit rate feedback schemes that use the ER field.

**2.4.2.3 Summary on ABR Traffic Management**

ABR is treated as the only connection type in this ATM simulation model as it has its interesting features to be treated on traffic management as well as congestion control. As compares with other service classes, such as CBR, where the flow of cells is constant and unable to be discarded. There would not be much traffic controls or congestion control to be performed on this connection.

The traffic flow for ABR connection can be controlled by the use of rate control. Several feedback parameters are used to ensure that the QoS of the connection could be achieved. In addition, the capacity allocation has to be fair shared among the applications using ABR in the network. Therefore, the complexity of ABR connections has been determined by its parameters to perform traffic control in ATM network environment. There are several functions and procedures used in the ATM network environment.

**2.4.3 Functions and Procedures for Traffic Control**

Based on Section 1.4.2, there are several functions, which have been defined for ATM traffic control functions. Sets of functions are used to maintain the QoS of ATM connections. There are such as connection admission control (CAC), usage parameter control, selective cell discard, traffic shaping, explicit forward congestion indication (EFCI), resource management using virtual path, frame discard and others.

**2.4.3.1 Connection Admission Control (CAC)**

The CAC function is defined as the set of action taken by the network management during virtual channel establishment, in order to determine whether a connection request can be progressed or should be rejected.
Based on the CAC function, a connection request is progresses only when sufficient resources are available at each successive network element to establish the connection through the whole network based on its service category, traffic contract, and QoS, in order to maintain the agreed QoS of existing connections [30].

In each connection request, the CAC function should be able to derive the following information from the traffic contract:

- Values of parameters in the source traffic descriptor
- Requested and acceptable values of each QoS parameter and the requested QoS class
- Value of the CDVT
- Requested conformance definition of each QoS parameter (peak-to-peak CDV, maxCTD, CLR)
- Service category (CBR, rt-VBR, nrt-VBR, ABR, UBR)
- Other aspect of traffic control that may be requested or assigned is cell loss priority [1].

The CAC function makes use of the traffic contract and the network definition of a compliant connection to determine whether the connection can be accepted or not. Also, the connection traffic parameters needed by usage parameter control and allocation of network resource is determined from the function. In addition, information such as the measured network load may be used while performing the CAC function. This may allow the network to achieve higher network utilisation while meeting the network performance objective. This function is network specific.

**Figure 2.6: Connection Admission Control**
The QoS for a new connection is specified at call setup. The network estimates its state to determine if sufficient resources are available to accept the connection. The CAC function performed by the CAC procedure translates incoming connection parameters into a bandwidth value [31].

2.4.3.2 Usage Parameter Control (UPC)

The UPC function is optional at the User Network Interface (UNI). It is defined as the set of action taken by the network to monitor traffic at the end–system access and enforce the traffic contract. The main purpose is to protect the network resources from malicious as well as unintentional misbehaviour. The misbehaviour could affect the QoS of the other established connections [30].

The monitoring task is performed for VCCs and VPCs by checking the validity of VPI and VCI whether they are associated with active VCCs. Besides, the traffic that entering the network from active VCCs is monitored in order to ensure that parameters agreed upon are not violated. Checking on validity of VPI is run to ensure that the association with active VPCs and the parameter agreed upon for VPCs are in order and not violated.

2.4.3.3 Selective Cell Discard

The end-system of an ATM network may generate traffic flows of different priority using the Cell Loss Priority (CLP) bit.

Selective cell discard comes into play when the network at some point beyond the UPC function starts to discard cells [1]. A congested network element may selectively discard cells that meet either both of the following conditions:

a) cells that belong to a non-compliant ATM connection
b) cells that have CLP = 1

Cells with CLP = 0 are not meant to drop, whereas cells with CLP = 1 are meant for drop when congestion occurs.
This protects the CLP = 0 cells flow as much as possible. However, if CLP = 1 cells are dropped from a compliant ATM connection, it is expected that the CLR objective for the connection, as determine by its conformance definition, would be still met.

However, there is a relationship between the UPC function and the CLP bit. The UPC function first tests the (CLP = 0) flow for compliance and then the combined (CLP = 0 +1) flow. If the tagging option is used, a non-compliant (CLP = 0) cell is tagged but still considered part of the (CLP = 0 +1) flow and subjected to the second test to re-identify the priority setting.

