CHAPTER 6: CONCLUSION

There are numerous architectures for ATM switches and each has its own merits and drawbacks. This report presents a general Banyan switch for simulating switching architectures in ATM networks using object-oriented programming.

Object-oriented programming is a type of programming in which programmer defines not only the attributes, but also the types of operations that can be applied to the attributes. Both attributes and operations are encapsulated into an object. One of the principal advantages of object-oriented programming techniques over procedural programming techniques is that they enable programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits features from existing objects. Moreover, object-oriented programming is simple because it models the real world objects. It has attributes such as modifiability, extensibility, maintainability, and reusability.

Java is an object-oriented language, which is similar to C++ but it is more robust than C++ with no pointer references to external data. The other great features of Java are built-in support for multithreading and the ability of the threads to run simultaneously. Finally, the ability of Java to work under different platforms and different browsers fulfil the objectives of this project.

This project is a simulation of Banyan 4x4 and Banyan 8x8 switching architectures. Every switching element has its own input buffer and there are output buffers at the outlets of the Banyan, which is connected to outgoing link. Incoming ATM cells are placed in input buffer before served by switching element. This switching element will take cells from the head-of-line of its two inlet for switching. If these cells are addressed to same outlet, blocking occurs and the other cell will stop from switching. The cell selection for transmitting is based on the PCR, MCR, and TCT values to ensure the fairness and QoS of each ATM application.

Designing of this simulator is based on the object-oriented approach where the classes with corresponding attributes and functions are defined first. This is followed by the algorithm design such as:

- how the ATM cell should be switched when using Banyan 4x4 or Banyan 8x8
- how to fulfil the fairness of switching between two inlets of an switching element
- how to ensure the QoS of ATM application.

The implementation phase covers the part of coding for this simulator. Every class and algorithm, which have been designed, should be included in Java code. Testing is done on all of the Java classes, followed by the algorithm testing. The system testing is performed to ensure the overall behaviour of the system. This testing process is also concerned with finding errors, which result from unanticipated interaction.

This project managed to achieve the overall project objectives and goals, i.e. development of an object-oriented, multithreading, cross-platform and web-enabled ATM switching simulator. Lastly, the following highlights strengths, limitations as well as the proposed future enhancements.

System Strengths

- The simulator is fully object-oriented whereby all the functions and modules are built in class. In addition, problem of coupling is highly reduced.
- The separation of super class, *Switch* class and subclass, *BanyanSwitch* class makes the switch design extensible. New switch model can be built by creating a new class that simply inherits the general attributes of an ATM switch from *Switch* class. Therefore, source code of *Switch* class need not be known or modified when building a new child class.
- *BanyanSwitch* class is extensible too. New Banyan 16x16, 32x32 or higher can be built by simply adding the NxN switching control function. The syntax and

logic are similar to Banyan 4x4 and Banyan 8x8, therefore making the work easier.

- The simulator is built using Java *Thread* technology. When more than one switch is created, switching environment becomes more realistic where all switches run concurrently instead of sequentially.

System Limitation

- The simulator works based on Banyan 4x4 and Banyan 8x8 architectures. Other switching models such as Knockout switch are not presented.

Future Enhancements

- In order to provide a better graphical user interface for data capturing, the ATM cells flow should be shown on the screen.
- It is hoped that the switching model is extended to Banyan 16x16 and onwards.
- It should include other switching models, which allows a user to select the architecture.
- It should also allow the execution of performance comparison for different types of switching architecture.