77

PERFORMANCE OF A ATM

LAN SWITCHING FABRIC

Lianghong Yu B.Sc (Eng.)

Post-Graduate Student

DEPARTMENT OF ELECTRICAL ENGINEERING

UNIVERSITY OF CAPE TOWN

PERFORMANCE OF A ATM LAN SWITCHING FABRIC

Lianghong Yu B.Sc (Eng.)

Post-Graduate Student

DEPARTMENT OF ELECTRICAL ENGINEERING

UNIVERSITY OF CAPE TOWN

Full research submitted to the Faculty of Engineering , University of Cape Town, in complete fulfilment of the requirements for the degree of Master of Science in Electrical Engineering.

Cape Town

1998

Declaration

I declare that the work contained in this dissertation is my own. All sources of reference material referred to for use in this research have been acknowledged. This dissertation has not been submitted for graduation purposes at any other educational institution.

Signed by candidate

(Signature of Candidate)

18 Day of June 1998.

Acknowledgements

Mr. M.J. Ventura of University of Cape Town, who is the project supervisor and provided guidance.

I am grateful to Telkom S.A. for all the financial assistance and providing the simulation software OPNET and for their support at various stages when needed.

Abstract

This thesis provides a focus on the architecture of a high-speed packet switching fabric and its performance.

The switching fabric is suited for existing transparent protocols, based on Asynchronous Transfer Mode (ATM) technology and standards in an environment of Local Area Network (LAN).

A high-speed switching fabric architecture which adopts Time Division mode and bases on a shared medium approach is proposed. This is an architecture for non-blocking performance, no congestion and high reliability. Its principle for performance is a method of sequentially scheduling the inputs and the transferring of bits in parallel.

To study the performance of the switching fabric architecture one uses OPNET communication simulation software. Some parameters including the throughputs, the transfer (the switching fabric) delay, the switching overflow and the packet size in the buffer (the input buffer and the output buffer) are implemented through the simulation.

And finally, an analysis for the results of the simulation for local ATM TDS fabric

architecture is discussed. The results display an architecture that provides a rational design with some expected characteristics.

Glossary of Teams and Abbreviations

ATM:	Asynchronous Transfer Mode
B-ISDN:	Broadband-Integrated Services Digital Network
CCITT:	Comite Consultatif International Telegraphique et Telephonique
CLP:	Cell Loss Priority
CSF:	Cell Switching Fabric
FDDI:	Fiber Distributed Data Interface
FIFO:	First In and First Out
FPS:	Fast Packet Switching
FSM:	Finite State Machine
GFC:	Generic Flow Control
HEC:	Header Error Control
IC:	Input Controller
IEEE:	Institute of Electrical and Electronic Engineers
IMs:	Input Modules
ITU:	International Telecommunication Union
LAN:	Local Area Network
LLC:	Local Link Control
MAC:	Media Access Control
NNI:	Network Node Interface
OC:	Output Control
OMs:	Output modules
OMT:	Object-oriented Modelling Technique
OPNET:	Optimised Network Engineering Tools

PI:	Presence Information
PRM:	Protocol Reference Model
PT:	Payload type
PTM:	Packet Transfer Mode
RAM:	Random Access Memory
SNMP:	Simple Network Management Protocol
STD:	State Transition Diagram
STM:	Synchronous Transfer mode
TCP/IP:	Transmission Control Protocol / Internet Protocol
TDM:	Time Division Multiplexing
TDS:	Time Division Switching
UNI:	User-Network Interface
VC:	Virtual Channel
VCC:	Virtual Channel Connection
VCI:	Virtual Channel Identifier
VP:	Virtual Path
VPC:	Virtual Path Connection
VPI:	Virtual Path Identifier

Table of Contents

Chapter 4 Local ATM TDS Fabric Architecture

Chapter 5 Performance of the ATM TDS Fabric

CHAPTER 1

INTRODUCTION

1.1 Background

Most networks are dedicated to specific services, telephony, TV distribution, circuit-switched or packet switched data transfer.

From 1985 to 1990, the **ITU-T** (International Telecommunication Union) Study Group XVIII agreed on a number of Recommendations to describe the basic parameters of **ATM** (Asynchronous transfer mode)[1] [2].

Asynchronous transfer mode is a switching and transmission technique in which all information is carried in short fixed-length packets called cells (53 bytes) [3]. ATM cells mean that simpler and faster hardware techniques can be used, and very high bandwidth links (Gbits/s) can be switched[4].

ATM offers a wide range to suit the diverse requirements of different types of traffic. ATM promises to deliver universal interconnection for all services, namely: voice, point-to-point data, point-to-multipoint data, video, and multimedia, together with simplified management and configuration of networks [5].

Current local area network technologies lack the scalability of performance, and manageability required of resources. The local and campus network can not provide the necessary bandwidth and performance guarantees for application by the hosts and also can not support the increasing user population.

The application of ATM technology and standards, in local area and campus area networking environments will have a significant impact on the future of local and campus networking.

ATM technology, designed for high bandwidth application and large user populations, are a major source of growth in ATM LANs. On the other hand, computer imaging, distributed file, and (video) TV conference requires very high bits-rates. An ATM LAN uses high-speed cell switching and mesh topology technologies.

In the past, various **switches** in networks were developed for different applications such as voice and data, based on transfer modes like **STM** (Synchronous Transfer Mode) [6] and **PS** (packet switching) [7]. The switching techniques range from circuit switching to packet switching. The switching provides an economical means to connect the source and the destination nodes by providing a path through the network. It can be used more and more in the design of the next generation integrated switch.

The flexibility of packet switching to adapt to the form of the ATM for the future B-ISDN has now been accepted. Figure 1.1 shows that the basic differences between them are largely a

result of their different historical backgrounds.

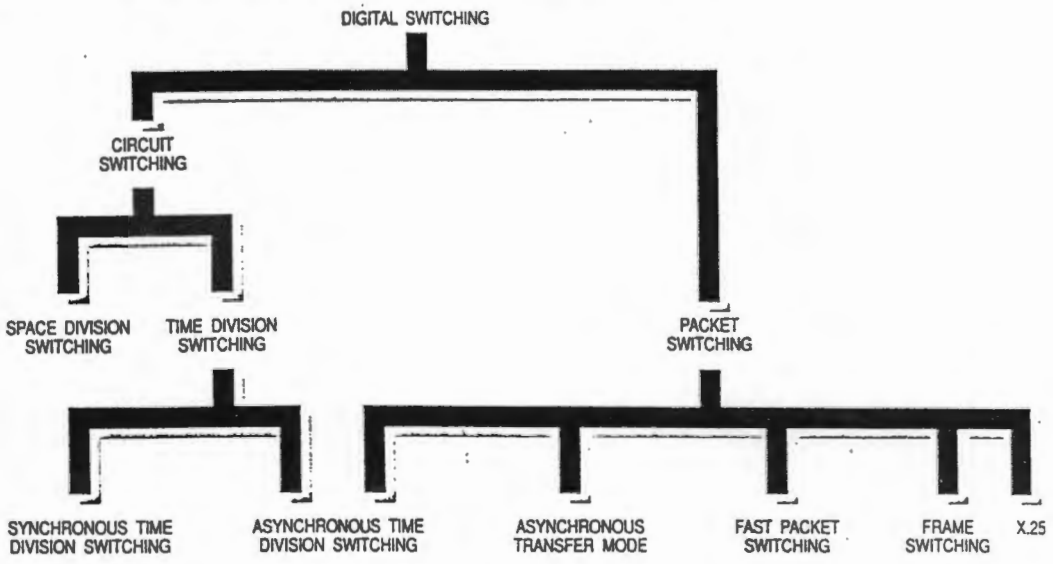


Fig. 1.1 Genealogy digital switching

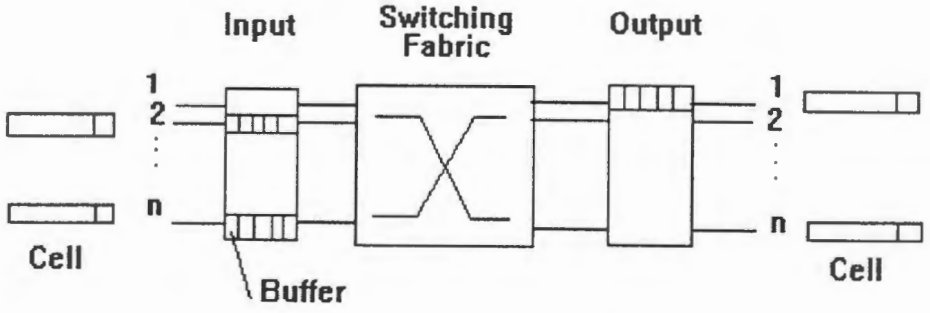


Fig. 1.2 ATM switch Model

Fig. 1.2 shows the model of a general ATM switching apparatus.

The transportation of information from an incoming logical ATM channel to an outgoing logical channel is characterised by:

* A physical input /output, characterised by a physical port number.
* A logical channel on physical port, characterised by a virtual channel identifier (**VCI**) and/or virtual path identifier (**VPI**).[8]

1.2 Solution

With the advances in transmission technology providing the bandwidth required to support **B-ISDN** (Broad-band integrated services digital network) [9] applications, the challenge is now on to design switches that can deal with several hundreds of millions of cells arriving at a node per second and to support different types of connection requirements.

Recently, a large number of architectures for ATM switching fabric have been proposed. The purposes of designing these architectures of switching fabric are just to provide a high reliability [which includes network guarantees of a particular quality of service for the duration of the call (in terms of delay and cell loss probability)], which includes low switching delay, low cell loss probability and has burst buffering; no congestion; and multicast characteristics.

But, it is difficult for an architecture to embrace a wide spectrum of subjects. For a practical switching fabric, there would be a need to solve the key problems for congestion and cell loss probability, or to strengthen reliability in the architecture. On the other hand, it is necessary to reduce the delay and increase the capacity of the switching fabric.

1.3 Introduction

This thesis attempts to explore and research the architecture of

the switching fabric based on Asynchronous transfer mode (ATM) Local area network (LAN) principle, and proposes a simple and high reliability architecture mainly to solve the congestion and cell loss problems in ATM switching.

The Time Division Switching (**TDS**) fabric architecture is one of many ATM switching fabric architectures, which is provided with non-blocking performance, no congestion problem, and a simple routing architecture.

For this reason, a TDS of 16 ports for ATM switching fabric architecture [10] becomes an object of study in this thesis and will be described. And this thesis will also present a simulation study of the performance for the switching fabric. The results of simulation of the switching fabric architecture will be shown. The architecture is essentially based on a shared medium ATM switch architecture. It is a sequential scheduling of the inputs, with parallel-bits transferring through the routing table of the medium, and loading the cell to the output module (in the switching fabric).

Seven chapters addressing aspects of ATM LAN switching fabric are presented. On the basis of chapter 1 for background, chapter 2 introduces the basic concepts and protocols for the technologies of ATM and ATM LAN, so that, we can understand that ATM LAN switching occupies an important position and plays a key role in terms of protocol theories and practical networks.

Chapter 3 gives an analysis of the several typical architectures of ATM switching fabric and switching elements, including the view of the models, the performance and their characteristics. This chapter will also deal with the requirements for designing an ATM switching fabric. The analysis and the requirements for such a design are foundation of this project selecting a basic architecture of ATM LAN switching fabric. Then, an architecture for ATM time division--shared medium switching fabric is to be included and considered in the plan for design of this project.

Chapter 4 discusses a high speed (2.5 Gbits/s) TDS--shared medium switching fabric architecture for ATM LAN in detail. In chapter 5, the simulation for the performance of the TDS fabric will be run on OPNET communicatien simulation software. The results of the simulation will be shown and analyzed in chapter 6. These results further testify the characteristics for the architecture of ATM time division switching fabric. The conclusion of this study will be found in chapter 7.

CHAPTER 2

ATM and ATM Local Area Network

Asynchronous Transfer Mode (ATM) is the switching and multiplexing technique chosen by ITU-T for broadband access to B-ISDN. ATM provides some advantages on flexibility and support for multimedia traffic. It will become a key high-speed network technology which facilitates networked multimedia applications. The technology is growing in application for local area, and campus areas networking environments [11]. The ATM LAN uses standards of ATM and LAN protocols, and high-speed ATM cell switching technology.

2.1 ATM

The ITU-T study group agreed on a number of recommendations to describe the basic parameters of ATM. Refer to Fig. 2.1 for the protocol reference model [12] [13].

ATM is considered the ground on which broadband-integrated services digital network (B-ISDN) is to be built. Within ATM, user information is transmitted between communicating entities using fixed-size packets, referred to as the ATM cells. An ATM cell is 53 bytes, consisting of a 48 bytes information field and a 5 bytes header (See Fig. 2.2 [14]). The primary role of the

header is to identify cells belonging to the same virtual channel within the ATM switch. Transfer capacity is assigned by negotiation and is based on the source requirements and the available capacity.

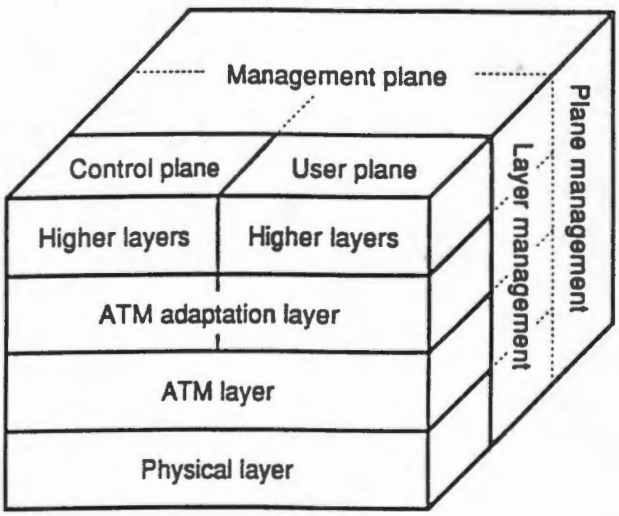


Fig. 2.1 The Protocol Reference Model

The term **"Asynchronous"** [15] refers to the fact that cells allocated to the same connection may exhibit an irregular recurrence pattern, as cells are filled according to the actual demand. (See Fig. 2.3.) This aspect is well-known from the existing packet transfer mode.

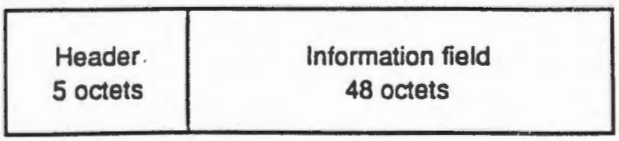


Fig. 2.2 The Cell structure

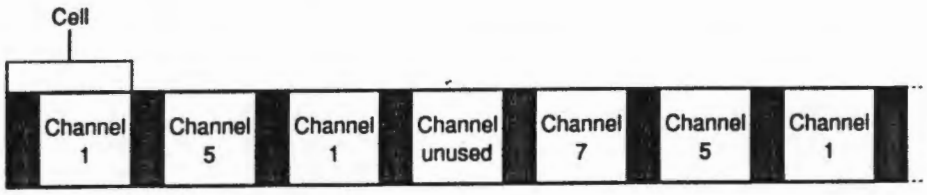


Fig. 2.3 ATM principles

ATM cell transport requires a connection because of the look up tables required by switching and multiplexing units. The connection can be executed at subscription time or dynamically. Connections can be point-to-point or point-to-multipoint.

ATM connections always guarantee delivery of cells in sequence. Two levels of ATM connections are defined by ITU-T: Virtual channel connections (VCC) and Virtual path connections (VPC). The label of the ATM cell header performs a switching and multiplexing function. Header of the ATM cell can see Fig. 2.4 (The cell format for Network Node Interface) [16].

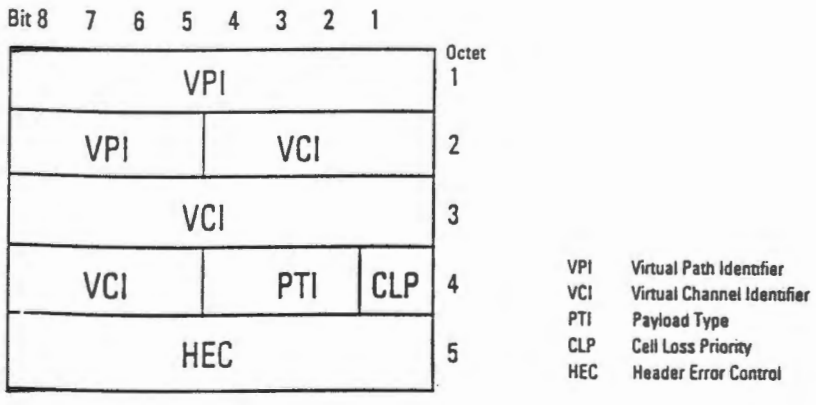


Fig. 2.4 Header of the ATM cell (NNI)

The implementation of these virtual channels and virtual paths

is done by fixed-size cell and provides the basis for both switching and multiplexed transmission. The use of short cells in ATM and the high transfer rates involved result in transfer delays which are sufficiently small to enable universal applicability to a wide range of services including real-time services, voice and video.

2.2 ATM LANs

The new technologies used by the network, include a managed high-speed communications system that supports new classes of enterprise and multi-enterprise applications such as electronic commerce, desktop video conferencing and medical imaging. The advantage of the technologies is that the new network fabric can integrate old network technologies current network technologies (for example Token Ring and FDDI [17]) and new switched LAN technologies (such as ATM and switched Ethernet) quickly, easily and painlessly.

The ATM technology can offer LAN-link service for data traffic and can be compatible with existing data communication protocols , applications and equipment. Most LAN equipment conforms to the IEEE 802 family of the protocol [18].

From Fig. 2.5, we know that the Data link layer is split into the IEEE 802.2 logical link control (LLC) sublayer which offers the interface to the network layer and medium access control (MAC) sublayers, which include Token Ring, Bus, DQDB, FDDI

.... ATM.

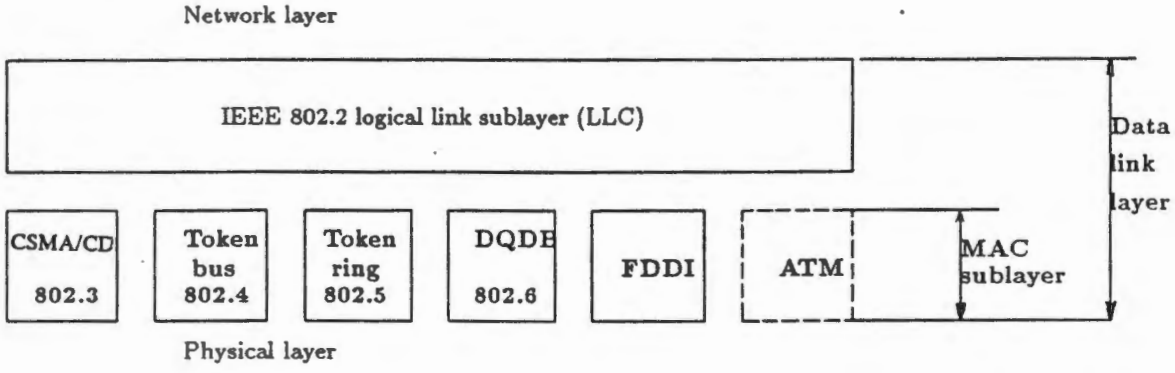


Fig. 2.5 The IEEE 802 family of LAN protocol

Therefore, there are many network layer protocols, and each one would have to be interfaced to ATM separately. Also it to offers general compatibility with the installed base network and protocols, and adapts MAC sublayer's requirements. It will permit existing LAN applications to migrate to the ATM environment without upheaval.

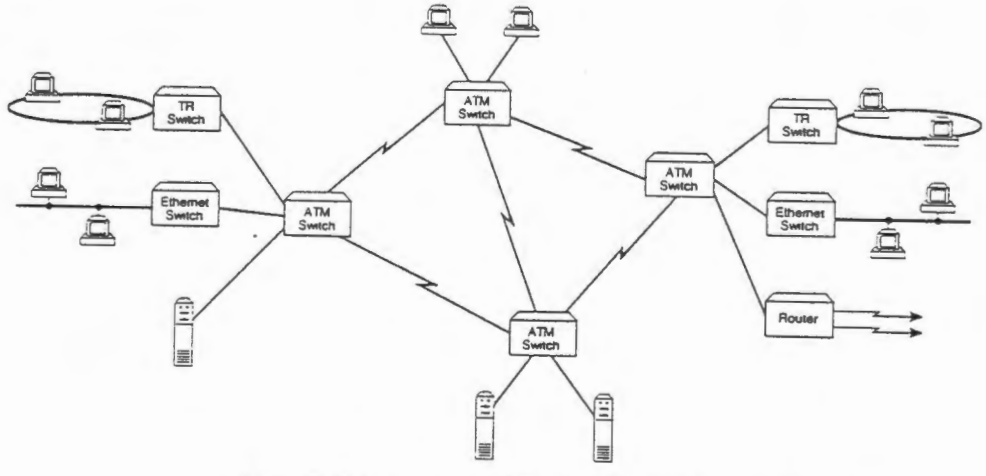


Fig. 2.6 View of ATM LAN environment [19]

A local ATM network can be used to cover a small geographical area (or a large one) with a small number of terminals (or a large number of terminals), by ATM LAN switch. Multimedia workstations can be connected directly to the ATM switches, and

existing LANs of Ethernet, FDDI, Token Ring. (See Fig. 2.6.)

ATM LANs have many advantages over these past solutions and are likely to be commercially viable [20]. ATM LAN are based on international standards and user population; prototypes of switching systems have already been demonstrated; and these networks are emerging at an appropriate moment as high bandwidth applications.

2.3 ATM LAN Switches

An ATM LAN contains two main components: ATM computer adapters and ATM LAN switches. ATM adapters connect workstations and personal computers into the ATM LAN over dedicated pairs of fibre-optic links. ATM switches form the core of ATM LANs and connect multiple workstations, personal computers and mainframes. The multiple switches can be connected together to build enterprise-wide networks [21].

ATM switches are the switching systems and much more than a fabric, which simply routes and buffers cells. There are two main functions in the ATM switching: virtual connection management and cell routing.

The ATM switch transfers cells (information) from multiple input ports and switch them to one or more output ports. This is essentially a packet-switching function, performed on fixed-length cells. The ATM switching is based on VCIs and VPIs.

Chapter 3 will describe in detail the ATM switching technology and some of the typical switching fabric architectures.

CHAPTER 3

ATM Switching

3.1 Introduction

The previous chapter described the current state of the protocols and principles of ATM and ATM LANs, as a preparation for the foundation for ATM switching.

In this chapter, the general requirements and the basic architectural models for ATM switching will be discussed. In addition, a few typical ATM switching fabric architectures will be analysed and discussed. The purpose is to use the characteristics of the architectures and the method of switching as a reference, to help and guide the design of ATM switching fabric.

The ATM switching uses a fast packet switching technique which provides high throughput, better bandwidth efficiency for bursty traffic, and inherent rate adaptation [22].

In an ATM switch, the ATM cells are transported from an input to one or more outputs by translating VPI/VCI of the cell header. The switching from input to output can be combined with concentration, multiplexing and demultiplexing of ATM traffic.

A switching fabric is composed of identical basic switching building blocks, interconnected in a specific topology. In

principle, a switch fabric can be implemented by a single element [1]. (Since such an element could not satisfy the requirements of a normal size ATM switching node, larger switch fabrics are used.)

3.2 Switching Requirements

As asynchronous transfer modes, the broadband network must be capable of transporting all kinds of information, including telecontrol over-voice to high quality video. These services have different requirements in terms of bit rate (kbits/s to Gbits/s) and time transparency (delay) [23]. These requirements can cope with broadband ATM switches and are some of fundamental concepts.

Information Rates:

A large number of information rates must be switched in future broadband switches. When these rates range from a few Kbits/s, up to values of around 155 Mbits/s, the maximum bit rate which ATM switches must be able to switch lies around 155 Mbits/s. In TDS ATM switches several incoming links at 155 Mbits/s can be multiplexed onto a single outgoing link to obtain a higher switching bandwidth (potentially Gbits/s).

Multicast:

ATM switches have the capability to provide a multicast and broadcast functionality. Broadcast can be defined as the

provision of information from one source to all destinations, whereas multicast provides the information from one source to many destinations.

Cell Loss:

The probability of losing a cell must be kept within limits to ensure a high semantic transparency. Typical values for cell loss probability mentioned for ATM switches range between 10^{-8} and 10^{-11} [24].

Some switching fabric architectures are designed so that they will not suffer (at input and /or output) from cells competing for the same resource internally. Therefore these architectures will not lose ATM cells internally, because they can perform as internally non-blocking architecture.

Switching delay:

The time that it takes to switch an ATM cell through a switch is called switching delay. It is also an important factor in the design, research and appraisal of an ATM switching architecture. Typical values mentioned for the delay of ATM switches range between 10 and 1000 μs[25].

3.3 Switching Architecture model

There are some functional blocks in the switching architecture model: input module, output module, and cell switching fabric.

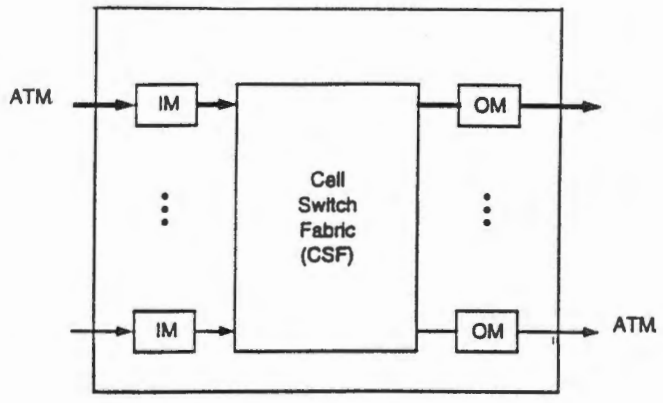


Fig. 3.1 Switching Architecture model

Input Modules (IMs):

The first function of an input module is the termination of an incoming signal. The cells must then be prepared for routing through the cell switching fabric. This requires a number of fields on the cell:

* "Validation and translation of the VPI/VCI values"
* "Possible filtering of signalling cells and to be passed to connection admission control entity."[26]

Output Modules (OMs):

The output modules are considerably simpler than the input modules, because their main responsibility is to prepare ATM cell streams for physical transmission. Specific fields may include:

* Possible mixing of signalling cells from CAC and management cells from OM with outgoing user data cell streams;
* Cell rate decoupling (adding empty cells);
* The payload in the data bus must be appended to the outgoing

VPI/VCI to build the outgoing cell;

* Cell buffering."

Cell Switching Fabric (CSF):

The cell switch is primarily responsible for transferring cells between the other functional blocks in the switch. In particular, user data cells must be routed from the input modules to output modules. Whilst, a switch fabric can be implemented by a single switching element. Since such an element could not satisfy the requirements of a normal size ATM switching node, larger switch fabrics are used.

A number of fields of the fabric should be considered:

* Traffic concentration and multiplexing;
* Multicasting or broadcasting;
* Cell scheduling based on delay priorities;
* Selective cell discarding based on loss priorities;
* Congestion monitoring."

3.4 ATM Switching Architecture

The objective of this section is to analysis and compare several architectures for ATM switching fabric. The purpose is to facilitate the future study of design and performance for switching fabric architectures.

The switching fabric is the central functional block in the ATM switch. ATM switch is often meant to refer just to the

switching fabric, and switch architecture to the design of the fabric.