2.4.3.4 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. This is done while meeting the QoS objectives, or to ensure conformance at a subsequent interface [30]. Traffic shaping should maintain cell sequence integrity on a connection.

Examples are peak cell rate deduction, burst length limiting, reduction of CDV by suitably spacing cells in time, and cell scheduling policy.

Besides, traffic shaping may also be used within the end-systems to ensure that the cells generated by the source or at the network are conforming to the negotiated traffic contract.

Leaky bucket algorithm is the most famous algorithm for traffic shaping [30]. This method provides a pseudo-buffer. Whenever a user sends a cell, the queue in the pseudo-buffer is increased by one. The pseudo-server serves the queue and the service-time distribution is constant. Thus, there are two control parameters in the algorithm: the service rate of the pseudo-server and the pseudo-buffer size. This algorithm is also named as Generic Cell Rate Algorithm.
2.4.3.5 Explicit Forward Congestion Indication (EFCI)

If a network element is congested, a congested switch may set an EFCI in the cell header so that this indication may be examined by the destination end-system. This mechanism is a modification of the DECbit scheme [32]. Such switches are called "binary" or "EFCI" switches. The destination then aggregates these EFCI bit information and returns feedback to the source in an RM cell. An initial version of the binary feedback scheme is illustrated in Figure 2.8. In the current specification, the RM cell is sent by the source periodically and is turned around by the destination with the bit-feedback.

Figure 2.8: Initial Binary Feedback Scheme
2.4.3.6 Resource Management using Virtual Path

The initial concept behind network resource management is to allocate network resources in such a way as to separate traffic flows according to service characteristics. So far, the only traffic control based on network resource management defined by the ATM Forum deals with virtual paths [1].

Virtual path in ATM is an important component of traffic control and resource management, it can be used as to:

- be a simplified CAC
- implement a form of priority control by segregating groups of virtual connections according to service category
- efficiently distribute messages for the operation of traffic control schemes
- aggregate user-to-user services such that the UPC can be applied to the traffic aggregate.

VPCs are important in resource management. By reserving capacity on VPCs, the processing that is required to establish individual VCCs is reduced. The QoS parameters that are of primary concern for network resource management includes, cell loss ratio, maximum cell transfer delay and peak-to-peak cell delay variation. All of these parameters are affected by the amount of resources devoted to the VPC. There are alternative ways to group up the VCCs into VPC. If all of the VCCs within a VPC are grouped similarly, they should expected similar network performance. If not, different VCCs in the same VPC could require different QoS. The subscriber for service in the network should be suitably set for the most demanding VCC requirement.

2.4.3.7 Frame Discard

When a network element needs to discard cells during congestion, it is more effective to discard at the frame level rather than the cell level. The "frame" means the AAL protocol data unit (PDU). The network detects the frame boundaries by examining the SDU-type in the payload type field of the ATM cell header. Frame discard may help to avoid congestion collapse and may increase goodput. If done selectively, frame discard may also improve fairness [30].

57
In Generic Flow Control (GRF), frame discard is enabled. But for other service categories in a network, the network may invoke frame discard mechanisms only for connections for which it is specifically enables, either via signalling or at subscription time.

2.4.3.8 Summary on Traffic Control Functions and Procedures Review
Throughout the traffic control functions discussed earlier, each of those functions has its methods in dealing with many types of traffics. The traffics could be of the different types of service categories of ATM networks.

In this particular project, a congestion control mechanism has to be implemented in the development of ATM network simulator. The general functions use for traffic control is needed to build up a concept of congestion control mechanism. When congestion occurs, a reactive control has to be enabled to control the network traffic before it burst. Therefore in ABR connection, controls at buffer can be effective as when frames or cells are discarded. The bandwidth for connections must be fairly distributed among all the connections.

There are several others scheme and policies available for congestion control mechanisms, which are based on the traffic control functions discussed above. The survey on those available schemes could help in the development of a better congestion control mechanism for this project.

2.5 Congestion Control Scheme in ATM Networks
There are several congestion control schemes that have been proposed. But some were not fully accepted by the ATM Forum. Part of the schemes discuss below are based on [3].