3.4.1 Requirements

The preceding chapter has defined the ATM switching fabric as more than a simple fabric. These basic functions are handled by a routing and buffering functional block within the switching fabric. The architecture for ATM switching fabric attempts to transmit cells without any cell loss occurring, to minimise cell delay and transfer the cells at high speed.

Therefore, designing an architecture of ATM switching fabric, the following are some factors to be considered: congestion, cell loss, cell delay, and multiplexing.

Chapter 1 referred to the problems which should be solved in the design of an ATM switching fabric architecture. The objectives of this switching fabric will focus on low probability of cell loss with high speed (around 2.5 Gbits/s) processing.

3.4.2 The Switching Fabric Architectures

This section deals with the general classification of switching fabric architectures. The switching fabric is the central functional block in the ATM switch.

At present, research in switch design has been widespread,

and many commercial ATM switches have been developed based on ideas from early research prototypes. A few switching fabric approaches are classified as [24][27][28]:

* **Shared memory;**
* **Shared medium;**
* **Space division;**
* **Fully interconnected.**

Actually, all current switch designs are based on variations or combination of these approaches. The shared medium and the space division will be described in detail in the following pages. The detail architecture and characteristics for Shared memory and Fully interconnected are described in reference [26].

Shared Medium

For shared medium approach, cells may be routed by means of a shared medium, ring, bus, [29] or dual bus. An example of a fabric using a **Time Division** (switching) multiplexed (TDM) [30] bus is shown in Fig. 3.2.

The principle of its implementation is simple. Incoming cells from the input ports in parallel are sequentially broadcast on the bus in a round-robin fashion. The address filters examine the internal routing tag on each cell to determine if the cell is destined for that output. The address filter passes the

appropriate cells through to the output buffers.

The time division switching fabric can be thought of as a "traffic circle", in which the traffic from each street (or input/output port) is entering. The traffic circle (switch) is controlled by a traffic light (or time slot). The traffic is only allowed to pass into the traffic circle when the light is green.

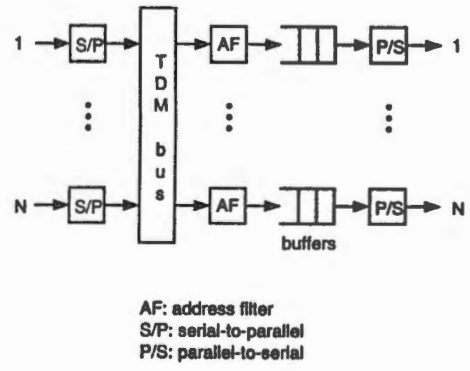


Fig. 3.2 The Shared medium approach

This switching fabric is often referred to as being non-blocking. This approach involves output queuing. In this architecture, the fabric capacity is greater than or equal to the sum of the individual input capacities, therefore no congestion is within the switching fabric. In addition, because the architecture is a single stage switch, its minimal delay is able to predict.

Space Division

In a Space Division fabric(or called **Multistage Matrix Switching**

fabric), concurrent paths can be established between input and output ports, so that many cells may simultaneously be transmitted across the switching fabric. Banyan fabric is a common example of multistage matrix switching (See Fig. 3.3).

Banyan fabric provides an important function of self-routing. The output port of an ATM cell is determined by its virtual channel and virtual path identifiers. This output port is specified by a routing header, which is appended to the start of the cell giving the output port address in binary.

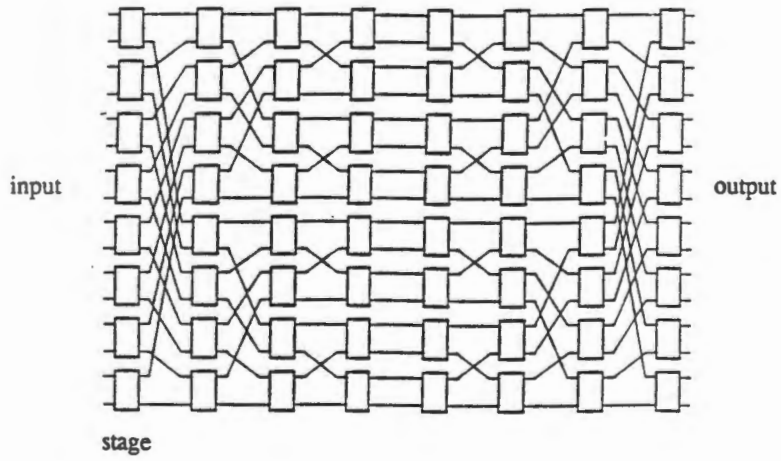


Fig. 3.3 Example of a Banyan network

The purpose of Banyan fabric is to achieve high speed switching and high throughput in hardware. The switching is performed by small, simple switching elements, in which cells are routed in parallel; all elements operate at the same speed; and larger fabrics can be constructed modularly and recursively.

On the other hand, a multistage matrix fabric is analogous to a city street grid in which a number of traffic intersection (2x2 switching elements) need to be traversed to reach the end port destination. Similar to a city street grid, some chance of delay, collision, and traffic congestion at each intersection does exist.

3.5 Summary

The two design approaches have been described in the preceding section 3.4. The two major types of switching fabric architectures exist in today's local ATM switch products: **time division fabrics** and **space division fabrics** (**multistage matrix switching**). Each of these architectures has its relative advantages and disadvantages.

For the shared medium approach of the switching fabric, the time division method is used. This is a simple architecture and satisfies the requirements of ATM switching (no congestion and no cell loss). Therefore, the shared medium architecture is most commonly used in time division switching for ATM switching fabric design.

The architecture of the TDS fabric is a **non-blocking**. All input/output ports can transmit at their highest rate without any chance of blocking or congestion within the switching fabric. For a 16 ports switch, each port occupies a time slot of

1/16 of the rate in the fabric. (If the rate is 2.5 Gbit/sec. in the fabric, each port would get a time slot of 155 Mbits/s.)

The capacity is greater than or equal to the sum of the individual input/output ports, so that there is no congestion at the port level. This type of switching is a single stage switch with minimal delay associated with cell transit, therefore its delay is very low.

CHAPTER 4

Local ATM TDS Fabric Architecture

In chapter 3, a few typical switching fabric architectures were analysed and discussed. This chapter will describe in detail an architecture and the principle of the performance for ATM time division switching (TDS) fabric. The plan for the switching fabric architecture will be described in three parts: input module, routing fabric and output module [31].

4.1 The Architecture Frame Work

Chapters 1, 2, and 3 provided the background and the requirements of the design for ATM LANs and architectures for the switching fabric, so that the foundation for a design of the switching fabric can be supplied.

The design has also considered when the cells cross the switching fabric and how the TDS switching fabric can manage the traffic.

In this 16 ports switch, the shared medium approach for ATM Switching fabric adopts the time division multiplexing technique. The aggregate capacity of the TDS fabric is 2.5 Gbits/sec.

According to the functions for the ATM switching fabric, this switching fabric will be divided into three major modules:

* Input module
* Routing fabric
* Output module.

The input module accepts the incoming cells and sends them to the routing fabric. The module schedules the incoming cells.

The routing fabric translates the VPIs/VCIs of these cells.

The output module reconstructs the outgoing cells. The capacity is the same as the routing fabric.

A centralised clock synchronises the exchange between these modules.

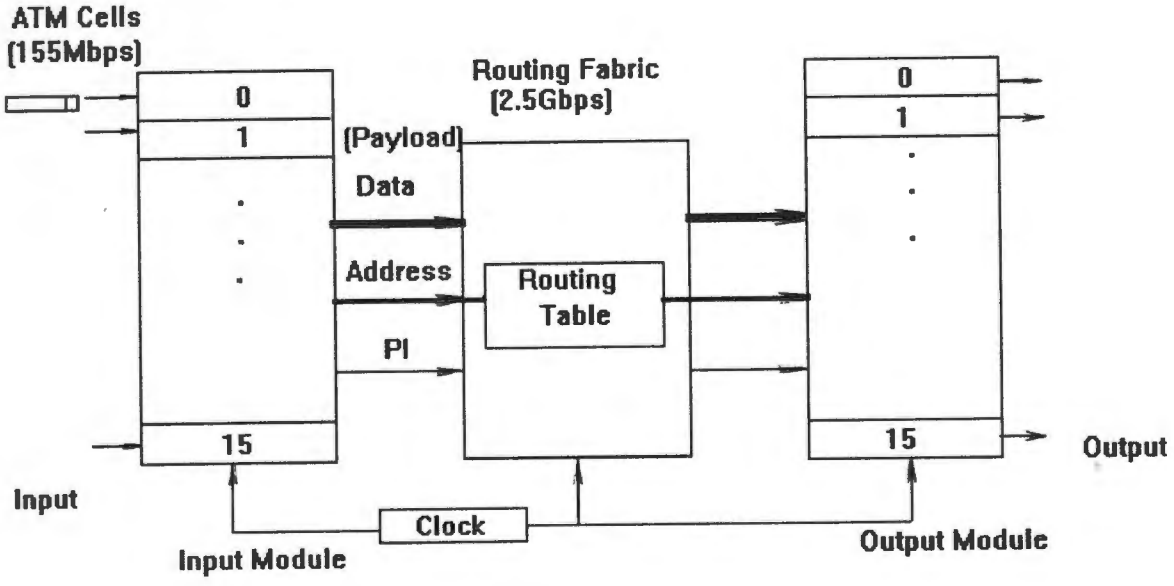


Fig. 4.1 The structure of the TDS switching model

4.2 Input Module

The TDS fabric requires that incoming cells are sequentially broadcast on the bus in a round-robin fashion. The input module has to manage the incoming cells from the 16 ports.

In order to guarantee a short transfer time, the pattern of the transmission for each cell (424 bits) is converted from a serial to a parallel bits stream in the input module (see Fig. 4.2.).

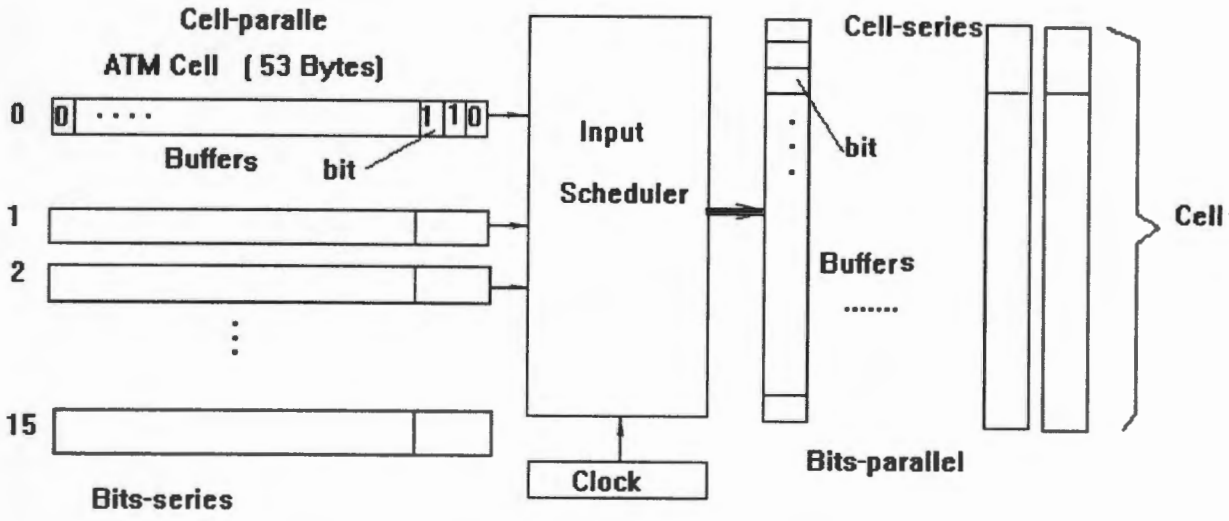


Fig. 4.2 The cells change from parallel to series

4.2.1 Input Scheduler

The input scheduling module is an important part of the time division switch. When the input buffers are filled with the incoming cells (each cell is 424 bits), the scheduler will provide a **time slot** for a predetermined time period to multiplex the 16 buffers onto the routing fabric.

The capacity of TDS switching fabric must be equal or greater than the sum of the capacities of all input links.

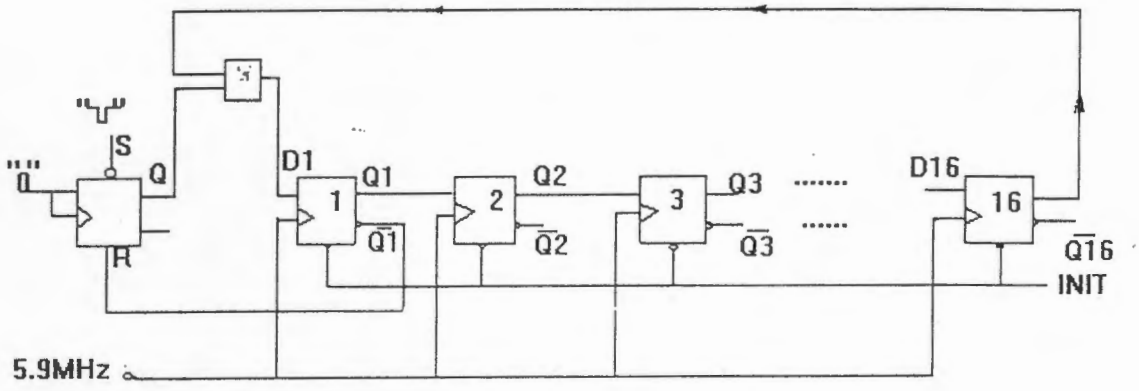


Fig. 4.3 Input scheduler

Referring to Fig.4.4, each input port is allocated a time slot (**Ti**) of 2.74 μs.

$$Ti = \frac{L}{V} = \frac{424 \text{ bits}}{155 \text{ Mbits/s}} = 2.74\,\mu s$$

(* Where **L** expresses the length of a cell, in bits.)

(* Where **V** expresses the bit rate of a cell, in bits/sec.)

Then the transfer time of each port (**ti**) is 170*ns*.

$$ti = \frac{Ti}{N} = \frac{2.74\ \mu s}{16 \text{ ports}} = 170\ ns$$

(* Where N is the number of N the port.)

The scheduler multiplex for 16 input buffers, the multiplex frequency (**f**) is 5.9 MHZ for each buffer (Fig. 4.4.).

$$f = \frac{1}{ti} = \frac{1}{170\ ns} = 5.9\ \text{MHZ}$$

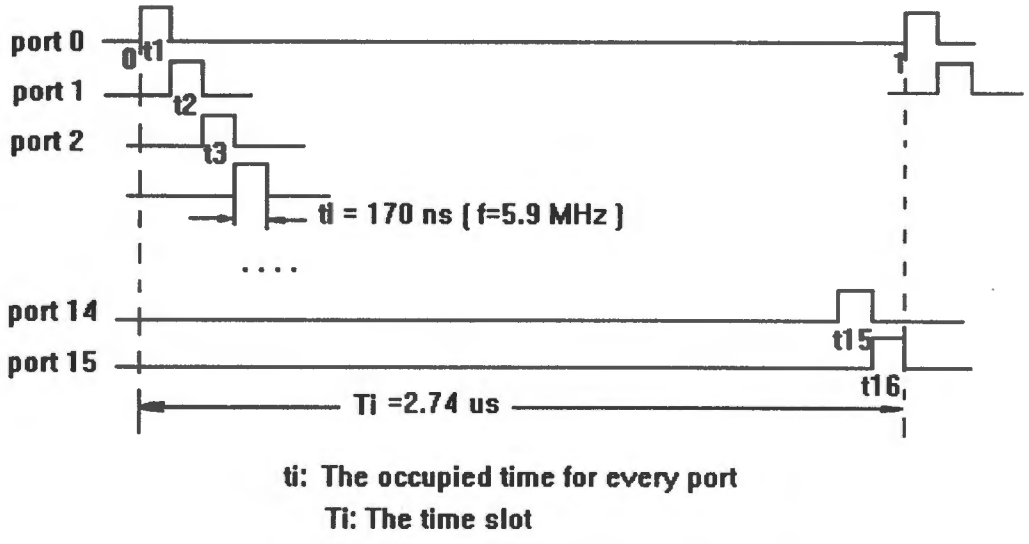


Fig. 4.4 Time scheme of the scheduler

4.2.2 Input Queue Module

In the input queue module, there are two registers associated with each input port, Register 1 (R1) and Register 2 (R2). Each buffer stores a 53 bytes (424 bits) cell plus 1 bit (PI) indicating whether the buffer is empty or full (For PI=1, the buffer is full).

The R1 buffer is to accept the incoming cells from only one port. It must be able to accept the incoming cells as a serial stream, and when the bit PI=1, it send those bits in a parallel stream to R2 buffer.

The R2 buffer forwards the cell to the routing fabric every 2.74 μsec. From the R2 buffer, a cell (424 bits) and PI (1 bit) will be sent to the **Data** bus (384 bits in the cell payload), the **Address** bus (40 bits in the header of the cell) and the **PI** (1

bit) to the routing fabric. The R2 buffer will be emptied within 170 *ns* and carried to the output buffer module.

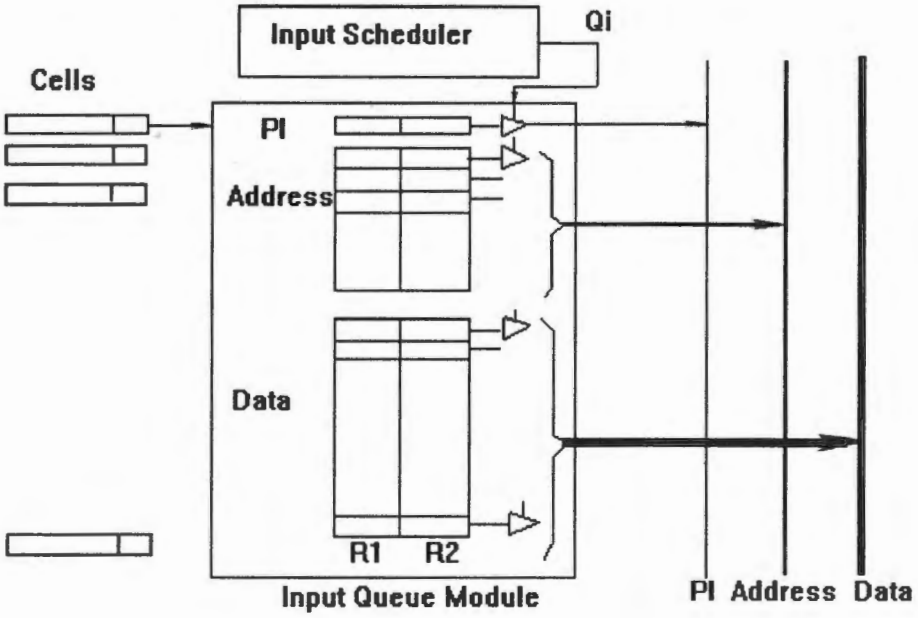


Fig. 4.5 Input Queue Module

4.3 Routing Fabric

The routing fabric is the core of the ATM switch. It is responsible for the transport of cells within the switch. The functions are to translate VCI/VPI, and routing the cells to the appropriate output port (ports), as quickly and efficiently as possible.

In the routing fabric, a table addressed by the incoming header of the cell (40 bits), translates the VCI/VPI and routes the cell to its output port (or ports).

Table 4.1 shows how cells are routed. If a cell arrives at input

port number 0, and the content of the address (VCI/VPI) is 00010010, the table will guide the cell to output ports number 2, and allocates a new VPI/VCI 00000010, while retaining the same cell content. The input port number 1 has an incoming cell, and its VCI/VPI is 00010000. The routing table will guide it to the output port number 0, with a new VPI/VCI of 00000000.

Input port number	VCI/VPI	Output port number	VCI/VPI
0	00010010	2	00000010
1	00010000	0	00000000
2	00011111	15	00001111
3	00100100	0, 4	00011101
:	:	:	:

Table 4.1 A example for routing table

A suitable hardware which performs this work will have to be found. According to the routing functions and access speeds of the routing fabric, RAM (random access memory) will be selected to do it.

The random access memory (RAM) [33] [34] can be used as a routing table for translating and guiding the address (VPI /VCI) of the cells in this design.

Fig. 4.6 shows the processing of the cell header in the switching fabric. In figure 4.6 a routing table element is shown (refer to Appendix C).

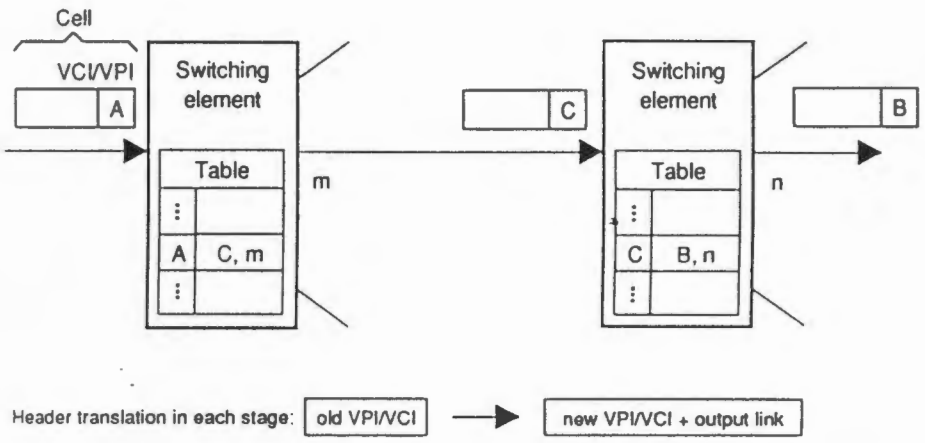


Fig. 4.6 The Table Routing element for the head of the cell

In Fig. 4.7, 40 address signals enter the routing fabric (RAM), and there should be 40 other address signals going out from the routing fabric to the output buffers, as well as a 16 bit tag identifying the output module for this cell.

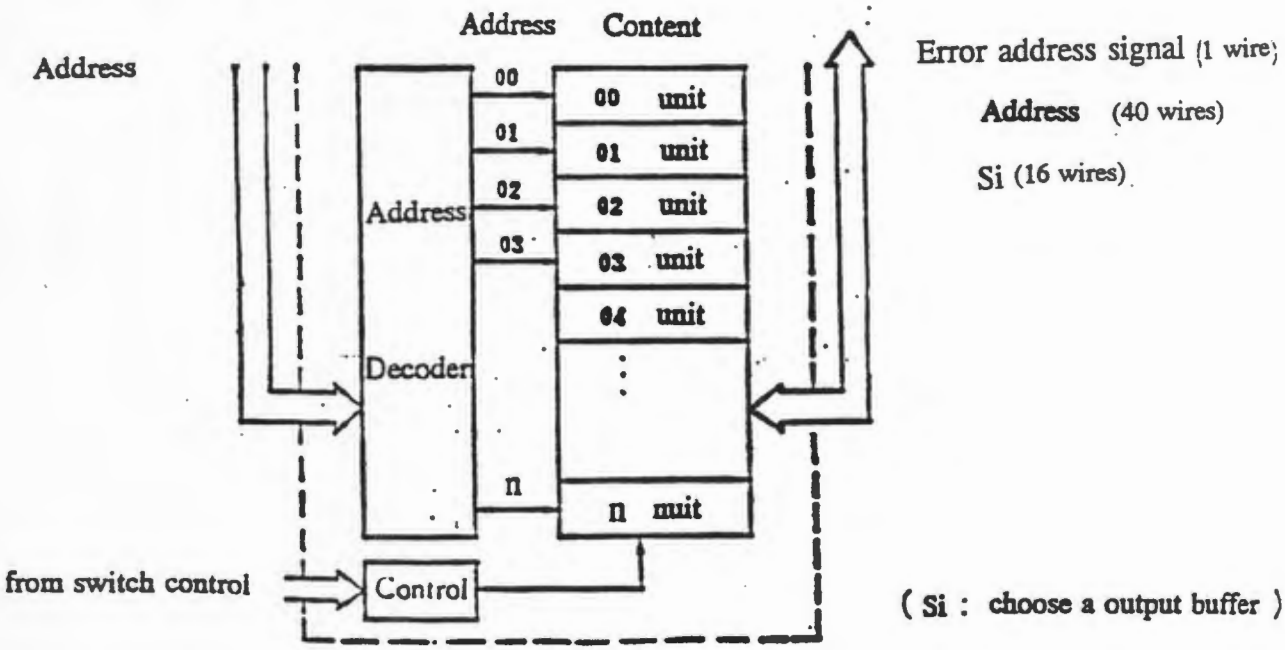


Fig. 4.7 Routing Table structure for head of the cell

The header of a cell passes through the routing fabric (RAM), Towards the output buffers in the output module.

4.4 Output Module

The routing table and output buffers are straightforward in their implementation. In the output module, the output buffers collect the parallel bit stream at a speed equal to N times that of the port speed.

When the output buffer(s) receives a signal Si=1, this (these) output buffer(s) open and accept the bits. The buffers can be loaded with the **Address Bus** (40 bits) which is passed through the routing fabric, the **Data Bus** (384 bits), and the single PI bit. The transfer time from the incoming module to the loading into the buffer of the input module is 170 *ns*.

If cells from different input ports are scheduled for the same output port, the output module will place cells into buffers that are associated with the same output port (refer to Fig.4.8).

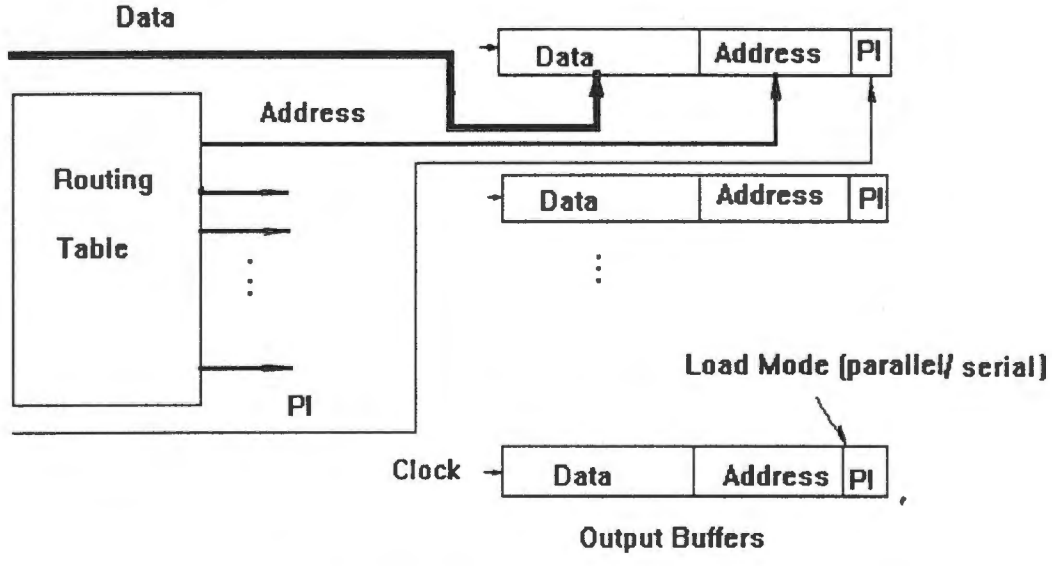


Fig. 4.8 The Output Module

In chapter 6, where the simulation results and analysed, the amount of storing cells for each output port will be presented.

4.5 Feature

In the above sections, the architecture for ATM Time Division Switching fabric was described in detail.

The bandwidth of the switching is high, due to the single stage routing. (The result for the transfer time will be shown in Chapter 5.)

This architecture is able to avoid the congestion, collision and blocking in the switching fabric.

CHAPTER 5

Performance of the ATM TDS Fabric

5.1 Introduction

Chapter 4 described Time Division Switching fabric architecture and its characteristics. This chapter will focus on the performance of the ATM TDS fabric architecture. The performance of this architecture will be studied using the **OPNET**, which is a communication network simulation software tool.

OPNET (Optimised Network Engineering Tools) is a communication simulation package consisting of three models[36]: Network, Node and Process, to describe the target system. The package provides a language called Proto-C which processes models depicting state-transitions in a graphic format to support visualisation of top level control flow. OPNET also includes ATM model description and ATM model simulation. These functions enable OPNET to simulate a wide range of communication systems.

OPNET is also a system of inter-related programs, libraries, data, file and simulation models, and it is invoked from a UNIX shell. OPNET provides a comprehensive development environment for the modelling and performance evaluation of communication networks. System behaviour and performance are analysed by

performing discrete event simulations.

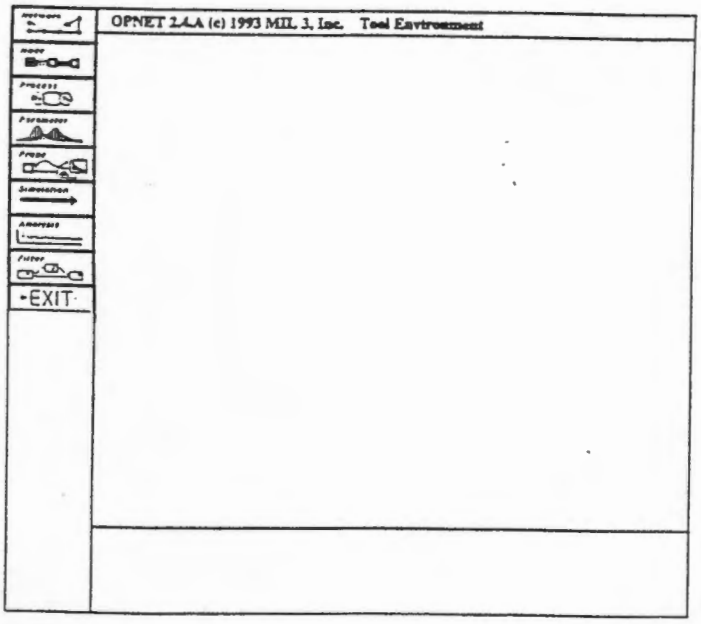


Fig. 5.1 Tool Window of OPNET

OPNET provides seven tools for creating models: Network, Node, Process, Parameter, Probe, Simulation and Analysis Tool Windows, these define parameters controlling simulation, and can analyse simulation results. They each have the single purpose of defining models at one level of the modelling hierarchy.

In OPNET, the Simulation Kernel Manual (available in the Library) serves as a reference for the services provided by the simulation kernel. The simulation services are accessed through Kernel Procedures (KP's) [37], which can be called up from within process models, and C functions which have been scheduled as interrupts.

Using OPNET simulation, the model selected will be involved in

the implementation of the character of the ATM switch.(ATM Model Suite Entity [38] is briefly described.) But it is difficult to establish the modules, because of the complexity of the ATM switching fabric. The modelling includes the structure model, and the model must be matched with OPNET modules. Experience has shown that TDS fabric model is divided into several big parts (or modules), linking each other up to make a **Network**. On the basis of the function of every module, nodes are set-up in **Node Tool Window,** then its dynamic model for ATM switch is used for Proto-c language to edit in OMT (Object-oriented Modelling Technique), thus directly mapping the object-oriented dynamic model into the OPNET **Process Model**.

5.2 Performance

5.2.1 Briefing

In OPNET, several examples of simulation on the communication network are cited in the OPNET Tutorial Manual [39], and there is an ATM switch ready-mode model in the ATM Model Description of OPNET Example Models Manual as well. Of course, they only promote use for users are if you make up your own model, you will need to utilise them with efficiency.

ATM switching fabric architecture is a complicated model. It is necessary to establish the frame (the structure model) [40] of TDS fabric architecture on the Network tool window first, so that it is easy to understand the performance of the architect-

ure and to further establish the sub-structures. Then, the Node and the Process Modules in the Network edition will be created. Fig. 5.2 displays the frame of operation process for simulation. During the processing of simulation, it can be consulted in the same way as **Packet Switching** simulation in the Tutorial Manual for OPNET.

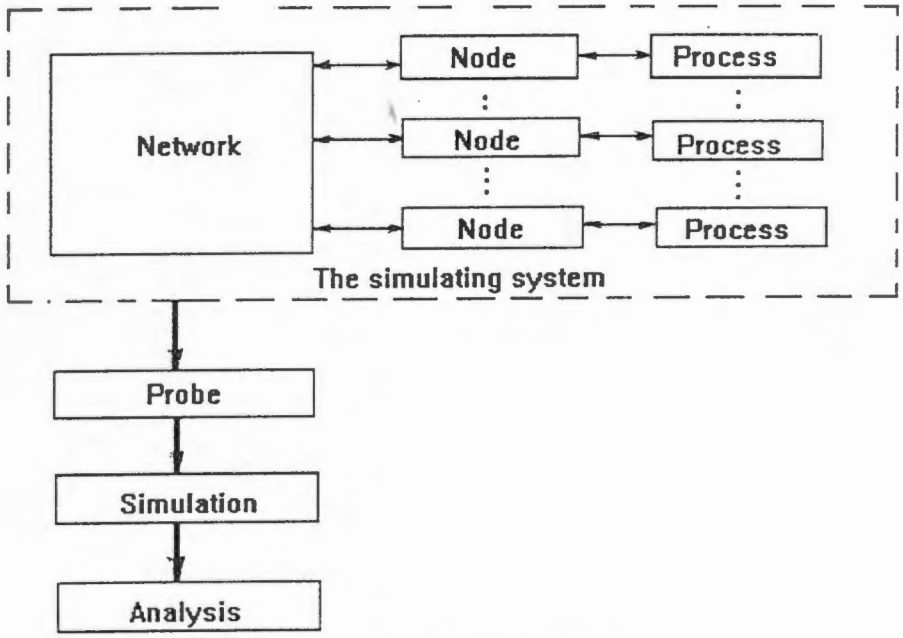


Fig. 5.2 The Frame of operation process

The switching fabric consists of some **nodes**, which are connected by link (the streams). Cells travel from the input node to the proper output(s) node(s) through the node for switching fabric. These nodes will perform the simulation in a couple of the processing programs (Proto-C)of their **processes**.

5.2.2 Establishing the Network of the TDS Fabric

The Network editor is used to specify the topology of a network model in terms of the objects that it contains and the relation-

ships between them.

The modelling of a local ATM Time Division Switching fabric has been described in chapter 4. The architecture consists of three modules (Input module, routing fabric module, and output module) . Therefore, according to the functions and the architecture of the three modules, the topological network will be created in Network Editor of OPNET.

Subnetwork:

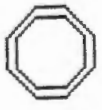
First, the Subnetwork is set-up, which is a container for additional network objects, and including other subnetwork. As the system become more complex, it is sometimes useful to view a group of nodes within the network as an entity. Such a grouping of nodes, along with the links connecting them, is called a subnetwork.

Subnetwork

This subnetwork includes the three modules: input module, routing fabric module and output module.

Input Module:

In the input module, if each input port is indicated by  , then the 16 input ports are in parallel.

The input module will arrange the incoming cells in parallel to change into cells in series. This function use another  to

express, that these cells queue in the buffers to wait to go through the switching fabric. The module is shown on Fig. 5.3.

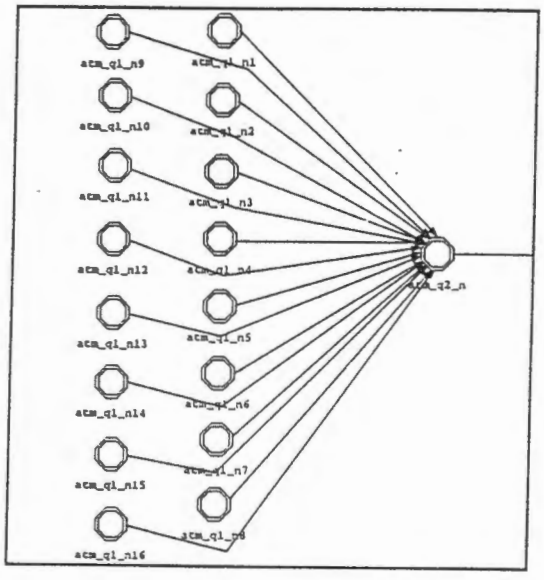
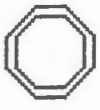


Fig. 5.3 Input Module in the Network

Each  node can provide itself Attributes in the OPNET. The node atm_q1_n1 is shown on Table 5.1 for example.

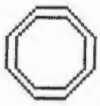
fixed node **atmtd16_IO_n.atm_q1_n1**			
attribute	*value*	*type*	*default value*
name	atm_q1_n1	string	f
model	aq1b_node0	enumerated	NONE
x position	-76.7199760256757	double	0.0
y position	45.3907737519817	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

Table 5.1 The **Attribute** for **atm_q1_n1**

Routing fabric Module:

Naturally, the routing fabric can be thought of as a medium, so it will be shown as a  in the middle, and connects with input module and output module by the links.

Output Module:

The output module has 16 output ports. And it uses 16  to express and connect with routing fabric module. It should cover the function of the output module.

TDS Fabric Frame:

Then, the point-to-point links connect these nodes organically. The TDS architecture in the Network Model is shown Fig. 5.4.

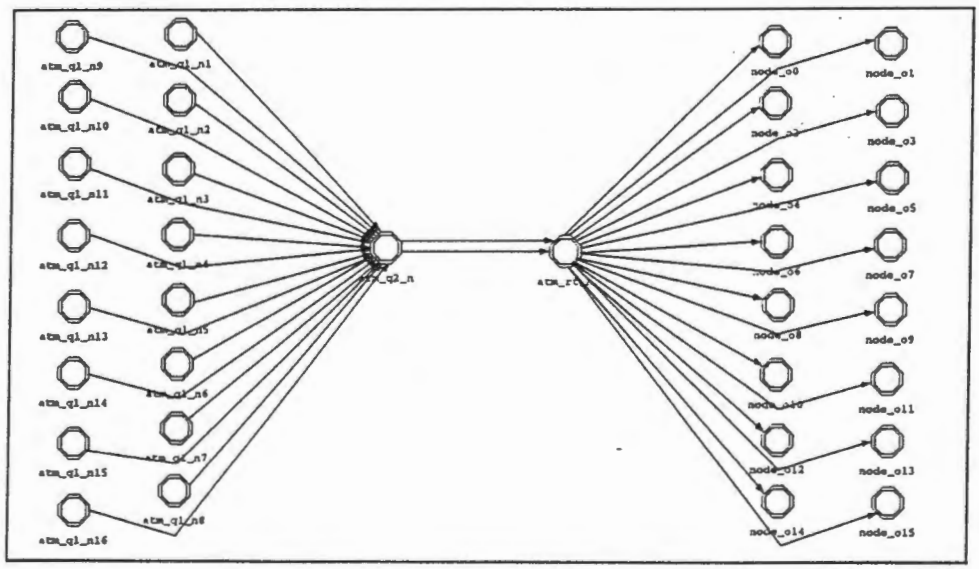


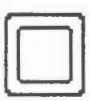
Fig. 5.4 TDS fabric architecture in **Network Tool Window**

In OPNET, the Network model is translated into a C program to execute the simulation which can contain the analysing behaviour and the performance of the (system) architecture. The model consists of some nodes which are interconnected by links, so next section will focus on creating Node and Process modules. The report of the **Attribute** for the **Network Model** will be shown in the Appendix.

5.2.3 Creating Nodes and Process Modules

5.2.3.1 Introduction

The **Node Editor** is used to specify the structure of device models that can be substantiated as nodes. Nodes are composed of several different types of objects. The following types of objects are used to construct node models:

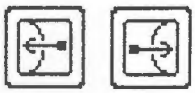


Processor: a general purpose, highly programmable object whose behaviour is specified by a Proto-C process model developed in the **Process Editor**. Processors are used to model such diverse hardware and software functions as protocol units.

Queue: a fully programmable object with built-in support for developing models of queuing disciplines. A Proto-C

process model, embedded within the queue, can insert packets into and retrieve packets.

Ideal Generator: a simple traffic source used to generate packets in a pattern specified by probability distributions. It is used for the time separating consecutive packets and for the size of the generated packets.



Point-to-Point Receiver and **Point-to-Point Transmitter**: incoming and outgoing interface from a point-to-point link to a node, and from a node to a point-to-point link. This model receives (and accepts) packets transmitted by the remote node attached to a point-to-point link, (from other modules in the local node, and forwards them to other modules in the local node (and forwards them to the remote node attached to a point-to-point link.).

Packet Stream: this object is a unidirectional connection capable of transferring packets between two modules in the same node.

When creating the Processor  , a couple of types of objects are concerned. Proto-C process model needs to develop in the Process Editor, so the Process Editor is a very important step, and it cannot be separated from the creation of the node module.

The Process Editor is the environment in which Proto-C specifications are developed. Proto-C is a mixed graphic and textual language specifically designed to support model development for interrupt-driven processes such as communication protocols and distributed algorithms. Proto-C process models include a Finite State Machine (**FSM**) diagram with embedded C statements, and various blocks containing codes for variable declaration, macros ,constants, and function definitions. The FSM diagram represents the functional flow of the process with an easily interpreted diagram of States and transitions (**STD**).

Proto-C process description consists of a graphic state transition diagram and the blocks of code embedded within its states. Several additional data and code blocks are incorporated into Proto-C models to support the activities specified in the main body of the STD.

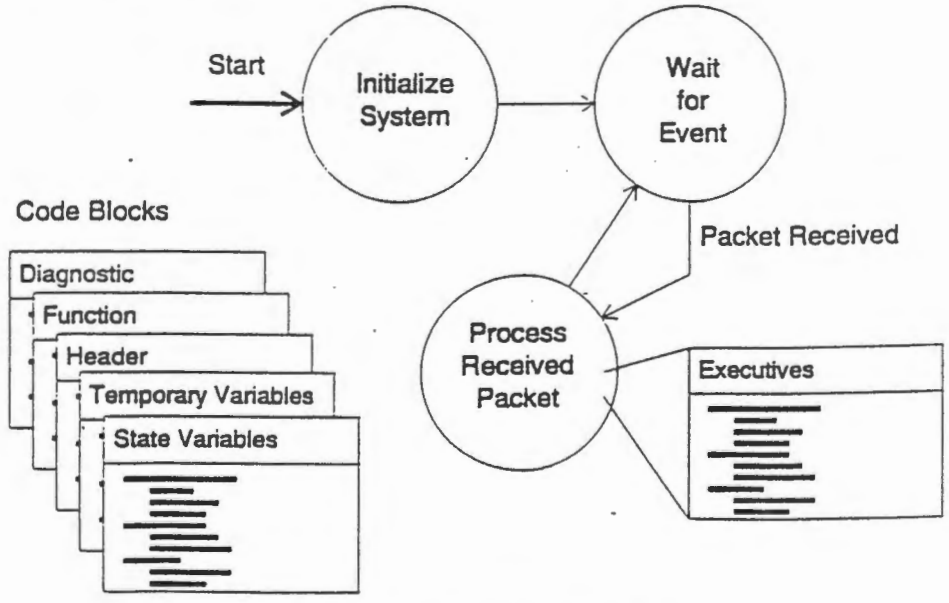


Fig. 5.5 An Example FSM

They are State Variable Block **(SV)**; Temporary Variable Block **(TV)**;

Header Block (HB); Function Block (FB); Diagnostic Block (DB); Termination Block (TB); Process Model Attributes; Child Process List.

5.2.3.2 Input module:

1) The source for ATM cell:

A probable source for ATM cells is similar to cells that come from other ATM switches, multi-stations or users. A convenient source for simulation ( ideal generator) will be added into the input modules.

Therefore, creating a Packet Format for ATM cells using the Parameter Editor, the packet format is one of the parameter model types. A packet format consists of a list of field definitions each with the following properties: "name, type, size, and default value." For example, the ATM cell is defined as the packet format for the ideal generator in the Packet Format Data Table. (To see Table 5.2)

Packet Format Report: **acellp_pk**	Tue Apr 21 21:55:10 1998	Page 1 of 2
...		
...		

Field 0	
Component Name:	*Component Value:*
Field Name	GFC
Type	integer
Size (bits)	4
Default Value	0
Default Set	set

Field 1	
Component Name:	*Component Value:*
Field Name	VPI
Type	integer
Size (bits)	8
Default Value	-1
Default Set	set

Field 2	
Component Name:	*Component Value:*
Field Name	VCI
Type	integer
Size (bits)	16
Default Value	-1
Default Set	set

Field 3	
Component Name:	*Component Value:*
Field Name	PT
Type	integer
Size (bits)	3
Default Value	0
Default Set	set

Field 4	
Component Name:	*Component Value:*
Field Name	CLP
Type	integer
Size (bits)	1
Default Value	0
Default Set	set

Field 5	
Component Name:	*Component Value:*
Field Name	HEC
Type	information
Size (bits)	8
Default Value	
Default Set	set

Field 6	
Component Name:	*Component Value:*
Field Name	payload
Type	packet
Size (bits)	0
Default Value	
Default Set	unset

Field 7	
Component Name:	*Component Value:*
Field Name	payload_bits
Type	information
Size (bits)	384
Default Value	
Default Set	set

Field 8	
Component Name:	*Component Value:*
Field Name	port
Type	integer
Size (bits)	0
Default Value	
Default Set	unset

Field 9	
Component Name:	*Component Value:*
Field Name	PI
Type	integer
Size (bits)	1
Default Value	
Default Set	set

Table 5.2 The **packet Format** for **ams_atm_cell**

2) Setup for a model for the loading of ATM cell

The model process for the loading of an ATM cell and setting-

up of parameters for the head of ATM cell can be based on the mathematics model (Distribution Packet) [41].

According to the function of "set" node, the process model is shown in Fig. 5.6.

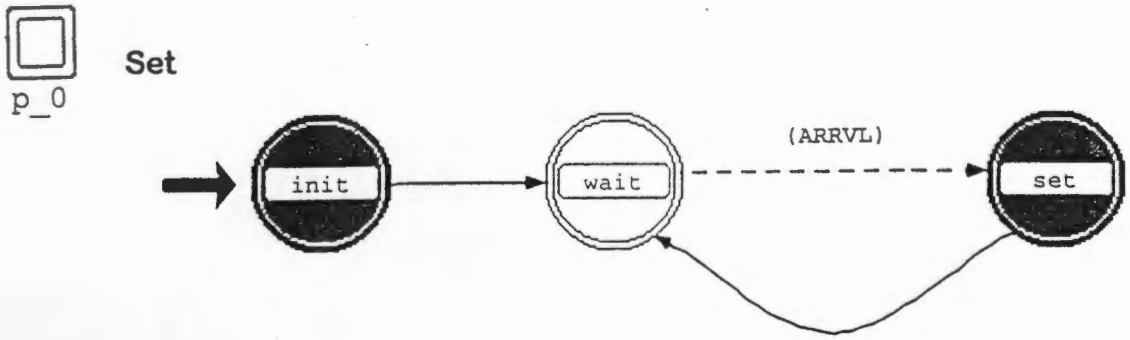


Fig. 5.6 The process model for **Set**

For this model, one possible implementation employs three states:

* Initial is the start of the FSM where the packet (cell) is initialised and where the FSM responds to setting up the mathematics model for loading cells.

* Wait, where the FSM waits for a cell arrival;

* Set, where the FSM responds to setting up the address: the VCI and the VPI, and a constant: PI. For different input ports, the different value (number) of VCI and VPI can be setup. For example:

  Input number #1: VCI = 1, VPI = 1, PI = 1;

  Input number #5: VCI = 5, VPI = 1, PI = 1;

  Input number #14: VCI = 1, VPI = 2, PI = 1;

  Input number #10: VCI = 0, VPI = 1, PI = 1;

....etc.

The Proto-C from **Simulation Kernel** (in Simulation Kernel Manual) will be written into the **Executives** of the Initial and the Set. The Simulation Kernel is event driven; it maintains a schedule of events which it uses to determine when process models should be invoked. The Simulation Kernel delivers an interrupt to process and passes control to the process.

In other words , the process model is the dynamic model, which uses Proto-C language to describe the processing of cells moving in the switching fabric. Table 5.4 shows the program of set process model.

Process Model Report: **atm_setp**	Thu Apr 30 09:21:57 1998	Page 3 of 4
...		
...		

Header Block

```
#define ARRVL (op_intrpt_type () == OPC_INTRPT_STRM)
#define IN_STRM 0
#define OUT_STRM 0
```

State Variable Block

```
Distribution* \address_dist;
Distribution* \PI_dist;
```

Temporary Variable Block

```
Packet* acellp_pk;
double  route_delay;
int     VCI, VPI, PI;
int     output_port;
int     temp;
```

forced state init

attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs init

```
address_dist = op_dist_load ("bernoulli",0,3);
PI_dist = op_dist_load ("constant", 0,1);
```

transition init -> wait

attribute	*value*	*type*	*default value*
name	tr_26	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state set

attribute	*value*	*type*	*default value*
name	set	string	st
enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs set

```
acellp_pk = op_pk_get (op_intrpt_strm ());

temp=op_dist_outcome (address_dist);
/*printf("%d.\n", temp);  */
op25_pk_nfd_set (acellp_pk, "VCI", temp);
op25_pk_nfd_set (acellp_pk, "VPI", temp);
op25_pk_nfd_set (acellp_pk, "PI", 1);
op_pk_send(acellp_pk, OUT_STRM);
/*printf("Going with VC and VP =  %d.\n", temp);
*/
```

transition set -> wait

attribute	*value*	*type*	*default value*
name	tr_28	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

unforced state wait			
attribute	*value*	*type*	*default value*
name	wait	string	st
enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

enter execs wait	

transition wait -> set			
attribute	*value*	*type*	*default value*
name	tr_27	string	tr
condition	ARRVL	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

Table 5.3 The Set Process Model Report

3) Queue

The queue becomes the input buffer in practical performance. It can store the incoming cells, and can change into a cell in bits-parallel from bits-series.

In the process model for the queue model, we select the Active, Concentrating, acb_fifo_ms **queue** process model [42]. It accepts packets (ATM cells) from any number of sources and autonomously forwards them to a single destination module. Enqueued packets wait for the completion of service of early arriving packets before they themselves may begin service and eventually be

forwarded. The atm_acbf_ms (Multi-service) has a number of concurrent servers supported by the queue,and is configurable via the process model attribute.

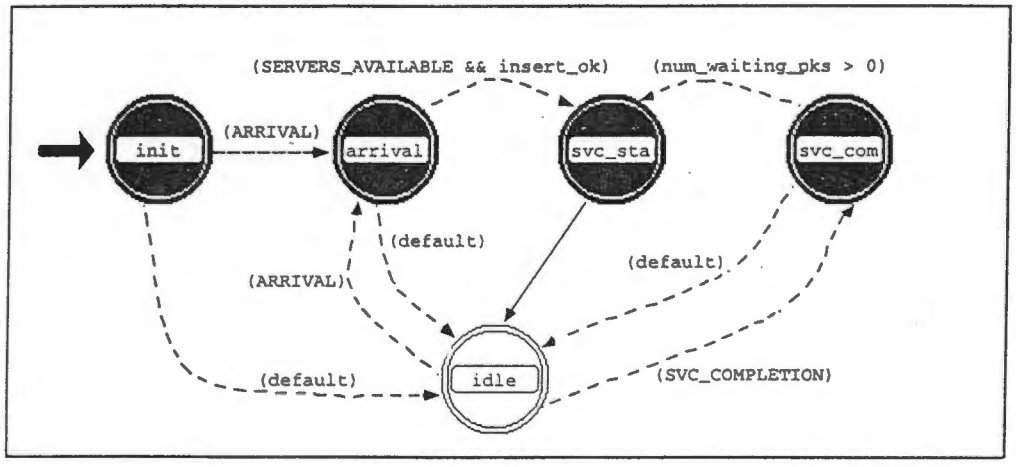


Fig. 5.7 The STD of (atm_acbf_ ms)

4) Count

Another process will count the amount of incoming cells, such as a monitor for the statistic.

Count:

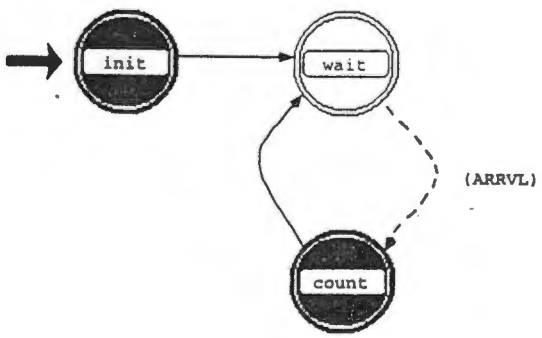


Fig. 5.8 The count in STD

Table 5.4 shows the report for the count process model. The purpose is to make a statistic for incoming cells to every input

port.

Process Model Comments

This is for count the amount of the ATM cells

Process Model Interface Attributes

Attribute **begsim intrpt** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	enabled	N/A
Default Value:	disabled	YES
Data Type:	toggle	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute specifies whether a 'begin simulation interrupt' is generated for a processor module's root process at the start of the simulation.	
Symbol Map:	NONE	YES

Attribute **endsim intrpt** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	toggle	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute specifies whether an 'end simulation interrupt' is generated for a processor module's root process at the end of the simulation.	
Symbol Map:	NONE	YES

Attribute **failure intrpts** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	enumerated	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute specifies whether failure interrupts are generated for a processor module's root process upon failure of nodes or links in the network model.	
Symbol Map:	NONE	YES

Attribute **intrpt interval** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	toggle double	N/A

...
...

Attribute Description:	Private	N/A
Units:	sec.	YES
Comments:		YES
	This attribute specifies how often regular interrupts are scheduled for the root process of a processor module.	
Symbol Map:	NONE	YES

Attribute **priority** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	0	N/A
Default Value:	0	YES
Data Type:	integer	N/A
Attribute Description:	Private	N/A
Low Range:	-32767 inclusive	YES
High Range:	32767 inclusive	YES
Comments:		YES
	This attribute is used to determine the execution order of events that are scheduled to occur at the same simulation time.	
Symbol Map:	NONE	YES

Attribute **recovery intrpts** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	enumerated	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute specifies whether recovery interrupts are scheduled for the processor module's root process upon recovery of nodes or links in the network model.	
Symbol Map:	NONE	YES

Attribute **super priority** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	toggle	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute is used to determine the execution order of events that are scheduled to occur at the same simulation time.	
Symbol Map:	NONE	YES

Header Block

```
#define ARRVL (op_intrpt_type () == OPC_INTRPT_STRM)
```

State Variable Block

```
double \count;
```

Temporary Variable Block

```
Packet * acellp_pk;
```

forced state init

attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs init

```
count=0.0;
```

transition init -> wait

attribute	*value*	*type*	*default value*
name	tr_0	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

unforced state wait

attribute	*value*	*type*	*default value*
name	wait	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition wait -> count

attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition	ARRVL	string	
executive		string	
color	RGB333	color	RGB333

forced state count

attribute	*value*	*type*	*default value*
name	count	string	st
enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs count	
	count++; **op_stat_global_write(op_stat_global_reg("Count1"), count);** **acellp_pk = op_pk_get (op_intrpt_strm ());**
5	**op_pk_send(acellp_pk,0);**

transition count -> wait			
attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

Table 5.4 The report for count process model

5) The point-to-point transmitter

The point-to-point transmitter will transmit the cells to the next module. And the data rate is entered as 155.52 Mbit/s in its **Attributes.**

These nodes (the source, the set, the queue, the input count and the point-to-point transmitter) will be connected by the Packet stream. Each input port module is shown in Fig. 5.9.