2.5.1 Fast Resource Management
A proposal from France Telecom [33] requires sources to send a RM cell requesting the desired bandwidth before it actually sends the cells. The RM cell would be dropped if the switch cannot grant the request. Then, the source would time out and it would resend the request. If successful, the switch would pass the RM cell to the next switch and waited for
positive reply. Until the destination returns the RM cell back to the source, then the source could transmit the burst. There seems to be a delay while negotiation the transmission. Therefore to avoid the delay, in the proposal, an immediate transmission (IT) mode was proposed. The burst is transmitted immediately following the RM cell. If the switch could not cater the request, then both the burst and cells are drop and an indication would be sent to the source.

2.5.2 Delay-based Rate Control
This proposal requires the source to monitor the round trip delay by sending RM cells that contains timestamp continuously. The source uses the timestamp to measure the round trip delay and to reduce the level of congestion. This advantage is that no explicit feedback is needed from the network and it would work even at non-ATM network [34].

2.5.3 Backward Explicit Congestion Notification (BECN)
This method has switches to monitor their queue length and send RM cell back to the source if congested [35]. Then the source would reduce their transmission rate by half on the receipt of the RM cell. If there are no BECN cells received within a recovery period, the rate for the VC is doubled once at each period until it reaches the peak rate. The source recovery period was made proportional to the VCs rate so that the lower the transmission the shorter the recovery period. ATM Forum dropped this scheme, as it is not fair. This is due to the source that receives BECNs might not be the one causing congestion on the network.

2.5.4 Partial Packet Discard
Together with early packet discard method, these methods were first proposed in [36]. This method is to efficiently transport TCP segment over ATM networks. Whenever it has to discard a cell due to buffer overflow, all subsequent cells (except the last cell) belonging to the same packet are dropped. The packet would be retransmitted entirely by the higher-layer protocol. Since this method drops half the packets, on average it improves the network utilization. Although this method provides better performance than TCP over plain ATM, but it is not as effective because it only discard the "tail end" of a datagram. Results have been reported in [37] and [38]. The discarding method is depicted in Figure 2.9. Cells that entered above the congestion threshold would be discarded and the sender will resent the lost cell.
2.5.5 Early Packet Discard

Early packet discard (EPD) was proposed by Romanow and Floyd [36] together with the partial packet discard algorithm. It is observed that a packet consist of several cells and they found that it is better to drop whole cells of one packet rather than randomly drop cells from different packets [39]. This prevents the congested link from transmitting useless cells and reduces the total number of corrupted packets. To implement EPD, the switch drops packets whenever the proportional of the buffer in use exceeds a fixed threshold. When the threshold is reached, the switch's buffer drops the first arriving cell and all subsequent cells of any incoming packet belonging to a VC designated as using EPD. As long as the buffer queue exceeds the fixed threshold, the switch continues to drop packets from multiple connections. The switch emulates a packet-based switch, dropping complete packet of cells.

As in AAL5, when the first bit of the payload type bit in the cell header is 0, the third bit would indicate the end of message (EOM). When a switch queue starts getting full, it would look for the EOM marker and drop all future cells of the VC until the following EOM marker is seen. This scheme might not be fair as the cells arrived at a full buffer may not belonging to the VC which is causing congestion.
Fairness remains a common problem for buffer management when the loss rate of packets of a particular size is concerned. EPD is biased when packets are of different lengths, therefore an Age Priority Packet Discarding (APPD) scheme is proposed to improve the packet level QoS and fairness for application over AAL5 by assigning older packets with higher priority [40]. APPD is used with other known dropping mechanisms such as EPD and PPD. However, the schemes are compared using on-off sources only and are not examined under high-load conditions nor over multi-hops networks.

![Diagram](image)

*Figure 2.10: Early Packet Discard*

A Weighted Fair Early Packet Discard (WFEPD) scheme is proposed, which relies on per-flow state information in that it keeps a record of the bandwidth usage of each individual IP stream [41]. This information is used to compare the actual bandwidth usage with the determined by the fair share of that IP stream. In case of congestion, cells of violating IP streams would be discarded, thus protecting conforming IP streams.

Besides, another buffer management algorithm is done, called the Fair Buffer Allocation (FBA) initially proposed by Heinanen and Kiki and extensively studied by Jain et. al [42]. Upon congestion, packet arriving for connection that occupy more than their fair share would be dropped.
Note that partial packet discard only discards the tail end of the frame when congestion is detected in its interior. Recognizing this shortcoming, the ATM Forum ratified the EPD technique. The EPD schemes proactively determine the state of congestion. When the EPD state is enabled for a given connection, any cells in the new frame(s) arriving on that connection are discarded until the queue threshold falls to the lower value of the congestion limit. EPD is depicted in Figure 2.10 [31].

2.5.6 Random Early Discard

Random Early Discard (RED) method is another buffer management policy proposed by Floyd and Jacobson for use in IP routers [43]. Its goal of RED is to avoid the performance degradation and unfairness caused by Drop-Tail policies. It maintained the average buffer occupancy at a level significantly below the total number of buffers. Packets are dropped with a certain probability (increase with the average queue occupancy), when the average queue occupancy exceeds a minimum buffer threshold. When it reaches the maximum buffer threshold, all arriving packets are dropped.

2.5.7 Link Window with End-to-End Binary Rate

This method presented has combined both good features of the credit-based and rate-based proposal [44]. It uses window flow control on every link and use binary (EFCI-based) end-to-end rate control. The window control is per-link but not per-VC, as in credit-based scheme. Therefore, it is scalable in terms of number of VCs and guarantees zero cell loss. But it was rejected as it contains element from both the credit-based and rate-based camp.

2.5.8 Adaptive Negative Feedback Congestion Avoidance on ABR

An adaptive negative feedback congestion avoidance scheme is proposed to resolve the overall network congestion collapse problem [45]. It is done to eliminate interval timers at the end systems in conventional congestion control scheme. Here, a source periodically sends RM cells containing current cell rate (CCR), explicit rate (ER) and fair share rate. The source utilization is set to 0.95, ER and fair rate fields to 0. A switch computes fair share rate as follows:
Fairshare rate \( = \frac{\text{Link Bandwidth}}{\text{Number of Active Source}} \)

When the destination received the RM cell, it simply returns the RM cell to the source. The ER indication is changed by backward according to the congestion at the switch which is detected by evaluating the change of queue length in a fixed time interval rather than comparing the queue length with a threshold value. Specifically the switch selectively sets,

- ER = fair share rate * ERF (explicit reduction factor which sets to 7/8) for connection with higher rate.
- ER = fair share rate * MRF (major reduction factor which sets to 1/4) for connection with higher rate when the number of cells in the buffer exceeds a high threshold.
- ER = fair share rate if the connection never exceeds higher rate.

Selective feedback takes place when:

- CCR exceeds the fair share rate, and
- Queue length exceeds the low threshold

2.5.9 Summary on Congestion Control Mechanisms Review

From the review of the congestion control mechanisms mentioned above, there are several features enumerates below. These features are highly desirable for congestion control mechanisms of ABR connections in ATM network. There are:

- Fairness
- Simplicity of operation
- Global synchronization

Since ABR mainly carries Transmission Control Protocol (TCP) packet with the use of AAL 5, the congestion control mechanism in this project has to cater for TCP congestion control methods as well. The TCP has its congestion control and ABR uses its control based on minimum cell rate (MCR), peak cell rate (PCR), explicit rate and other cell rates, that are useful for the TCP congestion control mechanisms.
Based on decision made in section 2.4.3.8, frame discard function, explicit cell rate feedback mechanism and ABR capacity allocation are chosen as the congestion control mechanism for this project. The frame discard function will be implemented using early packet discard (EPD) scheme.

This early packet discard scheme (with fair buffer allocation) has fairness consideration where the ABR capacity allocation produces fair shares for each of its connections. The operation of this scheme is made to be simple as in high-speed network, switches and routers need to forward cells at high rates. The global synchronisation issue is a problem related to TCP connections. Whenever a cell is dropped, apart from congestion control mechanism, the scheme should have feedback mechanism to alert the source upon congestion. The ABR’s explicit cell rate feedback mechanism is chosen as the feedback mechanism for the simulator. The combination of EPD with TCP congestion control mechanism seems to provide effective throughput for the average TCP connection [1].

The next section will discuss on the queuing techniques to be used in this ATM network simulator.