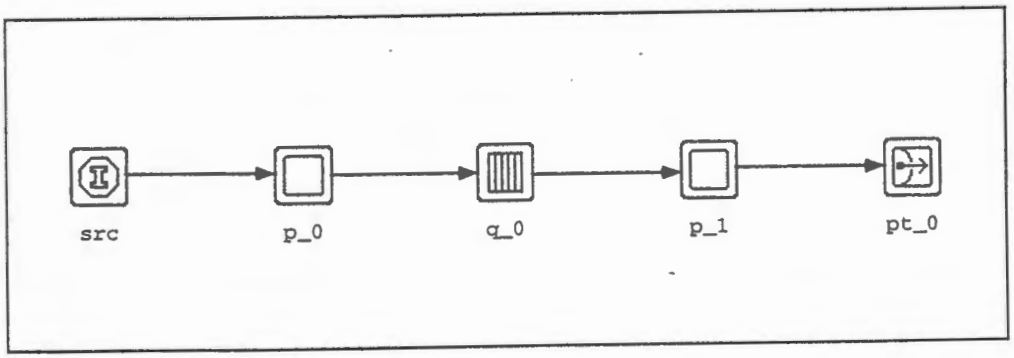


Fig. 5. 9 The Node Model of Input Port module

5.2.3.3 Input queue module:

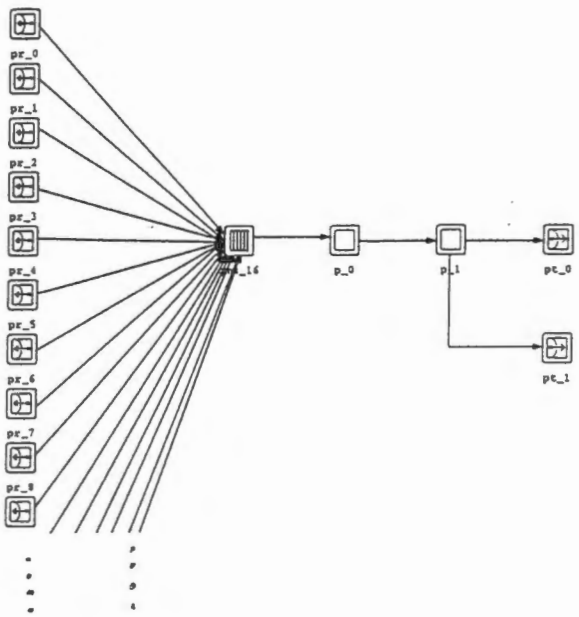


Fig. 5.10 The nodes of input queue module

* 16 point-to-point receivers receive the cells from the input module.

* Queue buffers have a special function, which can change qms_n from the incoming cells (from 16 input ports) in parallel into the cells in outgoing series. The atm_acb_fifo_ms process model is used, and the multi-services number is setup as 16 (The report is shown on Appendix F.). The capacity of the multi-services queue (buffer) is 2.5 Gbit/s. The time slots are decided by the service_rate (bits/s) and the size (in bits) of a cell in the simulation.

* A process model for count can still work for the monitor.

* Another process model can segment a ATM cell into two

packets, one for head of cell (40 bits) and the other one for payload (384 bits). For the simulation sake of convenience, PI (1 bit) could be put with the head of cell together in the simulation. Its attribute of node model is shown on Table 5.6 . It is clear in the making of this process model, the purpose is that the head of cell will go through the routing fabric (the routing table), and the payload of cell will go by bus.

5.2.3.4 Routing fabric module:

Routing fabric is a machine (or a table) which guides cells to the proper output port(s). The routing fabric as a routing table route the head of the cell (40 bits), and use a node model. The Node model is shown in Fig. 5.11. There is a **count** process model in this module as well, which make statistics of the cells going through the routing fabric. The capacity of the module is 2.5 Gbit/s.

A Process Model of the table for routing cells is created. The routing table is the index of the stream from which the packet (ATM_cell) is to be retrieved. The process must obtain the destination address contained in the packet (the head of the cell). The destination address is held in the VPI/VCI.

For example, the value of the VPIs/VCIs of the incoming cells are setup at set process model in the input module, and the

corresponding output port numbers for these addresses can be provided in the table (the module). Meantime, the new addresses (the content of containing original addresses) are brought out as well. They are shown as following:

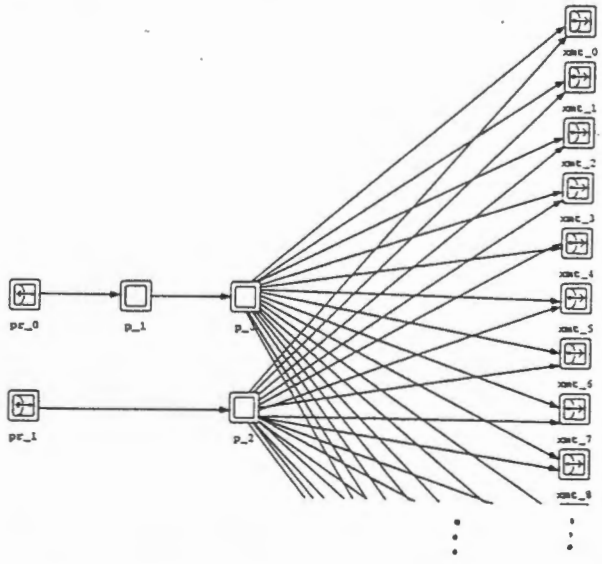


Fig. 5.11 The Node model of Routing Fabric

Input number #1

The value of input VPI=1, VCI=1.

The new address VPI'= 0, VCI'=2.

Output number: 2.

Input number #2

VPI=1, VCI=0.

VPI'=0, VCI'= 0.

Output number: 0 **or** 9. (Number 0 or number 9 output ports are opened.)

Input number #3

VPI = 16, VCI = 0.

VPI' = 6, VCI' = 0.

Output number: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

10, 11, 12, 13, 14, 15. (16 output ports open in same time.)

Input number #4

VPI=100, VCI=1;

Wrong address!

.....

Input number #15

VPI=2, VCI= 3

VPI' = 2, VCI' = 4;

Output number: 2 **and** 3.

Output number: 0 **and** 1. (Number 0 and 1 output port open in same time.)

Of course, this value of VPI'/VCI' is presumed. It can often be changed in the table. If a value of VPI/VCI for a incoming cell is not in the table, the routing table will tell you "Wrong address!".

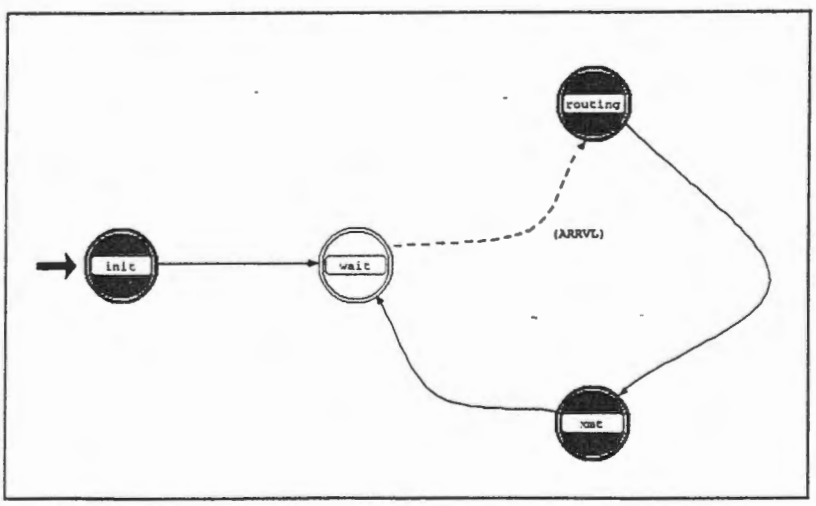


Fig. 5.12 The Process Model of Routing Fabric

The Routing Table

VCI	VPI	Output port number	VCI'	VPI'
1	0	0	0	0
1	1	1	0	1
1	2	2	0	2
1	3	3	0	3
1	4	4	0	4
1	5	5	0	5
1	6	6	0	6
1	7	7	0	7
1	8	8	0	8
1	9	9	0	9
1	10	10	0	10
1	11	11	0	11
1	12	12	0	12
1	13	13	0	13
1	14	14	0	14
1	15	15	0	15
1	16	0 or 9	0	16
1	17	1	0	17
2	1	0 and 1	1	1
2	5	0 and 5	1	5
2	20	1 and 2	1	17
3	0	0, 1 and 2	2	0

Table 5.6 The Routing Table

In the Table 5.6, the routing table is shown for VPI/ VCI of the input, output number, and new address (VPI'/VCI') which contains original address content. The routing table in the Proto_C program is made in the process model for the routing fabric. The VPI's / VCI's are output as they appear on UNIX standard output device (on the UNIX shell).

The fabric delay is monitored in this Process model. The Process model of routing table is shown on Fig. 5.12.

Alternatively, the node is as the bus, and can send the send payload (384 bit) to the output ports of the switching fabric. Its process model is shown on Fig.5.13.

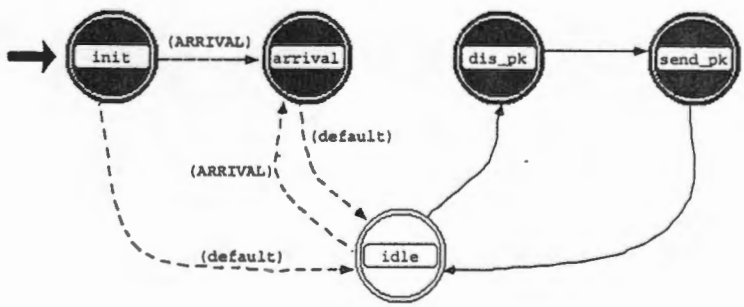


Fig. 5.13 The Process Model for the send

5.2.3.5 Output Module:

By creating the nodes model and the process model of the input and routing fabric modules, the output module is easily created. It receives the head of the cell from routing table, and the payload of the cell from the bus. These cells will be put into the output queue buffers. Then they are sent out successively. According to the principle of the time division ATM switching fabric, the capacity of the output module is as the same as the routing fabric 2.5 Gbits/s.

In the process Models of the output module, the buffers include the reassembly packet (ATM_cell) process model, queue process model and the count process model. The node model of the output module is shown on Fig. 5.14.

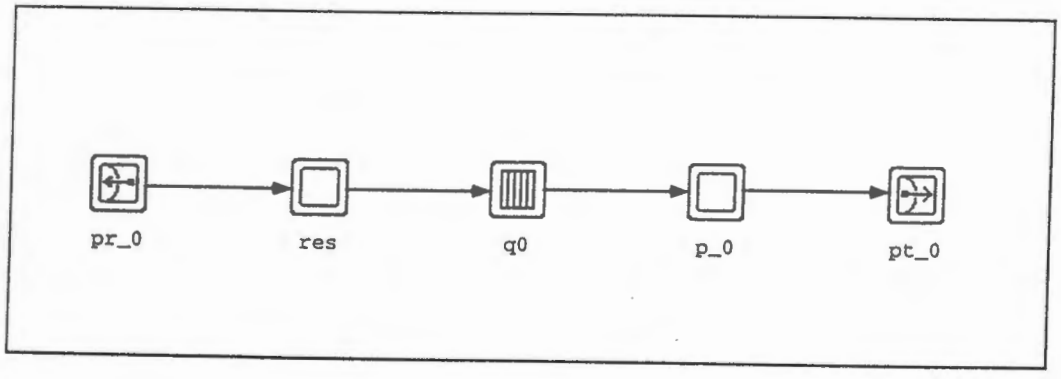


Fig. 5.14 The Node Model of Output Module

The process model for reassembly packet (ATM cell) is an important node in the output module of the TDS fabric. It can be the same as a buffer for a reassembly packet, which can reassemble the head of an ATM cell (40 bits) and the payload (384 bits) to become a whole ATM cell (424 bits), as is shown in

the example Fig.4.8.

Res:

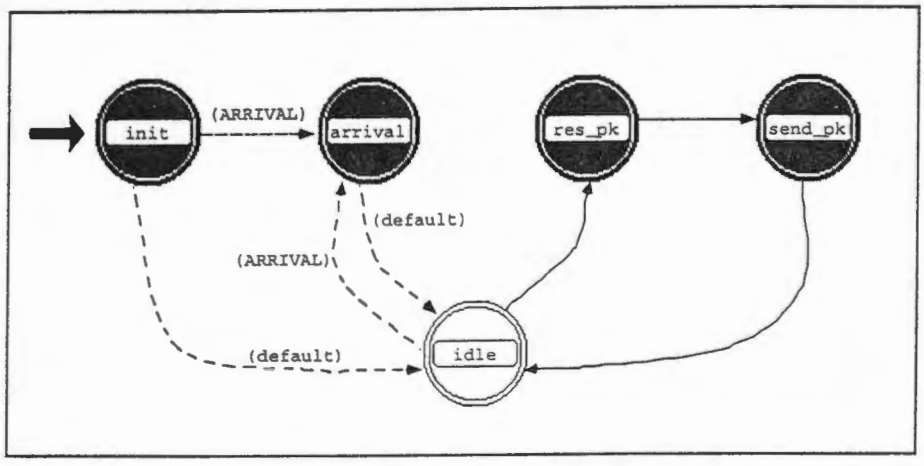


Fig.5.15 The Process Model for Reassembly Packet (cell)

The queue of the output module uses the atm_acb_fifo [43]process model. The STD process model is shown on Fig.5.16. It accepts packets (ATM cells) and autonomously forwards them to a single destination. Enqueued packets wait for the completion of service of earlier arriving the packets before they themselves may begin service and eventually be forwarded. The queue operates with a bits first-in first-out discipline.

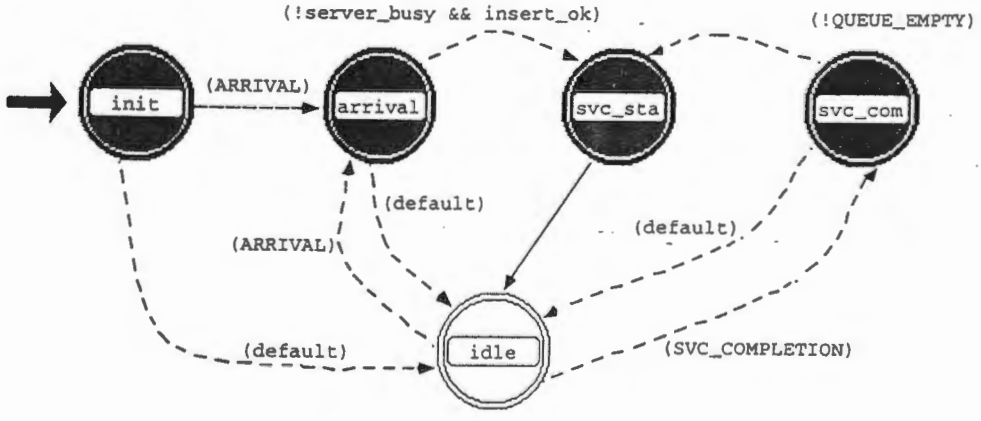


Fig.5.16 The process model for the **atm_acb_fifo**

Therefore, the ATM cells can be changed into the bits in series from the bits in parallel in this queue. The data rate for the transmitter of this node model is still 2.5 Gbits/s.

5.3 Simulation

In section 5.2, the Network, Node and Process models for a local ATM TDS have been described. This section will discuss the implementation of simulation.

5.3.1 Setup the parameters for simulation

In the Probe Tool Window, several parameters have been set-up, for example overflow, source throughput, delay, packet_throughput, packet_size, ... etc. The Probe Editor is used to define data collection requirements prior to executing a simulation. Each desired output is brought about by creating an object and configuring it appropriately. Several types of probes are used to collect different types of simulation results. Actually, the global statistics in the Process Model are used for recording switching fabric delay of a cell.

For this simulation, some parameters are setup in the Probe Tool (They will be shown in chapter 6.).

5.3.2 Executing the Simulation

Once prepared, the simulation will start to execute in the

Simulation Tool Window. The Simulation Tool Window reflects a condition that the Network modules, the Node modules and the Process modules execute in simulation.

The Simulation Tool provides a convenient way to specify simulation parameters and execute simulation from within the OPNET environment. It contains a **Data Table** for specifying the simulation parameters.

According to the characteristics of TDS and ATM protocols, some of the appropriate values of simulation parameter are set into the Data Table. In the following section a number of simulation results of the parameters will be shown on chapter 6.

5.4 Summary

This chapter described a progressing of the simulation for TDS fabric in OPNET in detail. According to the functions and the architecture of the ATM TDS fabric, some modules (the input module and input queue module, the routing fabric module and the output module) were created in **Network** model, **Node** models and **Process** models. Their execution is in a **Proto_c** program of OPNET, and many tables for these models can setup the parameters as the variable functions. The simulation for these models can give a reflection for dynamic implementation. The pattern of these results will be discussed in chapter 6.

CHAPTER 6

Simulation Results and Analysis

In chapter 5, the Network module, Node modules and Process modules which were created have been described in detail. In this chapter, the results of simulation are analysed, and these results also reflect some characteristics for the architecture of the switching fabric.

The Analysis Tool offers the capability of presenting simulation statistics in a variety of two-dimensional graphic formats. The output vector files provide dynamic time-series data show how recorded statistics change over time during a single simulation run. The statistics monitors reflect the result of simulation. The Analysis tool can have multiple vectors plotted on the same graph [44]. and the results of the simulation can be taken from any simulating step. For the TDS fabric, the results and their analysis will be divided into four steps: the input module,input queue module, routing fabric and the output module.

6.1 The Input module

If the Source interarrival args (in the data table of the simulation tool window) of each source is setup at 0.001, the throughput of each source is 1000 cells.

The simulation results can be checked in the input module. Any one of the input ports can be taken out of its simulation result , for example input port number 13, will shown on Fig.6.1.

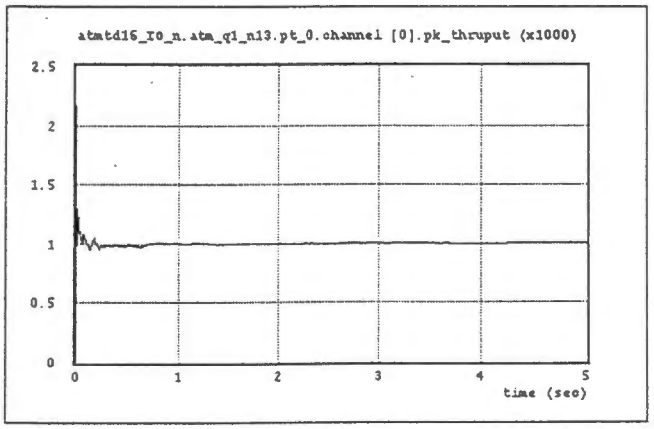


Fig.6.1 The **throughput** of the input port number 13

Or the input port number 12, can be seen on Fig.6.2.

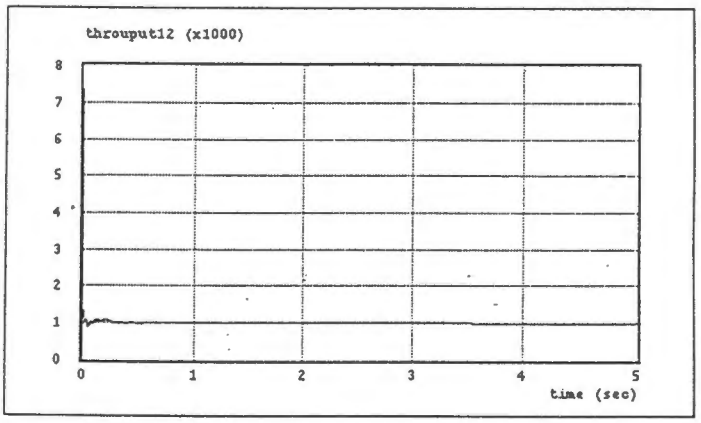


Fig.6.2 The **throughput** of the input port number 12

These two results are the same at 1000 cells. The meaning is that there is no lost cell in the input module.

In Fig.6.3, the statistic for delay of the input module is represented. The result is 7.227 x1e-9 second.

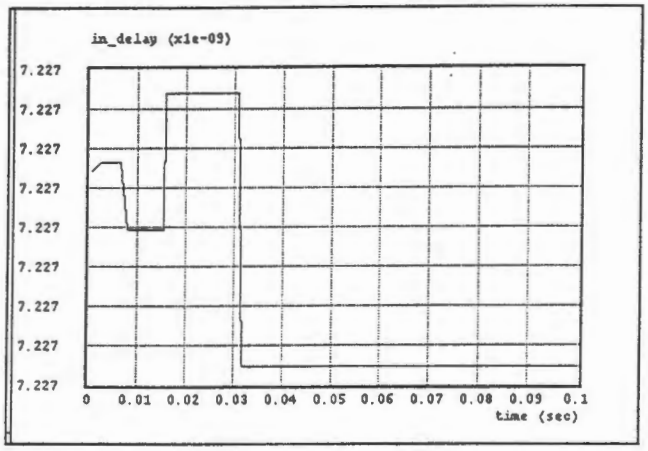


Fig.6.3 The delay for the input module

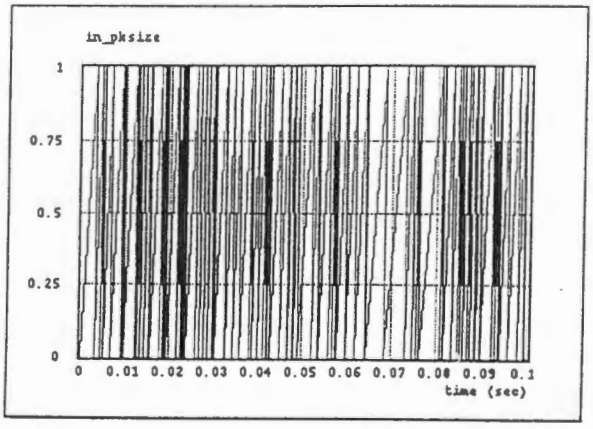


Fig.6.4 The **Pksize** in the input module

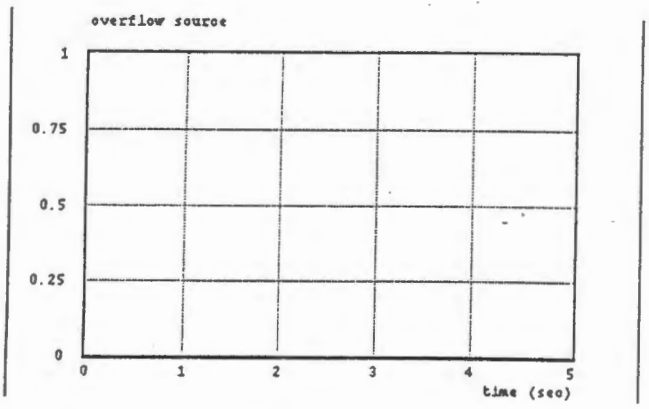


Fig.6.5 The **overflows** of the input module

Only one cell is stored in the input module (Fig 6.4). It testifies that there is no more cells waiting in the input buffer, and there is no overflows (Fig.6.5 will show out this result.)

6.2 The Input Queue Module

In the input queue module, incoming cells for 16 input ports are queued in series. The results of the simulation are shown in Fig.6.6, Fig.6.7, Fig.6.8 and Fig.6.9.

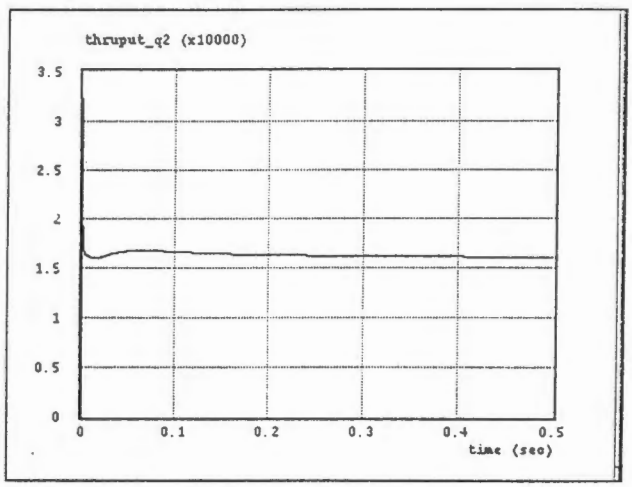


Fig.6.6 The **throughput** of the input queue module

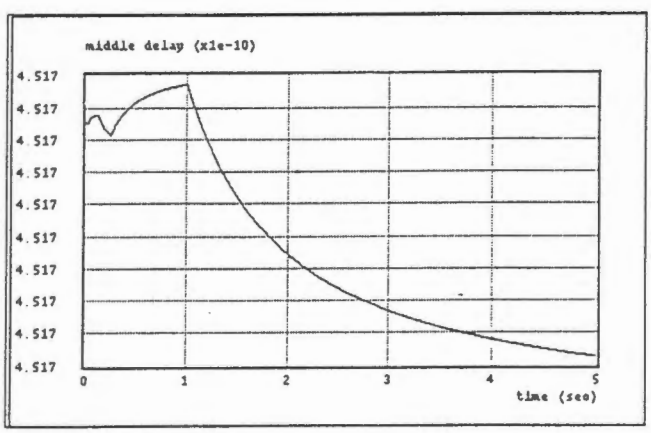


Fig.6.7 The **middle delay** of the input queue module

Fig.6.6 testifies that the sum of throughput of 16 input ports is at 16×1000 cells in the input queue module. The middle delay for the input queue module is very small at 4.517 ×1e-10 second (0.45 *ns*). It can be ignored. The overflow is still zero. The packet size is also 1, so that only one cell queues in the input queue module. The results are very good for the input queue module.

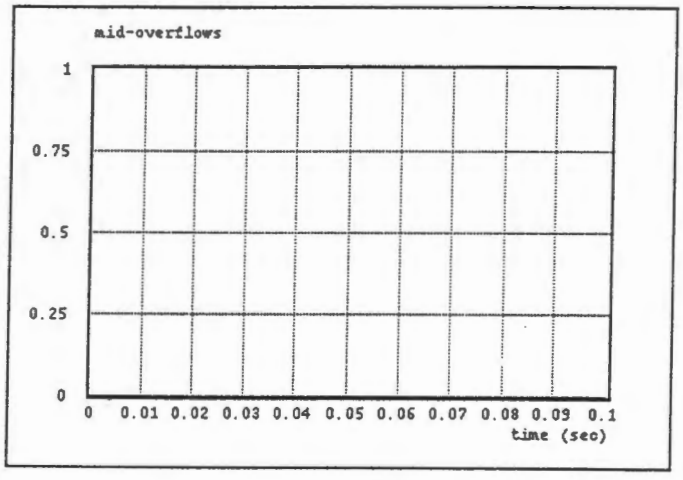


Fig.6.8 The overflows of the input queue module

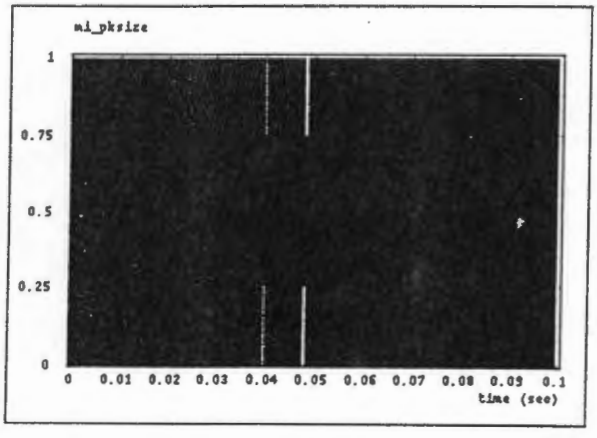


Fig.6.9 The pksize for input queue module

6.3 The Routing Fabric

For the routing fabric, a number of graphics will be shown for

the switching fabric delay (Fig.6.10) and throughput through the routing fabric (Fig.6.11). The throughputs at the point-to-point transmitter xmt_0--16 are shown, so that the effect for executing the routing table can be tested.