2.6 Simulation Queuing Techniques

The techniques used behind the building of a network simulator are started from the background of probability and stochastic processes. Since the simulator has specific cells with its connection information generated into the simulator, the probability and stochastic process is not needed in this project. The simulation queuing techniques continue its discussion on various queuing analysis and queuing disciplines.

2.6.1 Queuing Analysis

Analytic model based on queuing analysis is needed for networking and communication problems as it predict the performance projection of the network. Knowledge of elementary statistical concepts, example: means and standard deviations, and basic understanding of the applicability of queuing theories is required in the analysis performance using queuing theory.
Chapter 2

Queuing analysis is needed especially in the first step before building a simulation model where the reality of a network is being modelled. With the assumption of queuing theory, the result produced can be quite close to the result produced by the simulation analysis.

Queuing analysis theories can be modelled into queuing models. However, there are in fact a number of models, based on the two key factors:

- The manner in which blocked items are handled
- The number of traffic sources.

When an item arrives at a busy server, the item will be blocked. To handle this, the item can be placed in queue, waiting for a free server. The key element of a traffic model is whether the number of sources is assumed to be infinite or finite. For an infinite source, it is assumed to be a fixed arrival rate. For finite source, the arrival rate would be depends on the number of sources already engaged.

Traditionally, queuing theory has been used to model delay in networks. However, it can be also used to model computational load. System scalability of fusion computation related to number of targets is examined [46]. Besides, this theory analysis is modelled as a single M/M/1 queue. It modelled input process related to the sensor state “on” and “off” with Poisson arrivals, and related the queue service rate to computational complexity for using the data [47]. Furthermore, queue of human operation model can be done as well [48].

2.6.2 Queuing Models

Queuing models are needed when there are buffering techniques used to store outstanding requests of a server during busy times. In another way of putting this is to say that arriving requests enter a queue to await service. Therefore, different types of queuing models are examined. There are a single server queue, multiple server queue and a ON-OFF/D/1 queuing model.
2.6.2.1 Single Server Queue

The single-server queuing system is the simplest queuing system. The server acts as a centre to provide service to items. An arriving of items would be served by the server immediately if the server is idle, otherwise, items would be join in a waiting line.

![Diagram of a single-server queue](image)

- \( \lambda \): arrival rate
- \( w \): items waiting
- \( T_w \): waiting time
- \( q \): items in queuing system
- \( T_q \): queuing time
- \( T_s \): service time
- \( \rho \): utilization

*Figure 2.11: Queuing system structure and parameters for single-server queue.*

At a given time, a number of items would be waiting in the queue, the average is \( w \), and the mean time is that an item must wait is, \( T_w \). The server handles incoming items with an average service time \( T_s \). Utilization \( \rho \) is the fraction of time that the server is busy, measured over some interval of time. The average number of item being served and item waiting in the server is \( q \); the average time spend in the system is \( T_q \). As assumed the queue is infinite, then no item ever lost. Here, the departure rate equals to the arrival rate. As the arrival rate increased, the utilization increases and congestion happens. As the queue increased until \( \rho = 1 \), the server become saturated. Thus the theoretical maximum input rate that can handle by the system will be,

\[
\lambda_{max} = \frac{1}{T_s}
\]

2.6.2.2 Multiserver Queue

A multiple server queue can be shown in a generalized simple model with all sharing a common queue. If an item arrived and one of the servers is available, then the item
immediately dispatched to that particular server. This is assumed that all servers are identical. If all servers were free, then there would be no difference. Queue would be form when all the servers are busy.

![Diagram of multiple server queues](image)

**Figure 2.12: Multiple server queues**

The parameters are as the same as the single server queue, except for the utilization. If there are $N$ servers, and $\rho$ is the utilization of each server, $N\rho$ can be consider as the utilization of the whole system. By theoretical, the maximum input rate is,

$$\lambda_{\text{max}} = \frac{N}{T_s}$$

### 2.6.2.3 ON-OFF/D/1 Queuing Model

A typical $\Sigma$ ON-OFF/D/1 discrete time queuing model is modelled for an ATM switch output buffer [40]. It ensure that at each time slot, the most is only one cell can be transmitted. While at most $N$ cells may arrive, where $N$ is the number of traffic sources. All sources are assumed to be the same characteristics. This model uses probability to determine the change of ON/OFF state. All the cells of a packet are assumed to arrive at the rate 1 cell/slot within an ON period. With probability $r1$, the source were to change from ON to OFF state and with probability $r2$, the source were to change from OFF to ON state. When the source is in ON state, it is "Active" and its status is eligible, while it will be in eligible "idle" state when it is in OFF duration. The relationship of the state and status of an ON-OFF source is summarized in Table 2.4.
Table 2.4: The relation of the state of a source with its own on-off duration

<table>
<thead>
<tr>
<th>Status</th>
<th>On duration</th>
<th>OFF duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eligible</td>
<td>Active</td>
<td>Idle</td>
</tr>
<tr>
<td>Discarded</td>
<td>Blockeded</td>
<td>NA</td>
</tr>
</tbody>
</table>

When the cells arrive at the beginning of the slot, it can be served immediately, if the server is idle. The service discipline is first come first serve, while the associated buffer management arbiter controls the buffer management.

2.6.3 Basic Queuing Relationship

Consider the experience of a single item arriving. When the item arrives, there would be average \( w \) item waiting, the average time waiting for service is \( T_w \). In reasoning, total of \( w \) item would be arrived when item arriving in the rate of \( \lambda \) in the \( T_w \) time. Thus, \( w = \lambda T_w \). For a single server, the average number of items being served is \( \rho \). Therefore, the item waiting, \( q = w + \rho \) for a single server and \( q = w + N\rho \) for \( N \) servers.

The fundamental task of a queuing analysis is as follows: input information, are:

- Arrival time
- Service time

Output information, are:

- Item waiting
- Waiting time
- Items queued
- Queuing time.

Assumptions are needed in calculating the timing of queuing analysis. The most important assumption is that the arrival rate obeys the Poisson distribution, which is equivalent to saying that the inter-arrival times are exponential. The arrivals occur randomly and independent of one another. With this assumption, there can be many useful results being obtained if only the
mean and standard deviation of the arrival rate and service time are known. Matters can be simplified if assumed that the service time is exponential or constant.

With the basic queuing relationship knowledge, the following section will further discuss the queuing discipline for a queuing model.

2.6.4 Queuing Discipline

Queues are dynamic collections, which have some concept of order. In the queuing models discussed earlier, queuing disciplines are needed while dispatching a cell from one end to another end. This can be either based on the order of entry into the queue, First-in-first-out (FIFO) or Last-in-first-out (LIFO) queues.

Due to the drawbacks of traditionally used First-in-first-out (FIFO) queuing discipline, there are a few queuing disciplines being introduced. There are five types of queuing method will be discuss in the following section, that include the fair queuing, bit-round fair queuing, weighted fair queuing and priority queuing.

2.6.4.1 First in First Out (FIFO) and Last in First Out (LIFO) Queuing

In computer programming, FIFO (first in first out) is an approach that handles requests from queues or stacks. This approach is used so that the oldest request will be handled next [49]. In ATM network simulation, the group of cells are lined up in a queue. The cells objects are ready to queue without priorities.

LIFO (last in first out) is an approach in which the most recent request is handled next and the oldest request does not get handled until it is the only remaining request on the queue. Although LIFO seems “unfair”, it may be efficient in some cases. But definitely, this approach is not useful in ATM cells queuing.

2.6.4.2 Fair Queuing (FQ)

Nagle proposed a scheme called fair queuing (FQ), which is to overcome the drawbacks of FIFO queuing [50]. Here, a router maintains multiple queues at each output port. The router maintains one queue for each source and it would also be possible to maintain one queue for
each flow. With fair queuing, each incoming packet is placed in the appropriate queue, servicing the queue in round-robin fashion by taking one packet from each non-empty queue in turn and empty queues are skipped over. This is fair that each busy flow gets to send exactly one packet per cycle.

Unfortunately, there is a serious drawback to the fair queuing scheme. Short packets are penalized where flows with longer average packet size can pass more capacity to go through. The disadvantage can than be overcome with bit-round fair queuing (BRFQ), which uses packet length as well as flow identification.