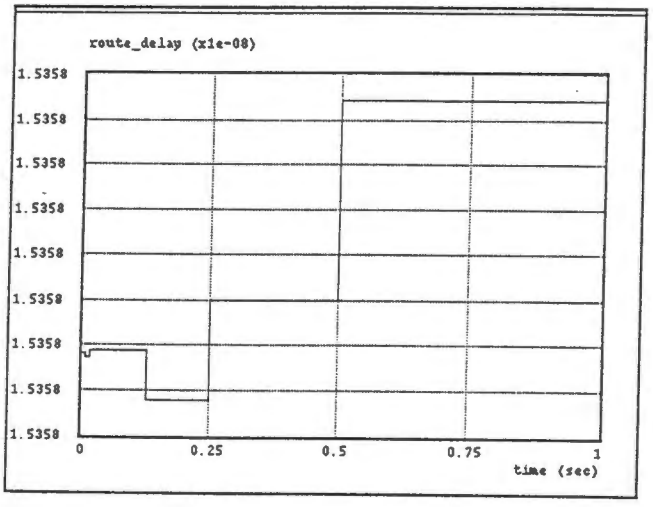


Fig.6.10 The **routing fabric Delay**

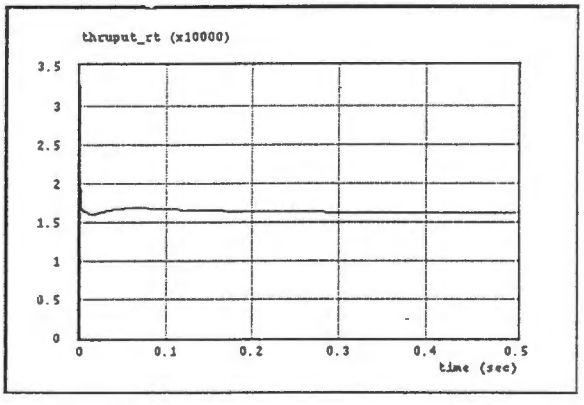


Fig.6.11 The **Throughput** of Routing Fabric

The time for routing fabric delay is very small at 1.5358×1e-8 second in the simulation. In other word, a cell going through this routing fabric goes through at 1.5358 ×1e-8 second. The

throughput of the routing fabric is still 16000 cells.

The input ports for VCI and VPI are setup in Table 6.1 (The VPIs /VCIs are setup at the set process model in the input module.).

Input port number	VCI	VPI	Output port number
#1	7	1	#7
#2	9	1	#9
#3	1	1	#1
#4	4	1	#4
#5	2	1	#2
#6	8	1	#8
#7	12	1	#12
#8	16	1	#0 or #9
#9	17	1	#1
#10	1	2	#0 and #1
#11	5	2	#0 and #5
#12	0	3	#0, #1 and #2
#13	11	1	#11
#14	13	1	#13
#15	15	2	--
#16	7	1	#7

Table 6.1 VCI/VPI setup for the input port

Input port #0: VCI=9, VPI=1. The throughput of the transmitter is (based on the Routing Table): xmt_9. (Another Input port #8 VCI=16 VPI=1 the transmitter is xmt_0 or xmt_9. The xmt_9 is

checked.) Input port #2: VCI=1, VPI=4. The transmitter: xmt_4.

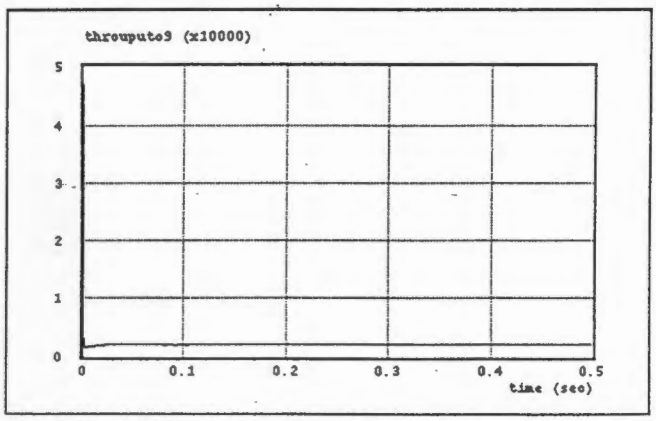


Fig.6.12 The **thruput** for xmt_9

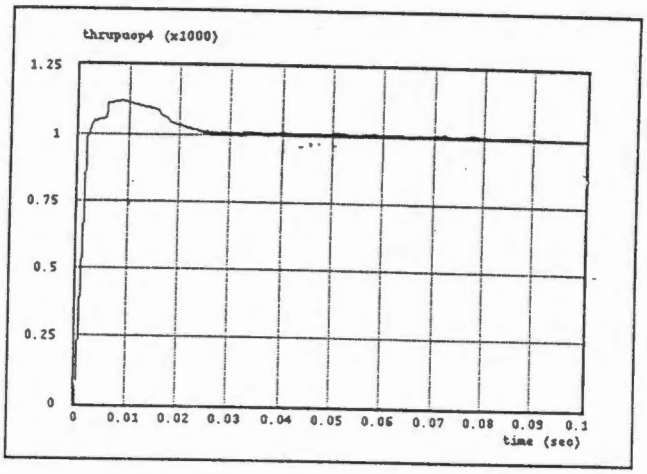


Fig.6.13 the **thruput** for xmt_4

Input port #10 VCI=1 VPI=2, the transmitter: xmt_0 and xmt_1. (Similarly, the input port #11 VCI=5 VPI=2, the transmitter: xmt_0 and xmt_5, and one other input port #12 VCI=0 VPI=3, the transmitter: xmt_0, xmt_1 and xmt_2.) The xmt_0 will be checked.

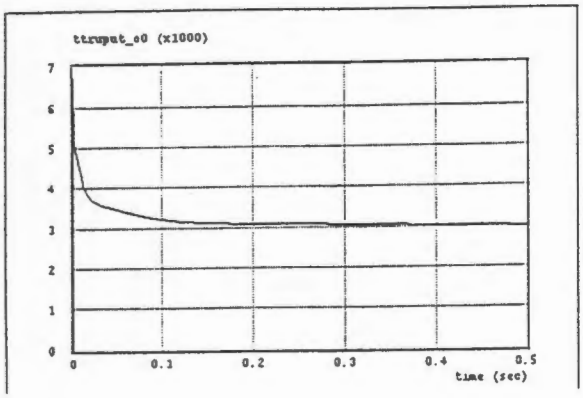
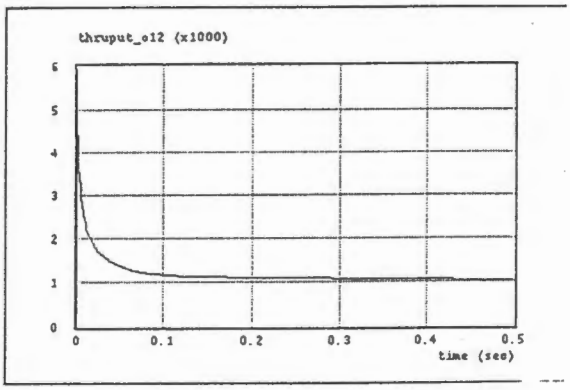


Fig.6.14 The **thruput** for xmt_0



Fig,6.15 The **thruput** for xmt_12

Input port #6 VCI=1 VPI=12, the transmitter: xmt_12;

Input port #13 VCI=1 VPI=11 , the transmitter: xmt_11 ;

Input port #15 VCI=1 VPI=7 , the transmitter: xmt_7 .

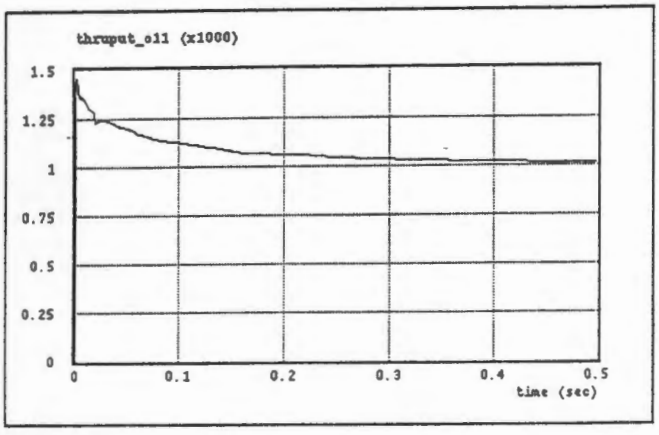


Fig.6.16 The **thruput** for xmt_11

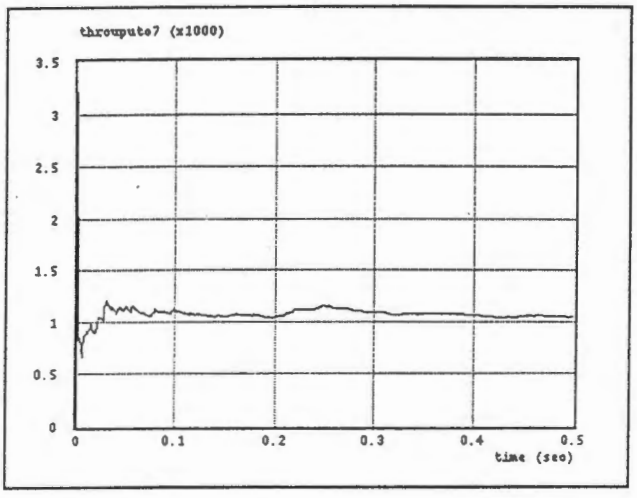


Fig.6.17 The **thruput** for xmt_7

It is very clear from the results for xmt_4, xmt_7, xmt_11 and xmt_12 that each of the throughput are all 1000 cells. For xtm_9, as there are two input ports (#0 and #8) for output xmt_9, and another "or" xmt_0, the machine selects output xmt_9. So, the throughput of xmt_9 is shown at 2000 cells. For xmt_0, as there are three input ports (#10, #11 and #12) for output xmt_0, the result is the throughput for 3000 cells. This is correct!

6.4 The Output Module

The pksize, the throughputs (ATM_cells), the delay of the output module, and the overflows will be checked in a number of the output ports. The output port number 3, number 10, and number 12 are selected as the object of the selective examination.

Output port number #3:

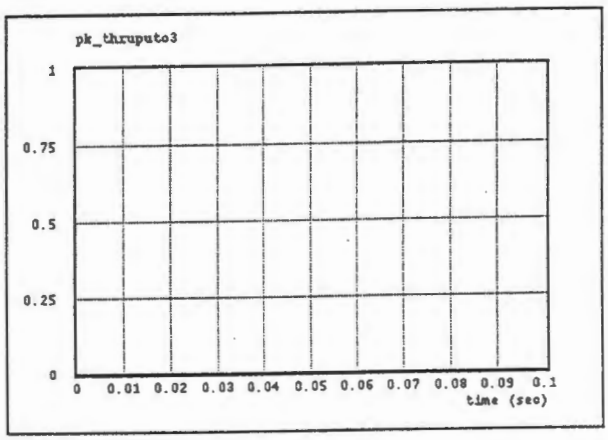


Fig.6.18 The **throughput** of output port #3

Output port number #4:

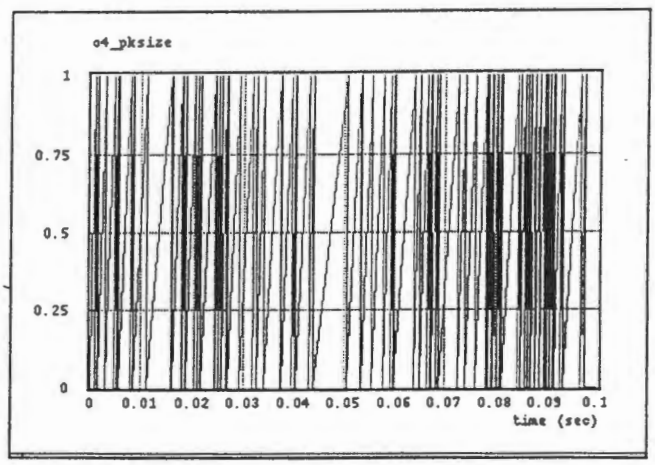


Fig.6.19The **Pksize**

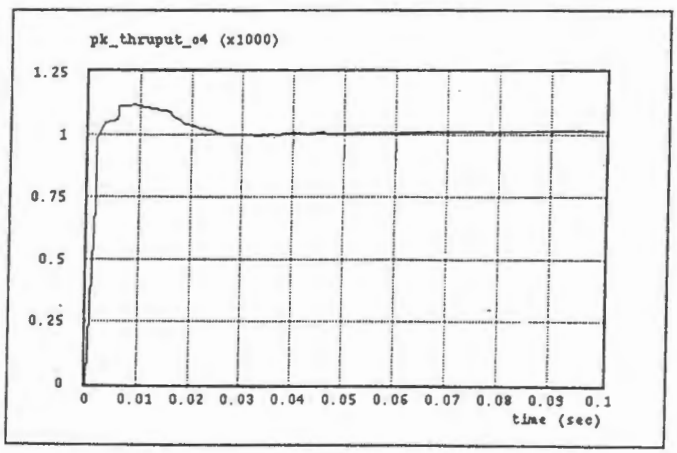


Fig.6.20 The **throughput** of output port #4

Output port number #13:

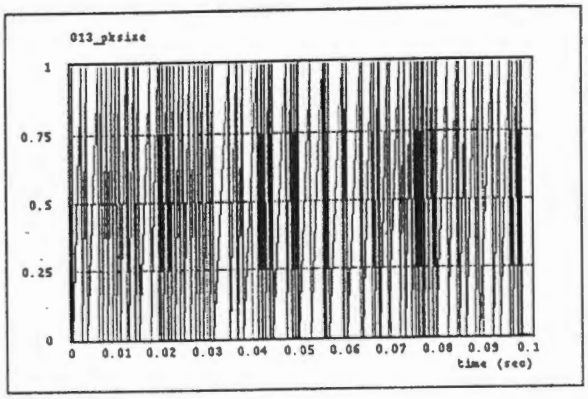


Fig.6.21 The **Pksize** for output port #13

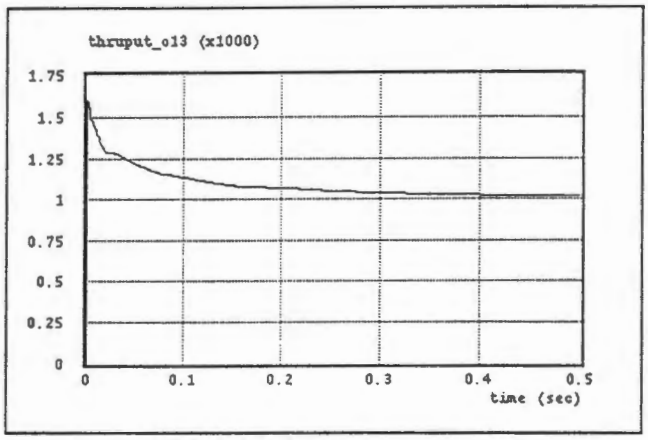


Fig.6.22 The **throughput** of output port #13

From these figures, output port #3's is zero [because there are no incoming cells going the through output port #3 (seen on the routing table)], each of #4 and #13 are at a 1000 cells for throughput, and their pksize is 1. For the output module, considering that the data rate for the output module is the same as the routing fabric's, and the delay for the switching fabric is very, very small (about 23ns) (much less than the time slot for 170ns), only one cell can be stored in the output buffer, therefore no other cells can queue in the output buffer. These are very good facts.

6.5 Others

In the preceding four sections, some of the parameters were fixed in the simulation. But we wish to know what results will change when certain parameters are changed.

In the simulation, some of the parameters can be changed, for example:

1) source interarrival args

The source interarrival args is an attribute for the parameter which determines the distribution packet interarrival time. In the preceding sections, the source interarrival args is setup at 0.001. Now, it will be changed into 0.00000274.

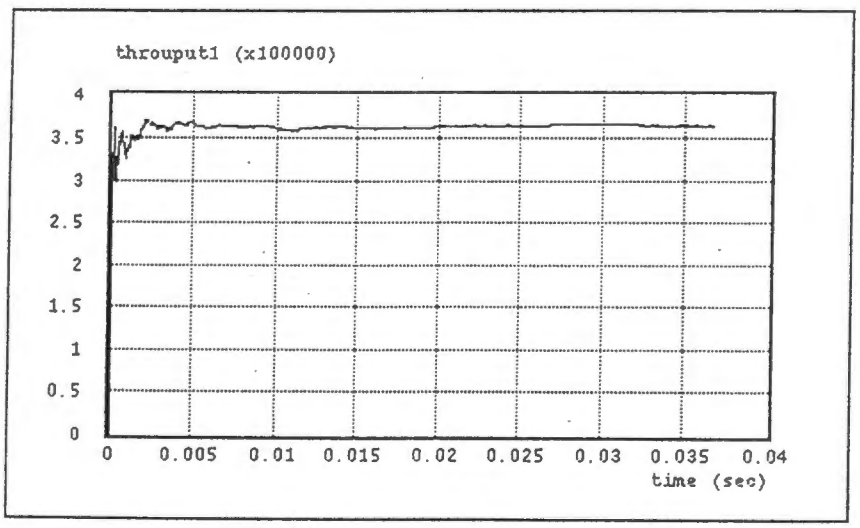


Fig.6.23 Throughput for atm_q1_n1

From Fig.6.23 and Fig.6.24, it is very clear when the source interarrival args is 0.00000274, the cell throughput is changed,

but the delay for routing fabric is not changed.

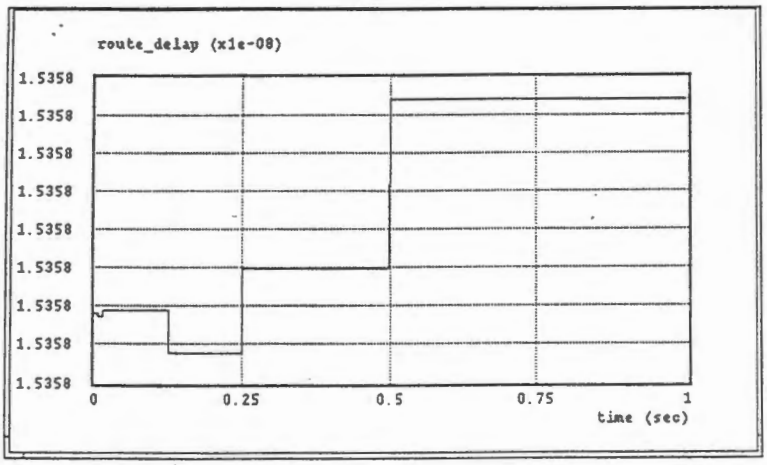


Fig.6.24 The delay for routing fabric

2) Input ports / output ports

This architecture is designed for 16 prts. If the input / output ports are changed into 4 input / output ports, what result will happen? Of course, the cell throughput is not changed (the source interarrival args is still 0.001). But the delay for the routing fabric is changed (see Fig.6.25).

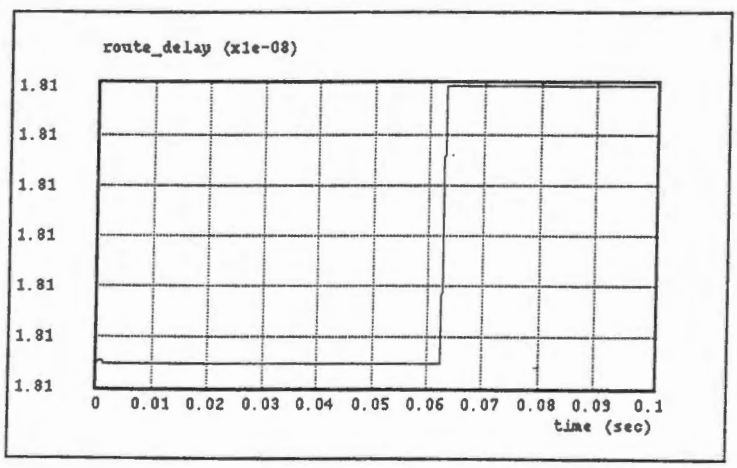


Fig. 6.25 The delay for the routing fabric

The value is shown at 1.810 xle-8 second. The regular pattern is not that input / output ports are less, but that the delay is less. They are not proportional.

These results (for the source interarrival args and the input /output ports) indicate other characteristics for the architecture of the switching fabric.

6.6 Summary

These graphics can provide us with a number of points such as:

1) The time division switching fabric system overflows are absolute Zero.

2) When the throughput of each input port is 1000 cells (to set-up the source interarrival args is 0.001), the sum of the input is 16000 cells. The throughput for the input queue module and the sum of receiving for the routing fabric are also 16000 cells. After going through the routing fabric, the cells-throughput depends on the routing table. The results are all correct. Therefore, the switching fabric system is implemented reliably.

3) In the simulation, the routing fabric delay is in 15.358 *ns* scope, the delay for input module is at 7.227 *ns*, and the delay for the input queue module is at 0.4517 *ns*. These values are considered small enough. However, a cell spends

only 23 *ns* going through the switching fabric in a time slot for 170 ns.

4) In each of the modules, the packet size (the pksize) is checked and their value of the results is all 1. This testifies that there is only one cell stored in the each buffer (the input buffer and the output buffer). No more cells queue in the buffers.

According to the design of the switching fabric architecture, it has been predicted that there will be low cell loss probability, no congestion, low delay and non-blocking performance. In conclusion, the above points show that the plan of design is rational, and tallies with the simulation results. Therefore, the switching architecture of the local ATM Time Division--shared medium for 16 ports has those expected characteristics previously referred to.

CHAPTER 7

Conclusion

The cell-based packet switching technique of ATM was driven by the research and development of high-speed switching fabrics. Innumerable different kinds of switching fabric architectures have been created. For example: shared memory, shared medium, Banyan, Crossbar,....etc. These architectures have very good individual characteristics.

In exploring ATM LAN switching fabric, the time division--shared medium architecture is selected to be used as the objective of a research project.

The ATM time division switching (TDS) has 16 ports, 2.5 Gbits/s capacity in the switching fabric. The performance is based on scheduling the input sequential and transferring (routing) the cells in bits-parallel.

A simple architecture, higher speed processing, non-blocking performance, and no congestion within are the important characteristics of the architecture.

Through the simulation in OPNET, the results for the switching fabric confirm no over flow, correct for routing cells and a very small switching fabric delay (only about 23 ns). In each

of the modules, there is only one cell stored in each buffer (the input buffer and the output buffer). Especially, no more other cells queue in the output module.

The design for the ATM switching fabric architecture is, at this stage to be regarded as exploratory and with need for further research. This research will promote and develop the communication network technology with other designs for ATM switching fabric architectures.

References

References

[1] Rainer Nandel, Manfred N. Huber " Integrated Broadband Network An introduction to ATM-Based Networks " 1993

[2] I Gallagher, J Ballance and J Adams "The application of ATM techniques to the local network " British Telecom Technology Vol.7 No.2 April 1989

[3] Michael J. Rider "Protocols for ATM Access Networks" IEEE Network January 1989

[4] David Cleevely "ATM - The End of Telecommunications as we Know it?"

[5] Seminar Handbook (HEWLETT PACKARD) "Broadband and Testing Technologies" 1993

[6] John D. Spragins "Evolution Toward Broadband ISDN" Telecommunications (Protocols and design)

[7] J.Dupraz, M.De Prycker "Principles and Benefits of the Asynchronous Transfer Mode" Electrical Communication, 1990 No. 2/3

[8] Ken-Ichi Sato, Satoru Ohta and Ikuo Tokizawa " Broad-Band ATM Network Architecture Based on Virtual Paths" IEEE Communication 1990

[9] Rainer Handel "Evolution of ISDN Towards Broadband ISDN" IEEE Network January 1989

[10] Lianghong Yu "Time Division ATM Switching" Sep. 1995

[11] Guru M. Parulkar "Local ATM Networks" IEEE Network, March 1993

[12] Laurence A. Crutcher, A.Gill Waters "Connection Management for an ATM Network" IEEE Network, Nov. 1992

[13] ITU-T Rec. I.321-B-ISDN Protocol Reference Model and its application

[14] Steven E. Minzer "Broadband ISDN and Asynchronous Transfer Mode (ATM)" IEEE Communication, Sep.1989

[15] Jean-Yves Le Boudec " The Asynchronous Transfer Mode: a tutorial" Computer Networks and ISDN System, 24 1992

[16] ITU-T Rec. I.361--B-ISDN ATM Layer specification

[17] " ATM Local-Area Network : Technical Overview" Business Communications Review January 1993

[18] Peter Newman "ATM Local Area Networks" IEEE Communications March 1994

[19] Author, Martin Taylor "LAN Emulation Over ATM" November 1994

[20] Daniel Abensonr, Randall Campbell "Where , Oh where, did the Routers Go?" SynOptics Press Background

[21] Edoardo Biagioni, Eric Cooper and Robert Sansom " Designing a Practical ATM-LAN" IEEE Network . March 1993

[22] Chris Dhas, Vijaya K. Konangi and M. Sreetharan "Broadband Switching Architectures, Protocols, design and analysis"

[23] Martin de Prycker "Asynchronous Transfer Mode Solution for B-ISDN " Second Edition 1993 Page 152 "Switching Requirements"

[24] Martin de Prycker "Asynchronous Transfer Mode Solution for B-ISDN " Second Edition 1993 Page 154 "Cell loss probability"

[25] Martin de Prycker "Asynchronous Transfer Mode Solution for B-ISDN " Second Edition 1993 Page 155 "Switching delay"

[26] Thomas M. Chen, Stephen S. Liu " ATM Switching System"

[27] Raif O.Onvural "Asynchronous Transfer Mode Networks" Performance issues 1994 Chapter 7

[28] David Wright " BROADBAND: Business Services, Technologies, and Strategic Impact"

[29] Ra'ed Y. Awdeh, H.T. Mouftah "Survey of ATM switch architecture" Computer Networks and ISDN Systems 27 (1995) 1567-1613

[30] Fore System, Inc. "ATM Switch Architectures" November 1993

[31] Joy Yu "Time Division ATM switching" Sep. 1995 Page 2-5

[32] ITU-T Rec. I.211 - B-ISDN Service Aspects

[33] C. Fayet, A.Jacques, G. Pujolle " High Speed Switching for ATM : the BSS" Computer Networks and ISDN Systems 26 1994

[34] J.Garcia-Haro " ATM Shared-Memory Switching Architectures " IEEE Network July / August 1994

[35] Intel Company "Memory Components Handbook" 1988

[36] MIL 3, Inc. "Modelling Manual " OPNET Modular No.2.0 1995

[37] MIL 3, Inc. "Simulation Kernel Manual" OPNET Modeler No. 5.0 1995

[38] MIL 3, Inc. "Example Models Manual" OPNET Modeler No. 8.1.0 1995

[39] MIL 3, Inc. "Tutorial " OPNET Modeler No. 1.0 1995

[40] Kuo-Hsien Chen, Liren Zhang " Implementation of ATM Switch Using OPNET" Australian Telecommunication Network & Applications Conference Dec. 1994

[41] MIL 3, Inc. "Simulation Kernel /Anim-PK " No.6 1997

[42] MIL 3, Inc. "Example Models Manual-Base and General Models Vol.0" Process Models/Queuing (Pro/Q-10) 1995

[43] MIL 3, Inc. "Example Models Manual-Base and General Models Vol.0" Process Models/Queuing (Pro/Q-4) 1995

[44] MIL 3, Inc. "Modelling Manual-- Data Analysis" OPNET Modeler No. 3 1997

Appendix A

ATM Local Area Network

Liang-hong Yu (Joy Yu)
Department of Electrical Engineering
University of Cape Town

Abstract

In future, Asynchronous Transfer Mode (ATM) is an important mode of communication. ATM is a high speed switching fabric that supports voice, data, images and video. Its technological application to the local and campus area networking environments is currently being researched. ATM network has become a trend of communication technology in the world. In this paper, one seeks to introduce the concepts and the protocols of ATM, the principles and functions of ATM Local Area Networks, an analysis of the ATM fast packet switching , finishing with a brief plan of the design.

Introduction

Most networks are dedicated to specific services, telephony, TV distribution, circuit-switched or packet switched data transfer.
From 1985 to 1990, ITU (International Telecommunication Union)-T (CCITT) study Group XVIII agreed on a number of recommendations to describe the basic parameters of ATM [1]. Refer to Fig.1 for the ATM reference model [2]. The ATM layer is to play the role of relaying that is the functions of switching and transmitting data in the network.

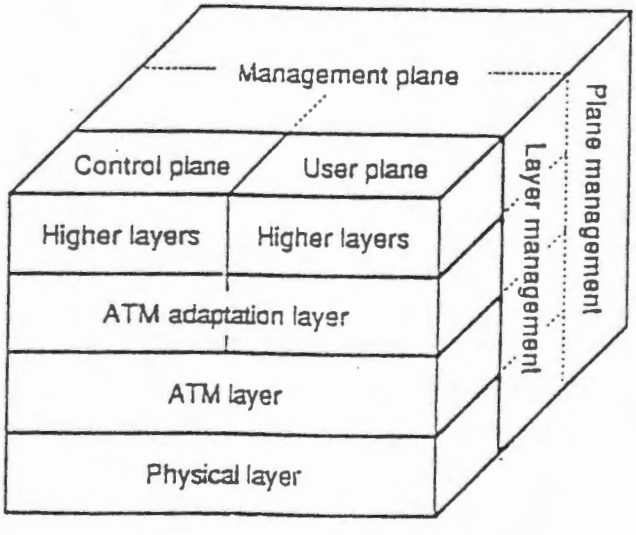


Figure 1. The Protocol Reference Model

The ATM transfer mode supporting Broadband - Integrated Services Digital Network (B-ISDN) is independent of transport at the Physical Layer (PL). ATM uses cells with a 48 bytes payload and a 5 bytes header to transfer information .

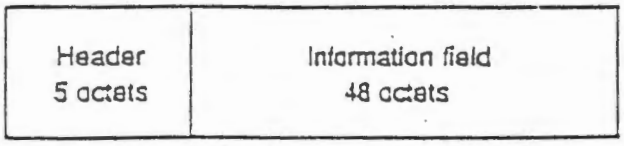


Figure 2. ATM Cell Structure

The main function of the header is to identify the virtual connection, that is the virtual channel identify (VCI), and the virtual path identifier (VPI) . (See Fig. 3) Additional fields cater for priority handling, and header error control, etc.

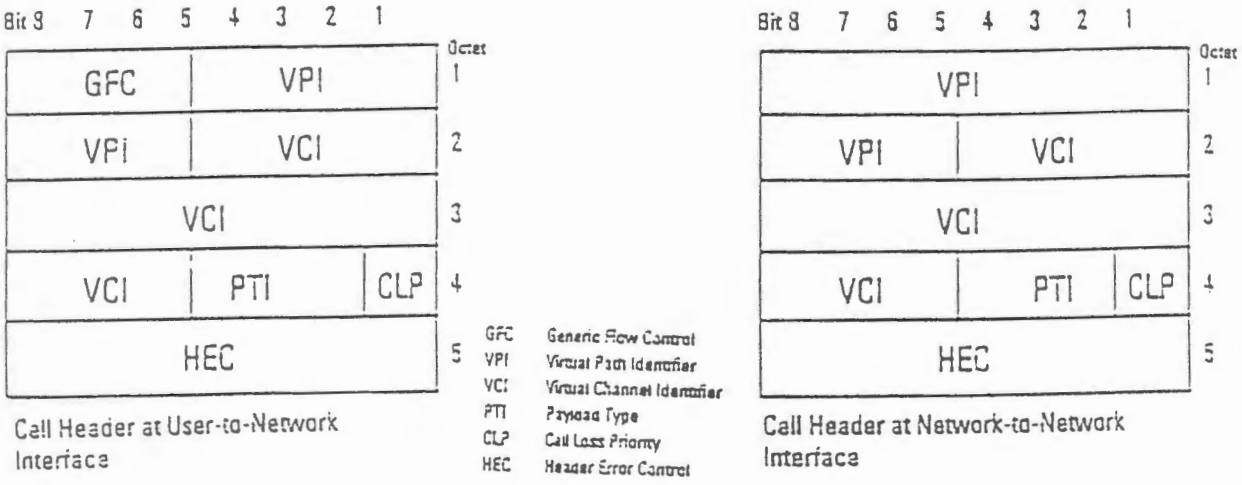


Figure 3. Header of the Cell

The header of the cell [3] mainly consists of a binary number representing a logical address, which is the only information available for routing the cell through the network. Data conveyed in the information field can be from any origin and of any type, including digitised images, digitised speech, computer files, or signalling information.
The cells are generated by the information source, some cells may be unassigned, and other assigned. The ratio between the number of cells assigned and the total number of the cells defines the load of the multiplexer. An ATM multiplexing is an infinite stream of continuous cells [4]. (See Fig. 4)

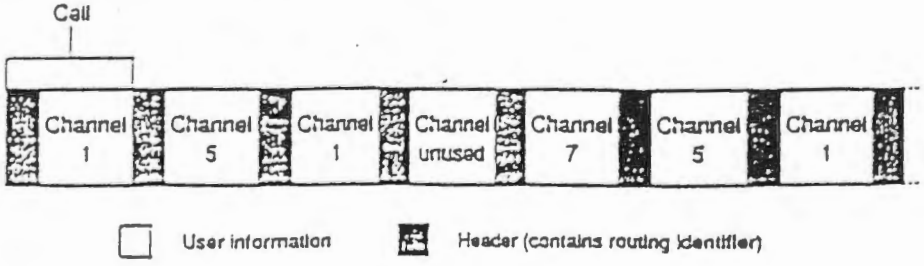


Figure 4. ATM Multiplexing

The multiplexing technique is statistical, allowing true integration of service of any type or bit rate on a single transmission medium. Its capacity is very flexible as it adapts the transport resources to the information flow rates.

The task of the switch is to translate the VCI/VPI, and cell transfer from its input to a dedicated output, by examining the header label of every cell in the stream and route it according to its predefined path. ATM switching using the packet switching resolved modern high speed transfer information.

A number of factors contribute to making ATM a very important development: the multiplex technology makes very efficient use of network capacity for a wide range of communications services. ATM can be applied in every area of network, LAN, WAN etc. ATM cell implies that simpler and faster hardware techniques can be used, and very high bandwidth links (2 Gbit/s) can be switched; it supports voice, data, video and multimedia. Therefore, ATM technology and applications would bring a revolution in communications.

The ATM LAN

The ATM LAN would use a mesh topology, high-speed cell switching, and standard ATM protocols.
In the IEEE802 family of LAN protocols (see Fig. 5) [5], a MAC (medium access control) sublayer should be developed for ATM LAN that offers the same connectionless MAC service as the IEEE802 and FDDI (Fiber Distribution Interface) MAC sublayer.

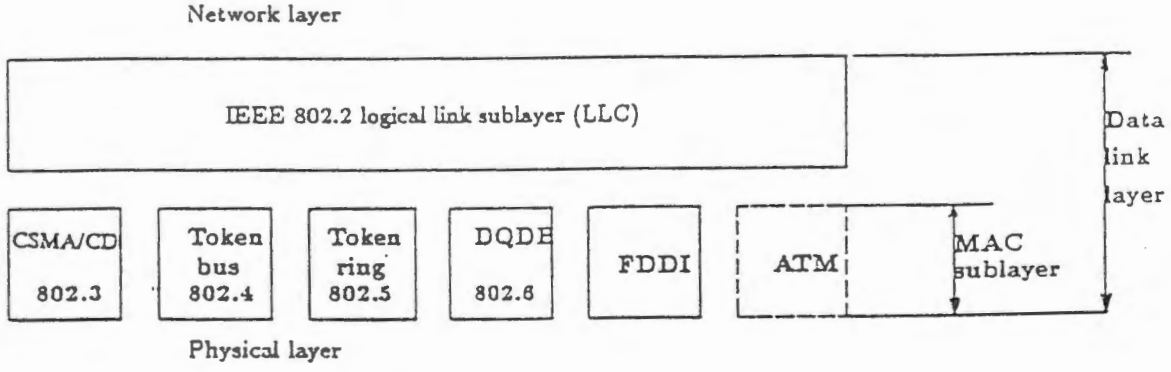


Figure 5. The IEEE 802 family of LAN protocols.

The purposes of using ATM LAN are:
1) Emerging network applications, such as video distribution, computer imaging, etc.
2) Performance of computers, including workstation , etc.
3) Backbone ATM, for connecting existing hubs, bridges and routes throughout a campus.
4) Connectivity with wide-area ATM networks.
5) High bandwidth links are relatively inexpensive in local and campus networking environments and its deployment will be economically justified.

As a result, an ATM LAN would include adapter and driver software for workstation, together with a local ATM switch and switch control software. (The data rates between workstation ATM adapter would be 155 Mb/s .) The ATM hardware and software provide some alternative services to current shared-media LANs.
ATM host interfaces and local ATM switches are the elements of an ATM LAN. The ATM host interface allows each host to connect to the network, and its higher layers could

Appendix D

Electronic Component for RAM

intel

2148H FAMILY
1024 x 4 BIT STATIC RAM

	**2148H-2	2148H-3	2148H	**2148HL-3	2148HL
Max Access Time (ns)	45	55	70	55	70
Max Active Current (mA)	150	**150	**150	125	125
Max Standby Current (mA)	30	30	30	20	20

- **Improved Performance Margins**
- **Automatic Power-Down**
- **Single +5V Supply**
- **Completely Static Memory—No Clock or Timing Strobe Required**
- **HMOS* III Technology**
- **Common Data Input and Output**
- **Three-State Output**
- **High Reliability Plastic or CERDIP Package**

The Intel 2148H is a 4096-bit static Random Access Memory organized as 1024 words by 4 bits using HMOS III, an ultra high-performance MOS technology. It uses a uniquely innovative design approach which provides the ease-of use features associated with non-clocked static memories and the reduced standby power dissipation associated with clocked static memories. To the user this means low standby power dissipation without the need for clocks, address setup and hold times, nor reduced data rates due to cycle times that are longer than access times.

$\overline{CS}$ controls the power-down feature. In less than a cycle time after $\overline{CS}$ goes high—disabling the 2148H—the part automatically reduces its power requirements and remains in this low power standby mode as long as $\overline{CS}$ remains high. This device feature results in system power savings as great as 85% in larger systems, where the majority of devices are disabled. A non-power-down companion, the 2149H, is available to provide a fast chip select access time for speed critical applications.

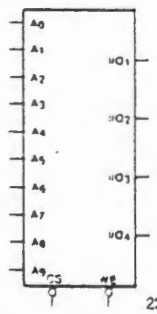
The 2148H is assembled in an 18-pin plastic package configured with the industry standard 1K x 4 pinout. It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. The data is read out nondestructively and has the same polarity as the input data.

*HMOS is a patent process of Intel Corporation. **Improved performance margins.

Pin Configuration

230781-1

Logic Symbol



230781-2

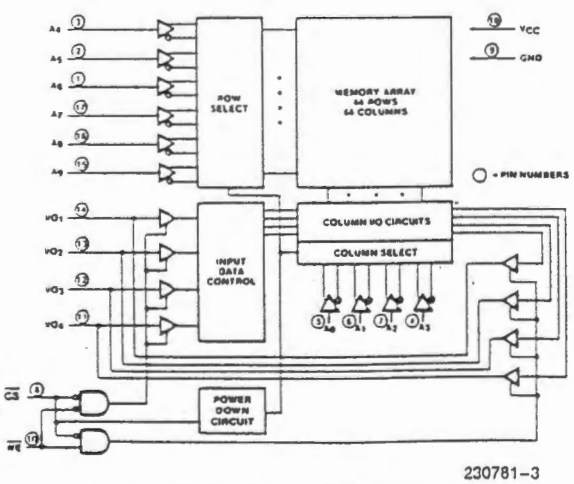
Pin Names

A_0–A_9	Address Inputs
$\overline{WE}$	Write Enable
$\overline{CS}$	Chip Select
VO_1–V_4	Data Input Output
V_{CC}	Power (+5V)
GND	Ground

Truth Table

$\overline{CS}$	$\overline{WE}$	Mode	VO	Power
H	X	Not Selected	High Z	Standby
L	L	Write	D_{IN}	Active
L	H	Read	D_{OUT}	Active

Figure 1. Pin Configuration, Logic Symbol, Pin Names and Truth Table



230781-3

Figure 2. 2148H Block Diagram

Appendix E

atm_tds16_net Node Model Report

Keywords			
attribute	*value*	*type*	*default value*
keywords	custom_model_list	typed file	
	atm_tds_m	typed file	

subnet atm_tds16_n			
attribute	*value*	*type*	*default value*
name	atm_tds16_n	string	n
priority	0	integer	0
user id	0	integer	0
x position	-67.2	double	0.0
y position	37.2	double	0.0
x center	-67.2	double	0.0
y center	37.2	double	0.0
x span	36	double	0.0
y span	18	double	0.0
threshold	0.0	double	0.0
map	NONE	typed file	NONE
icon name	subnet	icon	subnet
outline color	RGB133	color	RGB133
...1_n1.source interarrival args	promoted	string list	1.0
atm_q1_n1.q_0.num_servers	promoted	integer	1
atm_q1_n1.q_0.service_rate	promoted	double	9,600
atm_q1_n1.q_0.strict_fifo	promoted	integer	0
...1_n2.source interarrival args	promoted	string list	1.0
atm_q1_n2.q_0.num_servers	promoted	integer	1
atm_q1_n2.q_0.service_rate	promoted	double	9,600
atm_q1_n2.q_0.strict_fifo	promoted	integer	0
...1_n3.source interarrival args	promoted	string list	1.0
atm_q1_n3.q_0.num_servers	promoted	integer	1
atm_q1_n3.q_0.service_rate	promoted	double	9,600
atm_q1_n3.q_0.strict_fifo	promoted	integer	0
...1_n4.source interarrival args	promoted	string list	1.0
atm_q1_n4.q_0.num_servers	promoted	integer	1
atm_q1_n4.q_0.service_rate	promoted	double	9,600
atm_q2_n.p_1.service_rate	promoted	double	9,600
atm_q2_n.qms_16.num_servers	promoted	integer	1
atm_q2_n.qms_16.service_rate	promoted	double	9,600
atm_q2_n.qms_16.strict_fifo	promoted	integer	0
atm_rt_n.p_2.service_rate	promoted	double	9,600
node_o0.p_1.service_rate	promoted	double	9,600
node_o0.q0.service_rate	promoted	double	9,600
node_o1.p_1.service_rate	promoted	double	9,600
node_o1.q0.service_rate	promoted	double	9,600
node_o2.p_1.service_rate	promoted	double	9,600
node_o2.q0.num_servers	promoted	integer	1
node_o2.q0.service_rate	promoted	double	9,600
node_o2.q0.strict_fifo	promoted	integer	0
node_o3.p_1.service_rate	promoted	double	9,600
node_o3.q0.service_rate	promoted	double	9,600
...1_n5.source interarrival args	promoted	string list	1.0

...
...

atm_q1_n5.q_0.num_servers	promoted	integer	1
atm_q1_n5.q_0.service_rate	promoted	double	9,600
atm_q1_n5.q_0.strict_fifo	promoted	integer	0
...1_n6.source interarrival args	promoted	string list	1.0
atm_q1_n6.q_0.num_servers	promoted	integer	1
atm_q1_n6.q_0.service_rate	promoted	double	9,600
atm_q1_n6.q_0.strict_fifo	promoted	integer	0
...1_n7.source interarrival args	promoted	string list	1.0
atm_q1_n7.q_0.num_servers	promoted	integer	1
atm_q1_n7.q_0.service_rate	promoted	double	9,600
atm_q1_n7.q_0.strict_fifo	promoted	integer	0
...1_n8.source interarrival args	promoted	string list	1.0
atm_q1_n8.q_0.num_servers	promoted	integer	1
atm_q1_n8.q_0.service_rate	promoted	double	9,600
atm_q1_n8.q_0.strict_fifo	promoted	integer	0
...1_n9.source interarrival args	promoted	string list	1.0
atm_q1_n9.q_0.num_servers	promoted	integer	1
atm_q1_n9.q_0.service_rate	promoted	double	9,600
atm_q1_n9.q_0.strict_fifo	promoted	integer	0
..._n10.source interarrival args	promoted	string list	1.0
atm_q1_n10.q_0.num_servers	promoted	integer	1
atm_q1_n10.q_0.service_rate	promoted	double	9,600
atm_q1_n10.q_0.strict_fifo	promoted	integer	0
..._n11.source interarrival args	promoted	string list	1.0
atm_q1_n11.q_0.num_servers	promoted	integer	1
atm_q1_n11.q_0.service_rate	promoted	double	9,600
atm_q1_n11.q_0.strict_fifo	promoted	integer	0
..._n12.source interarrival args	promoted	string list	1.0
atm_q1_n12.q_0.num_servers	promoted	integer	1
atm_q1_n12.q_0.service_rate	promoted	double	9,600
atm_q1_n12.q_0.strict_fifo	promoted	integer	0
..._n13.source interarrival args	promoted	string list	1.0
atm_q1_n13.q_0.num_servers	promoted	integer	1
atm_q1_n13.q_0.service_rate	promoted	double	9,600
atm_q1_n13.q_0.strict_fifo	promoted	integer	0
..._n14.source interarrival args	promoted	string list	1.0
atm_q1_n14.q_0.num_servers	promoted	integer	1
atm_q1_n14.q_0.service_rate	promoted	double	9,600
atm_q1_n14.q_0.strict_fifo	promoted	integer	0
..._n15.source interarrival args	promoted	string list	1.0
atm_q1_n15.q_0.num_servers	promoted	integer	1
atm_q1_n15.q_0.service_rate	promoted	double	9,600
atm_q1_n15.q_0.strict_fifo	promoted	integer	0
..._n16.source interarrival args	promoted	string list	1.0
atm_q1_n16.q_0.num_servers	promoted	integer	1
atm_q1_n16.q_0.service_rate	promoted	double	9,600
atm_q1_n16.q_0.strict_fifo	promoted	integer	0
node_o4.p_1.service_rate	promoted	double	9,600
node_o4.q0.service_rate	promoted	double	9,600
node_o5.p_1.service_rate	promoted	double	9,600

...
...

node_o5.q0.service_rate	promoted	double	9,600
node_o6.p_1.service_rate	promoted	double	9,600
node_o6.q0.service_rate	promoted	double	9,600
node_o7.p_1.service_rate	promoted	double	9,600
node_o7.q0.service_rate	promoted	double	9,600
node_o8.p_1.service_rate	promoted	double	9,600
node_o8.q0.service_rate	promoted	double	9,600
node_o9.q0.service_rate	promoted	double	9,600
node_o10.p_1.service_rate	promoted	double	9,600
node_o10.q0.service_rate	promoted	double	9,600
node_o11.p_1.service_rate	promoted	double	9,600
node_o11.q0.service_rate	promoted	double	9,600
node_o12.p_1.service_rate	promoted	double	9,600
node_o12.q0.service_rate	promoted	double	9,600
node_o13.q0.service_rate	promoted	double	9,600
node_o14.q0.service_rate	promoted	double	9,600
node_o15.p_1.service_rate	promoted	double	9,600
node_o15.q0.service_rate	promoted	double	9,600

fixed node **atm_tds16_n.atm_q1_n1**

attribute	*value*	*type*	*default value*
name	atm_q1_n1	string	f
model	aq1b_node0p	enumerated	NONE
x position	-76.7199760256757	double	0.0
y position	45.3907737519817	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_0**

attribute	*value*	*type*	*default value*
name	simplex_0	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n1.pt_0	enumerated	
receiver	atm_q2_n.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024

...
...

delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted;...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node **atm_tds16_n.atm_q1_n2**

attribute	*value*	*type*	*default value*
name	atm_q1_n2	string	f
model	aq1b_node1p	enumerated	NONE
x position	-76.7273125555856	double	0.0
y position	43.2201438459456	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_2**

attribute	*value*	*type*	*default value*
name	simplex_2	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n2.pt_0	enumerated	
receiver	atm_q2_n.pr_10	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

...
...

fixed node **atm_tds16_n.atm_q1_n3**

attribute	*value*	*type*	*default value*
name	atm_q1_n3	string	f
model	aq1b_node2p	enumerated	NONE
x position	-76.7286442906307	double	0.0
y position	40.9748408800897	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_4**

attribute	*value*	*type*	*default value*
name	simplex_4	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n3.pt_0	enumerated	
receiver	atm_q2_n.pr_12	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node **atm_tds16_n.atm_q1_n4**

attribute	*value*	*type*	*default value*
name	atm_q1_n4	string	f
model	aq1b_node3p	enumerated	NONE
x position	-76.7299760256757	double	0.0
y position	38.8542109740536	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled

...
...

priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600

pt-to-pt simplex link atm_tds16_n.simplex_6

attribute	*value*	*type*	*default value*
name	simplex_6	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n4.pt_0	enumerated	
receiver	atm_q2_n.pr_14	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node atm_tds16_n.atm_q2_n

attribute	*value*	*type*	*default value*
name	atm_q2_n	string	f
model	atmq2_nodep	enumerated	NONE
x position	-69.8147210084684	double	0.0
y position	38.4273365299099	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
qms_16.num_servers	promoted	integer	1
qms_16.service_rate	promoted	double	9,600
qms_16.strict_fifo	promoted	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_33

attribute	*value*	*type*	*default value*
name	simplex_33	string	ls

...
...

model	Atd16_ql	enumerated	NONE
transmitter	atm_q2_n.p_rt	enumerated	
receiver	atm_rt_n.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_34

attribute	*value*	*type*	*default value*
name	simplex_34	string	ls
model	Atd16_ql	enumerated	NONE
transmitter	atm_q2_n.p_bus	enumerated	
receiver	atm_rt_n.pr_1	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node atm_tds16_n.atm_rt_n

attribute	*value*	*type*	*default value*
name	atm_rt_n	string	f
model	atm_rt16_nodep	enumerated	NONE
x position	-63.86	double	0.0
y position	38.31	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled

...
...

priority	0	integer	0
user id	0	integer	0
p_2.service_rate	promoted	double	9,600

pt-to-pt simplex link atm_tds16_n.simplex_17

attribute	*value*	*type*	*default value*
name	simplex_17	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_0	enumerated	
receiver	node_o0.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_18

attribute	*value*	*type*	*default value*
name	simplex_18	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_1	enumerated	
receiver	node_o1.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_19

attribute	*value*	*type*	*default value*
name	simplex_19	string	ls

...
...

model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_10	enumerated	
receiver	node_o2.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_20**

attribute	*value*	*type*	*default value*
name	simplex_20	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_11	enumerated	
receiver	node_o3.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_21**

attribute	*value*	*type*	*default value*
name	simplex_21	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_12	enumerated	
receiver	node_o4.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0

...
...

data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_22**

attribute	*value*	*type*	*default value*
name	simplex_22	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_13	enumerated	
receiver	node_o5.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_23**

attribute	*value*	*type*	*default value*
name	simplex_23	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_8	enumerated	
receiver	node_o6.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

...
...

pt-to-pt simplex link **atm_tds16_n.simplex_24**

attribute	*value*	*type*	*default value*
name	simplex_24	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_9	enumerated	
receiver	node_o7.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_25**

attribute	*value*	*type*	*default value*
name	simplex_25	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_14	enumerated	
receiver	node_o8.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_26**

attribute	*value*	*type*	*default value*
name	simplex_26	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_15	enumerated	
receiver	node_o9.pr_0	enumerated	

...
...

color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_27

attribute	*value*	*type*	*default value*
name	simplex_27	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_2	enumerated	
receiver	node_o10.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_28

attribute	*value*	*type*	*default value*
name	simplex_28	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_3	enumerated	
receiver	node_o11.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc

...
...

error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_29

attribute	*value*	*type*	*default value*
name	simplex_29	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_4	enumerated	
receiver	node_o12.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_30

attribute	*value*	*type*	*default value*
name	simplex_30	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_5	enumerated	
receiver	node_o13.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

...
...

pt-to-pt simplex link atm_tds16_n.simplex_31

attribute	*value*	*type*	*default value*
name	simplex_31	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_6	enumerated	
receiver	node_o14.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_32

attribute	*value*	*type*	*default value*
name	simplex_32	string	ls
model	Atd16_ol	enumerated	NONE
transmitter	atm_rt_n.xmt_7	enumerated	
receiver	node_o15.pr_0	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	2,500,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node atm_tds16_n.node_o0

attribute	*value*	*type*	*default value*
name	node_o0	string	f
model	a_outp_0p	enumerated	NONE
x position	-57.047432427207	double	0.0
y position	45.1221294613511	double	0.0
threshold	0.0	double	0.0

...
...

icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.service_rate	promoted	double	9,600

fixed node atm_tds16_n.node_o1

attribute	*value*	*type*	*default value*
name	node_o1	string	f
model	a_outp_1p	enumerated	NONE
x position	-53.2252789915316	double	0.0
y position	45.0449128030625	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.service_rate	promoted	double	9,600

fixed node atm_tds16_n.node_o2

attribute	*value*	*type*	*default value*
name	node_o2	string	f
model	a_outp_2p	enumerated	NONE
x position	-57.0081102818917	double	0.0
y position	43.1042828970264	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.num_servers	promoted	integer	1
q0.service_rate	promoted	double	9,600
q0.strict_fifo	promoted	integer	0

fixed node atm_tds16_n.node_o3

attribute	*value*	*type*	*default value*
name	node_o3	string	f
model	a_outp_3p	enumerated	NONE

...
...

x position	-53.1399520513514	double	0.0
y position	42.8370662387379	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.service_rate	promoted	double	9,600

fixed node **atm_tds16_n.atm_q1_n5**

attribute	*value*	*type*	*default value*
name	atm_q1_n5	string	f
model	atm_q1_node5p	enumerated	NONE
x position	-76.69	double	0.0
y position	36.73	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_8**

attribute	*value*	*type*	*default value*
name	simplex_8	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n5.pt_0	enumerated	
receiver	atm_q2_n.pr_2	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

...
...

fixed node atm_tds16_n.atm_q1_n6

attribute	*value*	*type*	*default value*
name	atm_q1_n6	string	f
model	atm_q1_node6p	enumerated	NONE
x position	-76.69	double	0.0
y position	34.64	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_10

attribute	*value*	*type*	*default value*
name	simplex_10	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n6.pt_0	enumerated	
receiver	atm_q2_n.pr_4	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node atm_tds16_n.atm_q1_n7

attribute	*value*	*type*	*default value*
name	atm_q1_n7	string	f
model	atm_q1_node7p	enumerated	NONE
x position	-76.69	double	0.0
y position	32.55	double	0.0
threshold	0.0	double	0.0

...
...

icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_12**

attribute	*value*	*type*	*default value*
name	simplex_12	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n7.pt_0	enumerated	
receiver	atm_q2_n.pr_6	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node **atm_tds16_n.atm_q1_n8**

attribute	*value*	*type*	*default value*
name	atm_q1_n8	string	f
model	atm_q1_node8p	enumerated	NONE
x position	-76.73	double	0.0
y position	30.5	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

...
...

pt-to-pt simplex link **atm_tds16_n.simplex_14**

attribute	*value*	*type*	*default value*
name	simplex_14	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n8.pt_0	enumerated	
receiver	atm_q2_n.pr_8	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node **atm_tds16_n.atm_q1_n9**

attribute	*value*	*type*	*default value*
name	atm_q1_n9	string	f
model	atm_q1_node9p	enumerated	NONE
x position	-80.33	double	0.0
y position	45.31	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_1**

attribute	*value*	*type*	*default value*
name	simplex_1	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n9.pt_0	enumerated	
receiver	atm_q2_n.pr_1	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1

...
...

condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node **atm_tds16_n.atm_q1_n10**

attribute	*value*	*type*	*default value*
name	atm_q1_n10	string	f
model	atm_q1_node10p	enumerated	NONE
x position	-80.21	double	0.0
y position	43.34	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_3**

attribute	*value*	*type*	*default value*
name	simplex_3	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n10.pt_0	enumerated	
receiver	atm_q2_n.pr_11	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

...
...

fixed node **atm_tds16_n.atm_q1_n11**

attribute	*value*	*type*	*default value*
name	atm_q1_n11	string	f
model	atm_q1_node11p	enumerated	NONE
x position	-80.17	double	0.0
y position	41.13	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_5**

attribute	*value*	*type*	*default value*
name	simplex_5	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n11.pt_0	enumerated	
receiver	atm_q2_n.pr_13	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node **atm_tds16_n.atm_q1_n12**

attribute	*value*	*type*	*default value*
name	atm_q1_n12	string	f
model	atm_q1_node12p	enumerated	NONE
x position	-80.17	double	0.0
y position	38.81	double	0.0
threshold	0.0	double	0.0

fashion where a fixed time-slot allocation scheme can be used. The ring capacity is required to be the sum of the capacities of all input links. Its time-slot allocation could be adjusted. The ring structure has the advantage over the bus structure in that a time-slot can be used several times within one rotation. When using this destination release mechanism, an effective utilization of more than 100% can be achieved [7] .