2.6.4.3 Bit-round Fair Queuing (BRFQ)
Bit-round fair queuing is an improvement over fair queuing (FQ) and first-in-first-out (FIFO) scheme. It fairly allocates the available capacity among all active flow through a node. However, it does not able to provide different amount of the capacity for different flows. This differential allocation capacity is needed to support QoS transport in ATM networks.

Firstly, it considers only transmission of one bit from each queue at each round of transmission. Now, each busy source receives exactly the same amount of capacity. If there are \( N \) queues and each queues always has a packet to send, then each queue receive \( 1/N \) of the available capacity. This bit-by-bit approach is called processor sharing (PS).

Since BRFQ is designed to emulate a bit-by-bit round-robin discipline, it is implemented by computing virtual starting and finishing times as PS were running.

2.6.4.4 Weighted Fair Queuing (WFQ)
Data is being transmit by entire packets rather than individual bits. Weighted fair queuing is a method of sorting packets into different conversion from hosts. Examples of different conversions are, all packets from transferring a large file by FTP, transmitting data entered into a telnet application, sending an email etc. This works to ensure that low volume traffic (like Telnet) gets an equal chance to be transmitted as high volume traffic (like FTP). Fair queuing groups together packets from different messages and places each of these in their own queue. Then it gives priority to smaller messages, which do not require much bandwidth and would service these packets before it services packets belonging to larger messages. It would service with larger bandwidth equally [51].
WFQ enables a router to assign the appropriate weight to each flow to guarantee service. Because it is possible to guarantee and upper bound on delay. In the case, the formula shows

\[ D_i < \frac{B_i}{R_i} + \left( \frac{K_i - 1}{R_i} \right) \sum_{m=1}^{K_i} \frac{L_{max}}{C_m} \]

Where

- \( D_i \) = maximum delay experienced by flow \( i \)
- \( B_i \) = token bucket size for flow \( i \)
- \( R_i \) = token rate for flow \( i \)
- \( K_i \) = number of nodes in the path flow \( i \) through the internet
- \( L_i \) = maximum packet size for flow \( i \)
- \( L_{max} \) = maximum packet length for all flows through all nodes on the path of flow \( i \)
- \( C_m \) = outgoing link capacity at node \( m \)

### 2.6.4.5 Priority Queuing

Priority queuing enables all packets from the high queue transmitted before any packets from the queue are transmitted. When the high queue is empty then packets from the medium queue will be transmitted until that is empty, next will be the normal queue and last is the low queue.

Since priority queuing works packets from the higher priority queues, lower priority queues might not get transmitted in a timely manner, or possibly never get transmitted at all [52]. Therefore, this queuing method could be unfair to other queues.

### 2.6.5 Summary on Queuing Techniques Review

The queuing models suitable for this project can be the single queuing models. This model simple to use and each of the server queues could be modelled as the buffer at a switch or input/output queue in a link. Once a cell entered the server, it will serve the cell if the queue is empty. If the queue is not empty, the cell will be posed at the back of the queue. A traditional first in first out queuing discipline will be used in this project, as the queue in each server has
no priority checking on each cell. Each cells are treated equivalently, the server will server the first going in cell into the server queue.

2.7 Chapter Summary

This chapter has covered the primarily research background of this project, reviewing some knowledge which is relevant as well as why certain decisions are taken.

This chapter has covered the review of various network simulators available. The review of these simulators includes the strengths and weaknesses of these simulators. Some other simulation models are also reviewed. Their reviews generally support the basic concept of the kind of ATM network simulator to be developed.

It also discussed on programming approaches It reviewed the procedural and object-oriented approaches in detail. The strengths and weaknesses of both techniques are analysed. Based on this review, object-oriented programming technique is a much better choice. And among the various object-oriented based programming languages, the Java programming language is chosen after an extensive research is made. And finally a programming tool is chosen as well to supplement this language.

Besides, it has covered the ATM network traffic management approaches or techniques. Discussion has been carried out on the various traffic management approaches, especially using the ABR service connection. Finally, the selection of a particular functional method is made for this project. After all the detailed congestion control schemes are discussed, it is then decided that ABR connection.

Lastly, it has covered a review on queuing techniques available. The reason for using single server and first in first out queue is discussed in detail at the concluding part of this section.

The next chapter will discuss on the simulation overview for this project. The overall idea and methods proposed for this project will be discussed in detail.