The switching networks can be classification as follows:
1) Shared Backplane, where switch ports have access to a backplane whose bandwidth is shared amongst them . The sharing can be performed by two alternative techniques. The throughout is small. Suit for use in local area.
2) Time division switching, a path of fixed bandwidth is set up during call initiation. The data are transmitted using time-division multiplexed frames, a path is formed by allocating slots into the trunks. Each of the switching nodes maps the incoming slots into the corresponding outgoing slots , using the routing information stored at the node.
3) Self-routing switching fabrics which the of Banyan fabric is an example (See Fig. 8).

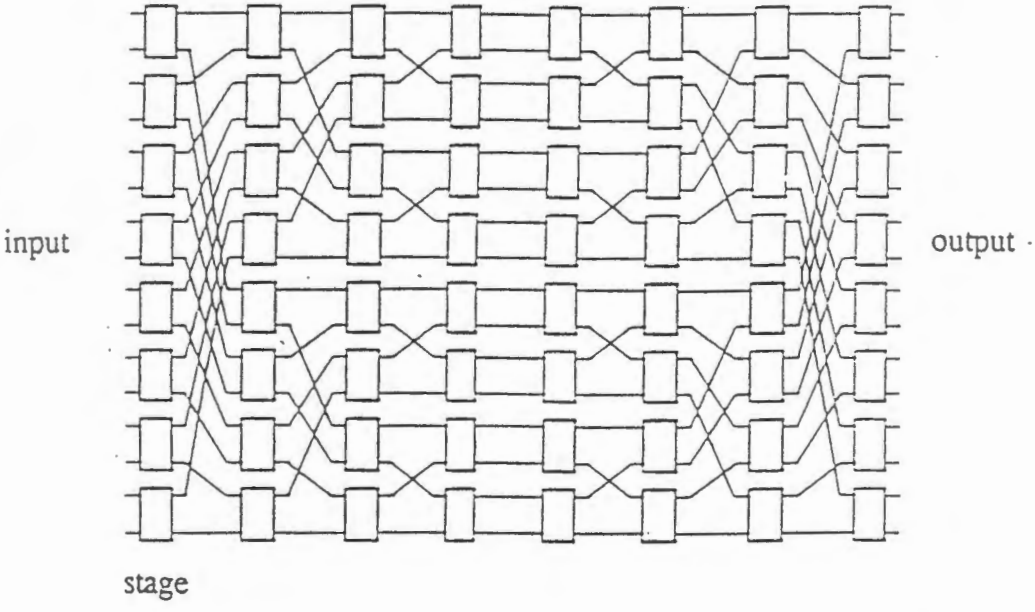


Figure 8. Example of a multi-path interconnection Banyan network

The purpose of this design is to achieve high speed switching and high throughput in hardware. Its delay is limited only by the speed of electrical signals through the semiconductor material. The output of an ATM cell is determined by its virtual channel and virtual path identifiers. There are many type of architectures for Banyan networks [8], for example multi-path interconnection network, Parallel Banyan network, etc. They could be used in local area network and wide area network.

According to the basic structures of the switching element and switching network, when the the local ATM switch is designed, the following features need to be addressed: best-effort service; burst buffering, Congestion control, delay Mechanisms, access flow control, multicast.
Therefore , the objectives of the design for the ATM switch will focus on a simple

support TCP/IP (Transmission Control protocol/ Internet Protocol) and OSI (Open System Interconnection) protocols, as well as ATM Adaptation Layers (AAL 3/4 and 5). [6]
The local ATM switch provides the virtual connection management, performed mainly in software, and cell routing implemented entirely in hardware.
System software is an essential component of a practical ATM LAN. The software would include the ATM switching control software ; implementing adaptation layer function and the host side of the signalling protocol by the device driver (software); the application programming interface is used for custom ATM application software; the network management and about network usage software.

ATM Switch

The ATM switching use the fast packet switching technique providing high throughput better bandwidth efficiency for bursty traffic, inherent rate adaption. The packet switching is based on the separation of signalling and data transfer capabilities.
The aggregate throughput of this switch will be in the Gbit/s range and the cross-node delay as well as the cell loss should be kept very low. The switch fabrics with highly parallel architectures can fulfil these stringent requirements.

A switching element has an input control (IC) for each incoming line and an output control (OC) for each outgoing line. In order to avoid excessive cell loss in the case of internal collisions, buffers have to be provide within the switching element.
Incoming cells will be synchronized to the internal clock by IC . The OC transports cells which will go towards destination from the interconnection network. ICs and OCs are coupled by the interconnection network. (See Fig. 6)

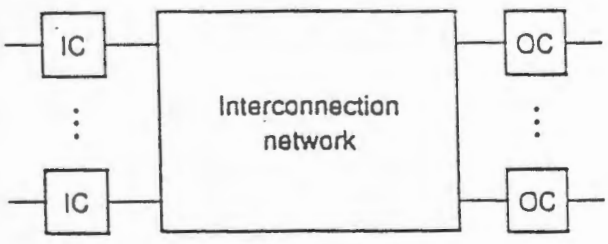


Figure 6. Model of a Switching Element

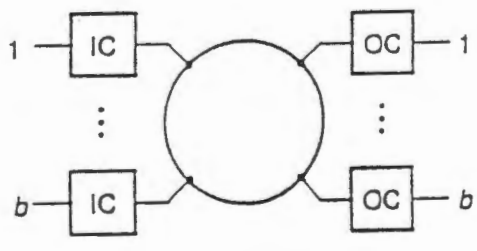


Figure 7. Ring -type Switching Element

For matrix-type switching elements, an internally non-blocking switch element will be achieved by using a rectangular matrix of cross-points for the interconnection network. Within this switching element, the buffer can be placed in the input, output or in the cross-point.
For a central memory switching element, the common memory can be organized to provide logical input as well as logical output buffers, and all buffers share one common memory.
For a bus-type switching element, the interconnection network can be realized by a high speed time division multi (TDM). On the bus system bit-parallel data transmission is required with buffers at the output controller.
For a ring-type switching element (See Fig. 7), the ring network should be operated in slotted

architecture, high reliability, low delay, low possibility congestion, non-blocking performance, higher speed processing.
In those basic type of switching fabric, the Ring-type switching element adopt the method of Time Division. Its principal operation is non-blocking performance, all input/output ports can transmit at their highest rate without any chance of blocking or congestion within the switching fabric. For a 16 ports switch, each port occupies a time slot of 1/16 of the rate in the fabric. (If the rate is 2.5 Gbits/sec in the fabric , each port would get a time slot of 155 Mbits/sec .)
Its capacity is greater than or equal to the sum of the individual input/output ports, so that there's no contention at the port level. This type of switching is a single stage switch which minimal delay associated with cell transit, therefore its delay is very low, no traffic is ever delayed or needs to wait before entering the traffic circle.
In this architecture, all of the input/output traffic can easily be accommodated.
This simple architecture has easy management and control, therefore it would be predicted that it would be the one to be adopted.

Conclusion

The ATM technology is currently being applied to local and campus area networking where it offers increased bandwidth, supporting broadband services.
Although local ATM switching have been designed none as yet has been universally acknowledged.
The Ring-type time division switching architecture provides non-blocking performance, low transit delay, flexible ports speeds and high network reliability, it could be used in local area ATM network.
Its detailed hardware and software will be researched and analyzed in another paper latter.

Acknowledgement

I am grateful to Telkom S.A. for supporting this project and to my supervisor M. Ventura.

References

[1] M.De Prycker "ATM Solution for B-ISND" Second Edition 1993
[2] CCITT Recommendation I.321 "Reference model and its application"
[3] CCITT Recommendation I.361 "B-ISND ATM Layer Specification"
[4] J. Dupraz and M. De Prycker "Principles and Benefits of the ATM" Electrical Communication. Vol 64 No. 2/3 1990.
[5] Peter Newman "ATM Local Area Network" IEEE Communication March 1994.
[6] E. Biagioni, E.Cooper and R. Sansom "Designing a Practical ATM LAN" IEEE Network March 1993.
[7] R.Handel and M.N. Huber "Integrated Broadband Network An introduction to ATM-Based Networks" Chapter 6.
[8] David Wright "Broadband:Business Services, Technologies, and Strategic Impact".

* Note: This paper has been published in "Teletraffic -- Engineering Symposim" at 22nd of Aug. 1994.

Appendix B

Time Division ATM Switching

Lianghong Yu (Joy Yu)
Department of Electrical Engineering
University of Cape Town

Abstract

In this paper, a high speed Time Division Switching architecture will be described to satisfy Asynchronous Transfer Mode (ATM) requirements. This architecture is essentially based on an original method of sequentially scheduling the inputs and parallel transfer. A time-slot allocation scheme is used with the capacity of the structure of the switching fabric equal to the sum of the capacities of all input links, and the transmission is provided through the medium. This system is designed as the output buffer, available to receive information, and has a cell loss probability which is nearly non existence. In the last part, the switch control will be briefly described.

Introduction

The switching fabric is responsible for the transport of cells within the ATM switch. Its function is to route cells through the fabric from the origination port to the appropriate destination port. The switching fabric operation is independent of the cell contents and type of traffic. Its purpose is simply to transfer all packets from their input lines to their output lines without loss, with a short delay and a good output line efficiency.

In the last paper [1] , the ATM fast packet switches were analyzed, and an interesting plan of design was proposed. Various ATM switch architectures have been proposed, following within three main structures [2]: 1) Shared memory, 2) Shared medium and 3) Space-division. The design proposed makes use of shared medium ATM switch architecture, and it deals with the shared memory as well.

This switching architecture can be thought of as a "train" in cyclical motion. It will take the "customers" (424 bits for a cell) from the input port, and guided the customers to the properly output port at high speed. The moving "train" can be perfectly timed so that no traffic is ever delayed or needs to wait before entering the "train". The "train" guides the troops of customers sequentially. It is particularly fast, without congestion or collision. A study of its architecture is presented in this paper.

Time Division Switching Architecture

The design idea to the kind of switching architecture is simple, and the structure could be divided into three main parts. The input module is transferred in parallel and sequentially into the "train". The information on the "train" is switched by the routing table and queued in the output module (Fig. 1.).

All these modules need to be synchronized through a clock. Its distributor can reduce or increase the clock frequency. The error address box collects the cells for the error address.

Its performance process and architecture are as follows:

The each input port gets its time slot of 155Mb/s. The time slot (the time period)

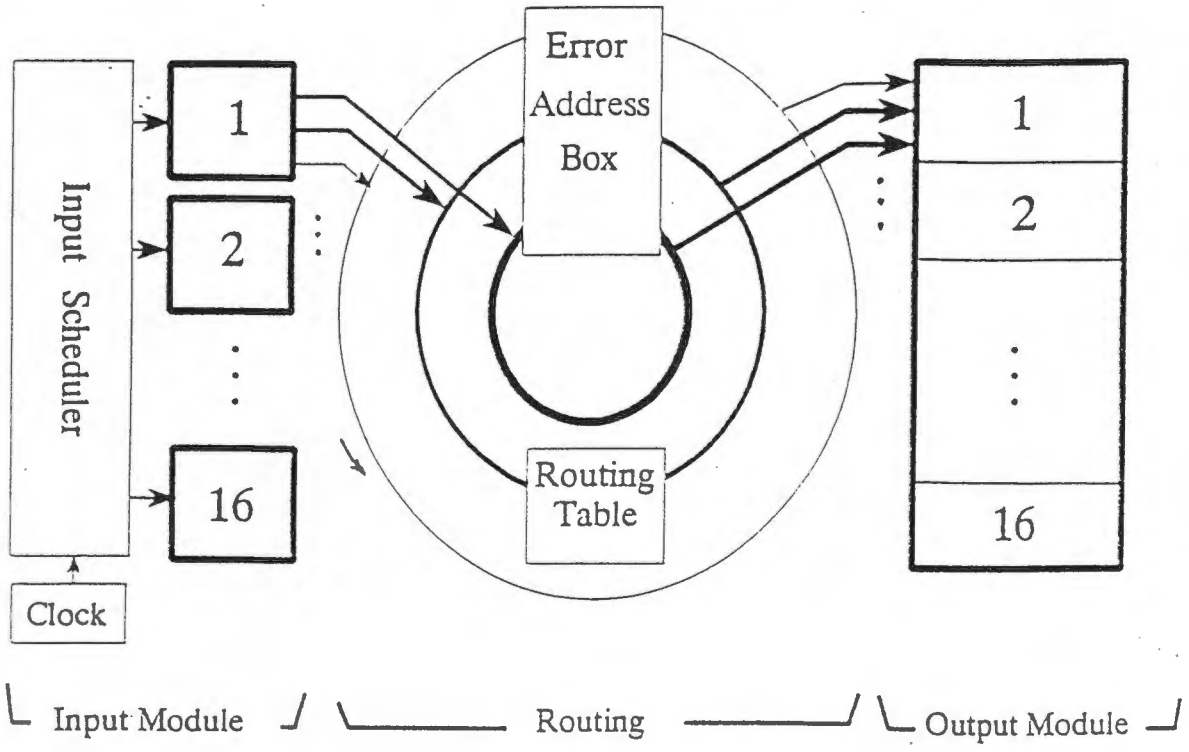


Fig. 1. TDS switching fabric architecture

1. Input Module

The input module is composed of an input scheduler and the input queue module. The capacity of each input line is 155Mb/s [3], and it is designed to support 16 input port.

The input scheduler is the quite important part of the Time Division Switching (TDS) switch element. It introduces the principle of the sequential handling of information present on the 16 input lines. When the input buffers are filled with the cell (the each cell is 424 bits), it would provide the time slot for a predetermined time period to control scanning 16 input buffers sequentially, so that these cells can be loaded one by one onto the "train".

The capacity of TDS switching fabric is required to be equal to the sum of the capacities of all input links, so it is 2.5 Gb/s (16 times 155Mb/s).

is

$$Ti = \frac{424 \text{ bits}}{155 \text{ Mbits/s}} = 2.74 \ \mu s.$$

Then the motion time (the occupied time) of every port is

$$ti = \frac{2.74 \ \mu s}{16} = 170 \ ns$$

Therefore control pulse frequency

$$f = 1 / 170 \ ns = 5.9 \text{ MHZ}$$

would be provided by the clock. The scheduler would be a sequential scanning to 16 input buffers, of course the scanning time period is 170 *ns* to every buffer (Fig. 2.).

In the input queue module, there are two layers Register1 (R1) and Register2 (R2), which are from a same input port. The each buffer can store 53 bytes (424 bits) of information plus 1 bit indicating whether the buffer is empty or full (for present

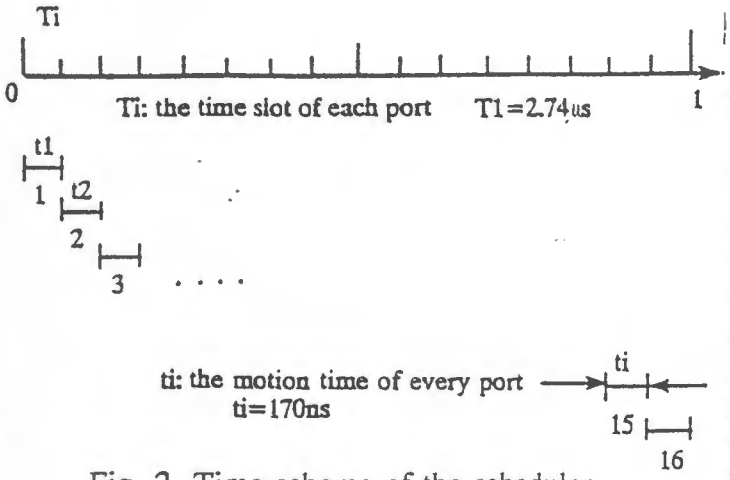


Fig. 2. Time scheme of the scheduler

information, called PI. PI=1, indicating that buffer is full). When a buffer is filled and receive the signal of setting out, the 425 tristate of the module get into the "train" immediately with the parallel fashion (Fig. 3.).

forward.

2. Routing

This part is the core of Time Division ATM Switching, its function is translating VCI/VPI of the cell, routing the cell to the appropriate output port and giving a "new" address [4], which contains the contents for the original address (Fig. 4.). Because the TDS Switching fabric is a kind of single stage structure, its transferring with bits-parallel is very fast.

The switching fabric includes 425 wires, which are 1 wire for PI, 40 (5 bytes head of the cell) for Address wires, and 384 (48 bytes for the payload) for Data wires. Their capacity is 2.5 Gb/s, and they provide the Routing Table through the

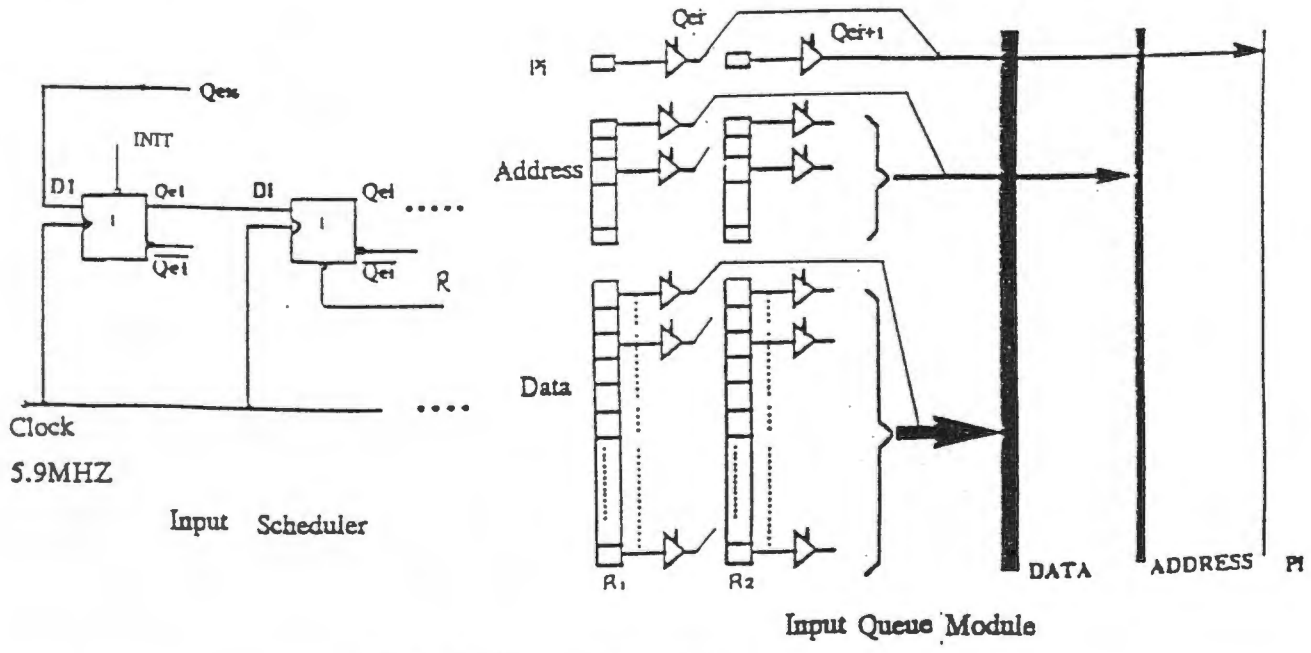


Fig. 3. The input module

The two buffer layers carry out the performance alternately so that the waiting time for the incoming cells can be reduced loss and waiting time. The buffers must be to accept the incoming bits-series, and send out those bits in parallel. Then the input scheduler will be moved one step

Address wires (Fig. 5.).

The basic hardware of the routing table is the random access memory (RAM) [5], [8]. Since the parameters of the different calling VP/VC could be often changed, and its speed is very fast (The commercial

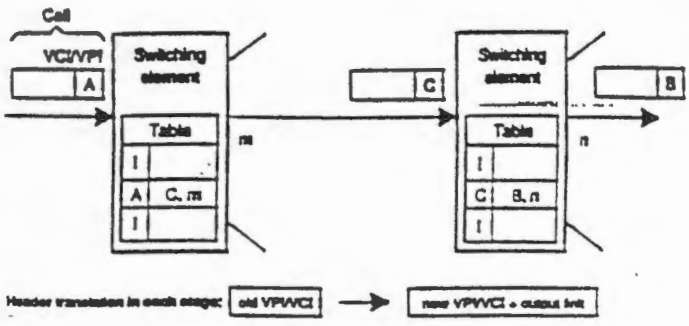


Fig.4. The routing table switching elements

(connected to the Error Address Box.) for the wrong address (VPI/VCI) of the cell from the output of the RAM. (Actually, HEC, PTI and CLP bits in the head of the cell [7] are not changed through the Routing Table). However, after the cell passes the "train", the Address wires load "new" address contents and Data wires load the data, which then is stored directly in the selected output buffer(s).

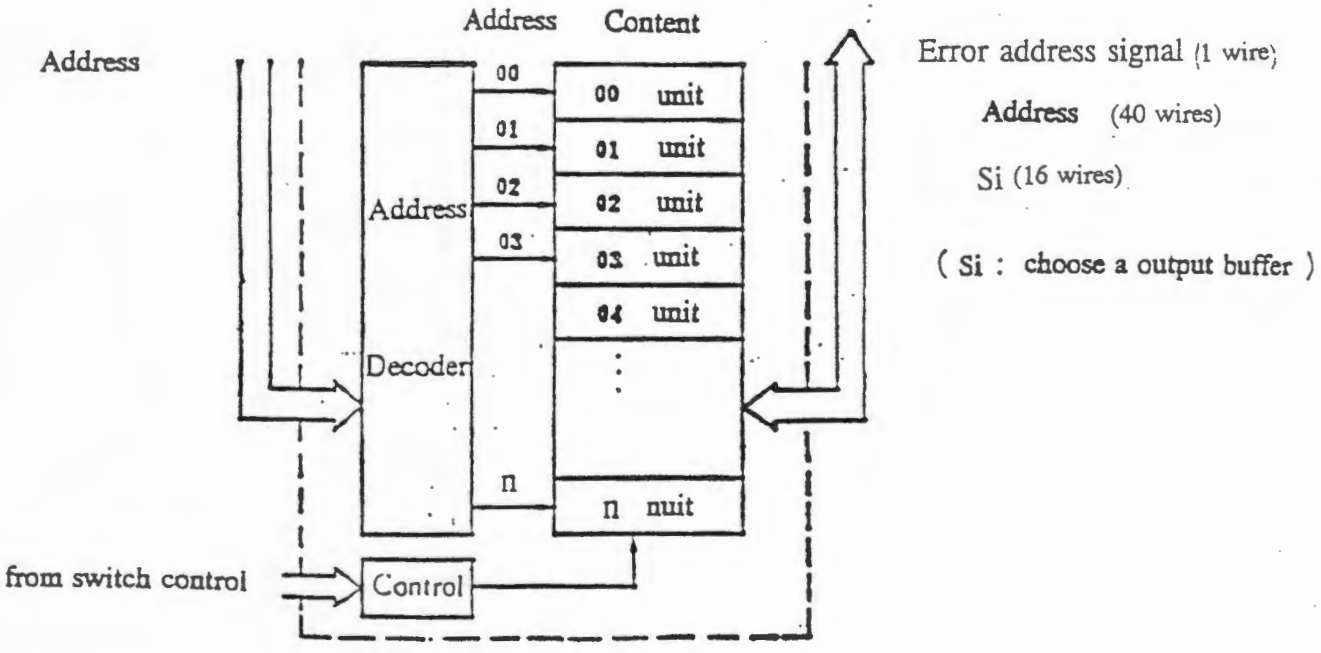


Fig. 5. Routing Table structure

memories only about 10 *ns* -- 70 *ns* access time [6]). The RAM is predetermined in that all input addresses contain all output contents, which include the next address(es) contents and the signal(s) for activating the correct output buffer(s) to this address(es).

As the input address, 40 address wires in parallel are connected to the RAM, there would be other 40 address wires, 16 wires for the activating signals of the output buffers, and adding 1 wire of the signal

3. Output Module

The output module is composed of a number of (a group) output buffers which, are bits in parallel coming in , bits in serial going out. Its capacity will change from 2.5 Gb/s to 155 Mb/s.

The output module extracts the information addressed to it. This is shown in Fig. 6. The buffer could be loaded in by Address (40 bits) and Data (384 bits) plus the PI 1 bit . When it is acknowledged all the

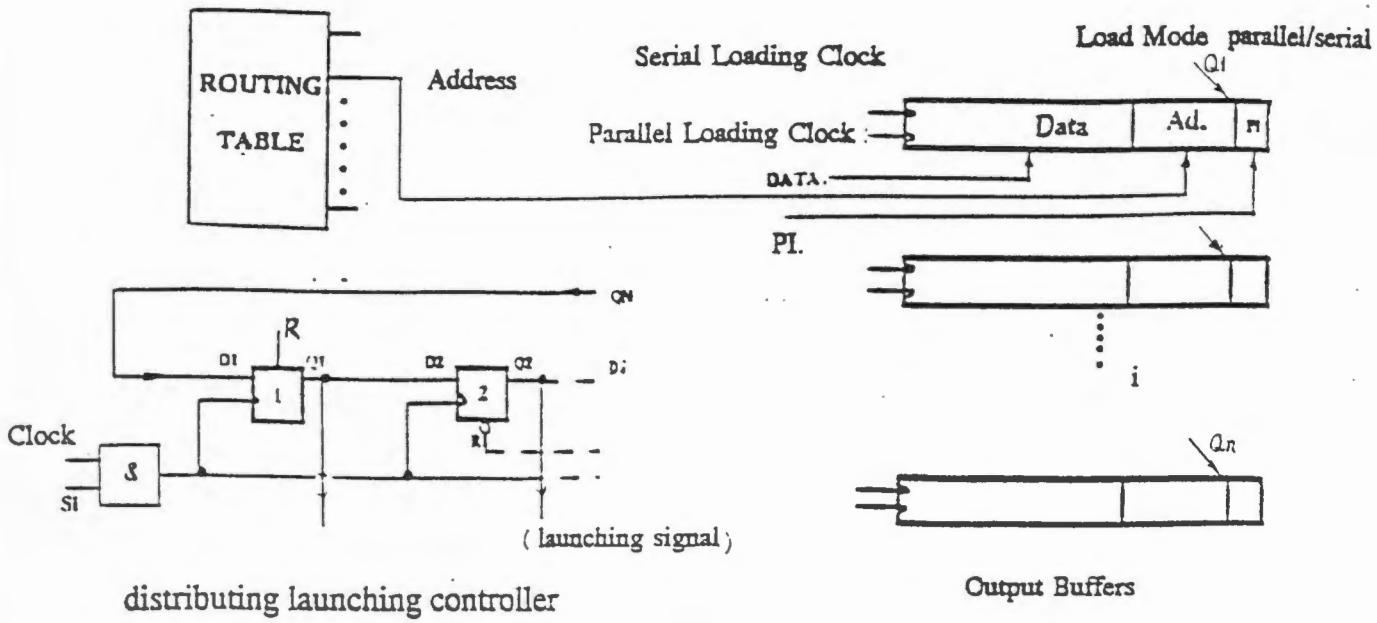


Fig. 6. The output module

425 bits to come in, the cell is registered to store to the output buffer. Up to the present, the transferring process of a cell has been implemented in the TDS switching fabric, it spent time for 170 *ns*. The parallel loading clock is the receiving frequency of the output buffer, which is 1/2.5Gb/s.

If a number of the cells of input ports come to the same output port , the output module will put in the cells to different buffers , but which are a same output port. So an output module is composed of a group of output buffers, the best for all the output buffers would be considered being 4096 (16 · 16 · 16=4096) cells.

In the each output module, the clock distributing launching controller [9] is set up as well, so that one output port is able to send out the cells sequentially. The output module will push out the cell in serial bits. This serial launching clock uses the 155Mb/s as the speed.

4. Feature

The TDS switching fabric structure is simple, it is not like a general Banyan network, being always the same sequence from the input to the output.

Its capacity is quite high, cell transferring is not in series but in parallel, it being only one stage routing. This can save time with minimal delay.

This architecture is able to avoid the congestion problem without any chance of blocking in the switching fabric. The cells are transmitted one by one, in fact a synchronous mechanism is used.

The port speed of this architecture can be chosen appropriately, since a port entering the switching fabric can use a variable number of time slots, so that the interface speed is flexible, for example, 622 Mb/s, 442 Mb/s....

According to the technology level at present, the ports will still be increased to about 64 ports, because the used main hardware, like the buffers, RAM, and

other electronic circuits, have the fast enough speed of the performance.

This system is designed to have quite large container output buffers for 4096 cells, therefore a cell loss probability is very small.

Switching Control

The switching control plays a very important role in the switch. In the switching control architecture of Time Division ATM switching, the distributed type architecture would be used. Because it has three major benefits: quick connection setup, disaster recovery and architecture scalability. A control processor (CPU) is installed in the switch, and runs UNIX operating system, which is in itself superior and at present a number of produces also use it [8].

The control processor is executing the software that is responsible for establishing virtual connections across the switch, such as the call setup, call tear-down, and managing software for the switch for instance local switch monitoring. It would also like to observe the ATM protocol .

Conclusion

The architecture of Time Division Switching for Local area ATM switching has been described in this paper. It is adapted time division, that is a shared medium structure . It has 16 input ports, 2.5 Gb/s capacity in the switching fabric and one cell takes 170 *ns* to travel through the switching fabric. Its performance is based on the scheduling the input sequentially and transferring it in parallel.

The advantages of this architecture are: 1) No congestions and contentions, 2) Predictable low delay, 3) Flexible interface speed and 4) Predictable low cell loss.

With the present technology level, this architecture of the switching fabric can be evolved to increase the capacity and the ports. Therefore, it is considered being a valuable research in local area ATM Switching field.

Acknowledgment

I am grateful to Telkom S.A. for supporting This project, to Prof. C. Fayet, Mr. B. Peterssen, Mr. L. Rheeder, and my supervisor Prof. M. Ventura for their help.

References

[1] Joy Yu "ATM Local Area Network" Teletraffic-Engineering Symposim Aug. 1994
[2] Thomas M.Chen, Stephen S.Lin "ATM Switching Systems" Chapter 8, 1994
[3] CCITT Recommendation I.211 "B-ISDN Service Aspects"
[4] R.Handel and M.N.Huber "Integrated Broadband Network An Introduction to ATM-Base Networks" Chapter 6, 1993
[5] J.G-Haro and A.Jajaszczyk "ATM Shared-Memory Switching Architectures" IEEE Communication Jul./Aug. 1994
[6] "Memory Components Handbook" Intel
[7] CCITT Recommendation I.361 "B-ISDN ATM Layer Specification"
[8] C.fayet, A.Jacques, G.Pujolle "High speed switching for ATM : the BSS" Computer Networks and ISDN System 26(1994) 1225-1234
[9] E.Biagioni, E.Cooper and R.Sansom "Designing a Practical ATM LAN" IEEE Network March 1993

Appendix C

Cell Header Processing in Switching Fabric

The two main tasks of ATM switching nodes are VPI/VCI translation, and transport of cells from the input to the appropriate output. The Table-control principle can be applied:

Table Switching Elements:

The principle the VPI/ICI of the cell header will be translated in each switching element into a new one. During the connection setup phase the contents of the tables are updated. Each table entry consists of the new VPI/VCI and the number of the appropriate output. The figure shows the header processing in a switching network which consists of table switching elements.

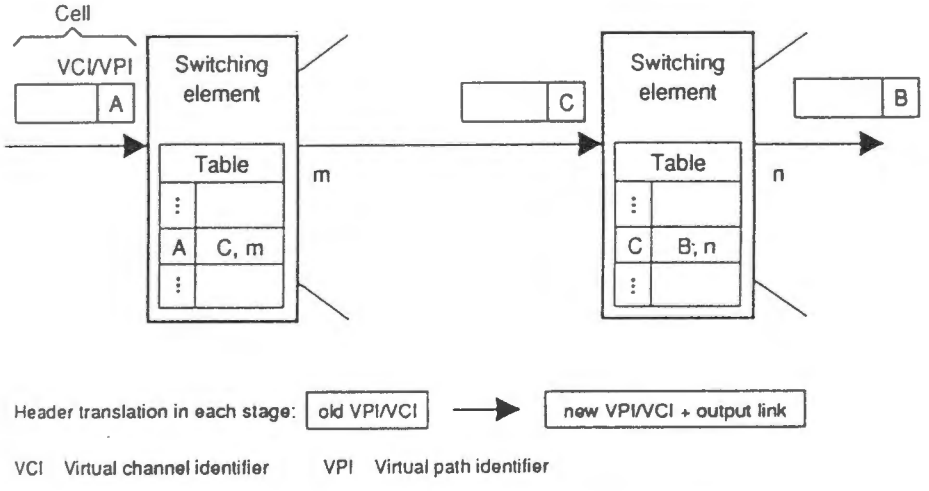


Figure. Table switching element

...
...

icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_7

attribute	*value*	*type*	*default value*
name	simplex_7	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n12.pt_0	enumerated	
receiver	atm_q2_n.pr_15	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node atm_tds16_n.atm_q1_n13

attribute	*value*	*type*	*default value*
name	atm_q1_n13	string	f
model	atm_q1_node13p	enumerated	NONE
x position	-80.13	double	0.0
y position	36.49	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

...
...

pt-to-pt simplex link atm_tds16_n.simplex_9

attribute	*value*	*type*	*default value*
name	simplex_9	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n13.pt_0	enumerated	
receiver	atm_q2_n.pr_3	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node atm_tds16_n.atm_q1_n14

attribute	*value*	*type*	*default value*
name	atm_q1_n14	string	f
model	atm_q1_node14p	enumerated	NONE
x position	-80.13	double	0.0
y position	34.33	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_11

attribute	*value*	*type*	*default value*
name	simplex_11	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n14.pt_0	enumerated	
receiver	atm_q2_n.pr_5	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1

...
...

condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node atm_tds16_n.atm_q1_n15

attribute	*value*	*type*	*default value*
name	atm_q1_n15	string	f
model	atm_q1_node15p	enumerated	NONE
x position	-80.1	double	0.0
y position	32.05	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link atm_tds16_n.simplex_13

attribute	*value*	*type*	*default value*
name	simplex_13	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n15.pt_0	enumerated	
receiver	atm_q2_n.pr_7	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

...
...

fixed node **atm_tds16_n.atm_q1_n16**

attribute	*value*	*type*	*default value*
name	atm_q1_n16	string	f
model	atm_q1_node16p	enumerated	NONE
x position	-80.06	double	0.0
y position	29.84	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
source interarrival args	promoted	string list	1.0
user id	0	integer	0
q_0.num_servers	promoted	integer	1
q_0.service_rate	promoted	double	9,600
q_0.strict_fifo	promoted	integer	0

pt-to-pt simplex link **atm_tds16_n.simplex_15**

attribute	*value*	*type*	*default value*
name	simplex_15	string	ls
model	Atd16_il	enumerated	NONE
transmitter	atm_q1_n16.pt_0	enumerated	
receiver	atm_q2_n.pr_9	enumerated	
color	RGB233	color	RGB233
ber	0.0	double	0.0
channel count	1	integer	1
condition	enabled	toggle	enabled
cost	0.0	double	0.0
data rate	155,000,000	double	1,024
delay	0.0	double	0.0
ecc model	dpt_ecc	typed file	dpt_ecc
error model	dpt_error	typed file	dpt_error
packet formats	acellp_pk	string	all formatted,...
propdel model	dpt_propdel	typed file	dpt_propdel
txdel model	dpt_txdel	typed file	dpt_txdel
user id	0	integer	0

fixed node **atm_tds16_n.node_o4**

attribute	*value*	*type*	*default value*
name	node_o4	string	f
model	a_outp_4p	enumerated	NONE
x position	-56.9326874444144	double	0.0
y position	40.7901917945942	double	0.0
threshold	0.0	double	0.0

...
...

icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.service_rate	promoted	double	9,600

fixed node atm_tds16_n.node_o5

attribute	*value*	*type*	*default value*
name	node_o5	string	f
model	a_outp_5p	enumerated	NONE
x position	-53.100534008739	double	0.0
y position	40.6682781021616	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.service_rate	promoted	double	9,600

fixed node atm_tds16_n.node_o6

attribute	*value*	*type*	*default value*
name	node_o6	string	f
model	a_outp_6p	enumerated	NONE
x position	-56.8960287691891	double	0.0
y position	38.6189559568462	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.service_rate	promoted	double	9,600

fixed node atm_tds16_n.node_o7

attribute	*value*	*type*	*default value*
name	node_o7	string	f
model	a_outp_7p	enumerated	NONE
x position	-53.1612118634237	double	0.0
y position	38.5097057345037	double	0.0

...
...

threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.service_rate	promoted	double	9,600

fixed node **atm_tds16_n.node_o8**

attribute	*value*	*type*	*default value*
name	node_o8	string	f
model	a_outp_8p	enumerated	NONE
x position	-56.814163025405	double	0.0
y position	36.5736355902711	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.service_rate	promoted	double	9,600

fixed node **atm_tds16_n.node_o9**

attribute	*value*	*type*	*default value*
name	node_o9	string	f
model	a_outp_9p	enumerated	NONE
x position	-53.1040191794594	double	0.0
y position	36.4217218978384	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
q0.service_rate	promoted	double	9,600

fixed node **atm_tds16_n.node_o10**

attribute	*value*	*type*	*default value*
name	node_o10	string	f
model	a_outp_10p	enumerated	NONE
x position	-56.8608456749546	double	0.0
y position	34.328404547388	double	0.0

...
...

threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.service_rate	promoted	double	9,600

fixed node **atm_tds16_n.node_o11**

attribute	*value*	*type*	*default value*
name	node_o11	string	f
model	a_outp_11p	enumerated	NONE
x position	-53.1846970341441	double	0.0
y position	34.178500444685	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.service_rate	promoted	double	9,600

fixed node **atm_tds16_n.node_o12**

attribute	*value*	*type*	*default value*
name	node_o12	string	f
model	a_outp_12p	enumerated	NONE
x position	-56.7755187347744	double	0.0
y position	32.2058609489192	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.service_rate	promoted	double	9,600

fixed node **atm_tds16_n.node_o13**

attribute	*value*	*type*	*default value*
name	node_o13	string	f
model	a_outp_13p	enumerated	NONE
x position	-53.1467066238738	double	0.0

...
...

y position	32.1626155214416	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
q0.service_rate	promoted	double	9,600

fixed node atm_tds16_n.node_o14

attribute	*value*	*type*	*default value*
name	node_o14	string	f
model	a_outp_14p	enumerated	NONE
x position	-56.736196589459	double	0.0
y position	30.1959568462163	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
q0.service_rate	promoted	double	9,600

fixed node atm_tds16_n.node_o15

attribute	*value*	*type*	*default value*
name	node_o15	string	f
model	a_outp_15p	enumerated	NONE
x position	-53.1073844785584	double	0.0
y position	30.1487162136035	double	0.0
threshold	0.0	double	0.0
icon name	fixed_comm	icon	fixed_comm
altitude	0.0	double	0.0
condition	enabled	toggle	enabled
priority	0	integer	0
user id	0	integer	0
p_1.service_rate	promoted	double	9,600
q0.service_rate	promoted	double	9,600

Appendix F

atm_q1_node9 Node Model Report

Node Model Comments

Node Model Keywords

a_im9

Node Model Types

Node Type	*Supported*	*Default Icon*
fixed	YES	fixed_comm
mobile	NO	
satellite	NO	

Node Model Interface Attributes

Attribute **altitude** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	0.0	N/A
Default Value:	0.0	YES
Data Type:	double	N/A
Attribute Description:	Private	N/A
Comments:	This attribute represents the distance of a satellite from sea-level on the surface of the earth.	YES
Symbol Map:	NONE	YES

Attribute **condition** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	enabled	N/A
Default Value:	enabled	YES
Data Type:	toggle	N/A
Attribute Description:	Private	N/A
Comments:	This attribute represents the operational status of a node	YES
Symbol Map:	NONE	YES

Attribute **phase** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	0.0	N/A
Default Value:	0.0	YES
Data Type:	double	N/A
Attribute Description:	Private	N/A

...
...

Units:	sec.	YES
Comments:		YES
	This attribute supports the translation in time of a satellite's orbit.	
Symbol Map:	NONE	YES

Attribute **priority** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	0	N/A
Default Value:	0	YES
Data Type:	integer	N/A
Attribute Description:	Private	N/A
Low Range:	0 inclusive	YES
High Range:	32000 inclusive	YES
Comments:		YES
	This attribute represents a node's priority. When multiple events are scheduled for the same simulation time, those for a higher priority node will be executed first.	
Symbol Map:	NONE	YES

Attribute **q_0.num_servers** properties

Property	*Value*	*Inherit*
Assign Status:	promoted	
Default Value:	1	YES
Data Type:	integer	N/A
Attribute Description:	Private	N/A
Symbol Map:	NONE	YES

Attribute **q_0.service_rate** properties

Property	*Value*	*Inherit*
Assign Status:	promoted	
Default Value:	9,600	YES
Data Type:	double	N/A
Attribute Description:	Private	N/A
Units:	bits/sec	YES
Symbol Map:	NONE	YES

Attribute **q_0.strict_fifo** properties

Property	*Value*	*Inherit*
Assign Status:	promoted	
Default Value:	disabled	YES
Data Type:	toggle	N/A
Attribute Description:	Private	N/A
Symbol Map:	NONE	YES

Attribute **source interarrival args** properties

Property	*Value*	*Inherit*
Old Names:		N/A
	src.interarrival args	

...
...

Assign Status:	promoted	
Default Value:	1.0	YES
Data Type:	string list	N/A
Attribute Description:	Private	N/A
Comments:	This attribute is a list of numerical parameters which complete the specification for computation of packet interarrival time.	YES
Symbol Map:	NONE	YES

Attribute **user id** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	0	N/A
Default Value:	0	YES
Data Type:	integer	N/A
Attribute Description:	Private	N/A
Comments:	This attribute is an integer which is provided for user-specific identification purposes.	YES
Symbol Map:	NONE	YES

processor **p_0**

attribute	*value*	*type*	*default value*
name	p_0	string	p
process model	atm_setb	typed file	sink
icon name	processor	icon	processor
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
intrpt interval	disabled	toggle double	disabled
priority	0	integer	0
recovery intrpts	disabled	enumerated	disabled
super priority	disabled	toggle	disabled

packet stream **p_0 [src stream [0]] -> q_0 [dest stream [0]]**

attribute	*value*	*type*	*default value*
name	strm_1	string	strm
src stream	src stream [0]	enumerated	src stream [0]
dest stream	dest stream [0]	enumerated	dest stream [0]
intrpt method	scheduled	integer	scheduled
delay	0.0	double	0.0
color	RGB030	color	RGB030

...
...

processor p_1

attribute	value	type	default value
name	p_1	string	p
process model	atmq1_count16_9	typed file	sink
icon name	processor	icon	processor
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
intrpt interval	disabled	toggle double	disabled
priority	0	integer	0
recovery intrpts	disabled	enumerated	disabled
super priority	disabled	toggle	disabled

packet stream p_1 [src stream [0]] -> pt_0 [dest stream [0]]

attribute	value	type	default value
name	strm_3	string	strm
src stream	src stream [0]	enumerated	src stream [0]
dest stream	dest stream [0]	enumerated	dest stream [0]
intrpt method	scheduled	integer	scheduled
delay	0.0	double	0.0
color	RGB030	color	RGB030

queue q_0

attribute	value	type	default value
name	q_0	string	p
process model	atm_acbf_ms	typed file	pc_fifo
subqueue	(See below.)	compound	
icon name	queue	icon	queue
begsim intrpt	enabled	toggle	disabled
endsim intrpt	disabled	toggle	disabled
failure intrpts	disabled	enumerated	disabled
intrpt interval	disabled	toggle double	disabled
num_servers	promoted	integer	1
priority	0	integer	0
recovery intrpts	disabled	enumerated	disabled
service_rate	promoted	double	9,600
strict_fifo	promoted	toggle	disabled
super priority	disabled	toggle	disabled

subqueue q_0.subqueue

attribute	value	type	default value
count	1	integer	0
list	(See below.)	object list	

...
...

list q_0.subqueue [0]

attribute	*value*	*type*	*default value*
bit capacity	infinity	double	infinity
pk capacity	infinity	double	infinity

packet stream q_0 [src stream [0]] -> p_1 [dest stream [0]]

attribute	*value*	*type*	*default value*
name	strm_2	string	strm
src stream	src stream [0]	enumerated	src stream [0]
dest stream	dest stream [0]	enumerated	dest stream [0]
intrpt method	scheduled	integer	scheduled
delay	0.0	double	0.0
color	RGB030	color	RGB030

ideal generator src

attribute	*value*	*type*	*default value*
name	src	string	g
interarrival pdf	exponential	typed file	constant
interarrival args	promoted	string list	1.0
pk size pdf	constant	typed file	constant
pk size args	53	string list	1024.0
field (0) pdf	constant	typed file	constant
field (0) args	0.0	string list	0.0
packet format	ams_atm_cell	typed file	NONE
start time	0.0	double	0.0
stop time	infinity	double	infinity
icon name	ideal_gen	icon	ideal_gen

packet stream src [src stream [0]] -> p_0 [dest stream [0]]

attribute	*value*	*type*	*default value*
name	strm_0	string	strm
src stream	src stream [0]	enumerated	src stream [0]
dest stream	dest stream [0]	enumerated	dest stream [0]
intrpt method	scheduled	integer	scheduled
delay	0.0	double	0.0
color	RGB030	color	RGB030

pt-to-pt transmitter pt_0

attribute	*value*	*type*	*default value*
name	pt_0	string	pt
channel	(See below.)	compound	
icon name	pt_tx	icon	pt_tx

...
...

channel pt 0.channel

attribute	*value*	*type*	*default value*
count	1	integer	0
list	(See below.)	object list	

list pt 0.channel [0]

attribute	*value*	*type*	*default value*
data rate	155,250,000	double	1,024
packet formats	ams_atm_cell	string	all formatted,...

Appendix G

atm(ap)_acb_fifo Node Model Report

Process Model Comments

This is the process model for the queueing packets

Process Model Attributes

Attribute **service_rate** properties

Property	*Value*
Default Value:	9,600
Data Type:	double
Attribute Description:	Private
Auto. assign value:	FALSE
Units:	bits/sec

Process Model Interface Attributes

Attribute **begsim intrpt** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	enabled	N/A
Default Value:	disabled	YES
Data Type:	toggle	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute specifies whether a 'begin simulation interrupt' is generated for a processor module's root process at the start of the simulation.	
Symbol Map:	NONE	YES

Attribute **endsim intrpt** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	toggle	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute specifies whether an 'end simulation interrupt' is generated for a processor module's root process at the end of the simulation.	
Symbol Map:	NONE	YES

Attribute **failure intrpts** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	enumerated	N/A
Attribute Description:	Private	N/A
Comments:		YES

...
...

	This attribute specifies whether failure interrupts are generated for a processor module's root process upon failure of nodes or links in the network model.	
Symbol Map:	NONE	YES

Attribute **intrpt interval** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	toggle double	N/A
Attribute Description:	Private	N/A
Units:	sec.	YES
Comments:		YES
	This attribute specifies how often regular interrupts are scheduled for the root process of a processor module.	
Symbol Map:	NONE	YES

Attribute **priority** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	0	N/A
Default Value:	0	YES
Data Type:	integer	N/A
Attribute Description:	Private	N/A
Low Range:	-32767 inclusive	YES
High Range:	32767 inclusive	YES
Comments:		YES
	This attribute is used to determine the execution order of events that are scheduled to occur at the same simulation time.	
Symbol Map:	NONE	YES

Attribute **recovery intrpts** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	enumerated	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute specifies whether recovery interrupts are scheduled for the processor module's root process upon recovery of nodes or links in the network model.	
Symbol Map:	NONE	YES

...
...

Attribute **super priority** properties

Property	*Value*	*Inherit*
Assign Status:	set	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	toggle	N/A
Attribute Description:	Private	N/A
Comments:	This attribute is used to determine the execution order of events that are scheduled to occur at the same simulation time.	YES
Symbol Map:	NONE	YES

Header Block

```
#define QUEUE_EMPTY    (op_q_empty ())
#define SVC_COMPLETION op_intrpt_type () == OPC_INTRPT_SELF
#define ARRIVAL        op_intrpt_type () == OPC_INTRPT_STRM
```

State Variable Block

```
int    \server_busy;
double \service_rate;
Objid  \own_id;
```

Temporary Variable Block

```
Packet*      acellp_pk;
int          pk_len;
double       pk_svc_time;
int          insert_ok;
int          temp;
int          VCI, VPI, PI;
```

forced state **init**

attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **init**

```
/* initially the server is idle */
server_busy = 0;

/* get queue module's own object id */
own_id = op_id_self ();

/* get assigned value of server processing rate */
```

...
...

```
op_ima_obj_attr_get (own_id, "service_rate", &service_rate);
```

transition init -> arrival

attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB300	color	RGB333
drawing style	line	toggle	spline

transition init -> idle

attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition	default	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state arrival

attribute	*value*	*type*	*default value*
name	arrival	string	st
enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs arrival

```
/* acquire the arriving packet */
/* multiple arriving streams are supported. */
acellp_pk = op_pk_get (op_intrpt_strm ());

/* attempt to enqueue the packet at tail of subqueue 0 */
if (op_subq_pk_insert (0, acellp_pk, OPC_QPOS_TAIL) != OPC_QINS_OK)
   {
   /* the inserton failed (due to to a full queue) */
   /* deallocate the packet */
   op_pk_destroy (acellp_pk);

      /* set flag indicating insertion fail */
      /* this flag is used to determine transition */
      /* out of this state */
   insert_ok = 0;
   }
else{
      /* insertion was successful */
   insert_ok = 1;
```

...
...

transition **arrival -> svc_start**

attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition	!server_busy && ins...	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

transition **arrival -> idle**

attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition	default	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

unforced state **idle**

attribute	*value*	*type*	*default value*
name	idle	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

transition **idle -> arrival**

attribute	*value*	*type*	*default value*
name	tr_5	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

transition **idle -> svc_compl**

attribute	*value*	*type*	*default value*
name	tr_6	string	tr
condition	SVC_COMPLETION	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

...
...

forced state **svc_start**

attribute	*value*	*type*	*default value*
name	svc_start	string	st
enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **svc_start**

```
/* get a handle on packet at head of subqueue 0 */
/* (this does not remove the packet) */
acellp_pk = op_subq_pk_access (0, OPC_QPOS_HEAD);

/* determine the packets length (in bits) */
pk_len = op_pk_total_size_get (acellp_pk);
/* determine the time required to complete */
/* service of the packet */
pk_svc_time = pk_len / service_rate;

/* schedule an interrupt for this process */
/* at the time where service ends. */
op_intrpt_schedule_self (op_sim_time () + pk_svc_time, 0);

/* the server is now busy. */
server_busy = 1;
```

transition **svc_start -> idle**

attribute	*value*	*type*	*default value*
name	tr_7	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	line	toggle	spline

forced state **svc_compl**

attribute	*value*	*type*	*default value*
name	svc_compl	string	st
enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **svc_compl**

```
/* extract packet at head of queue */
/* this is the packet just finishing service */
acellp_pk = op_subq_pk_remove (0, OPC_QPOS_HEAD);
```

...
...

```
op_pk_nfd_get(acellp_pk, "VPI", &temp);
op_pk_nfd_get(acellp_pk, "VCI", &temp);
/*printf("Received cell and sending to %d.\n", temp);
*/
/* forward the packet on stream 0, */
/* causing an immediate interrupt at dest. */

op_pk_send_forced (acellp_pk, 0);

/* server is idle again. */
server_busy = 0;
```

transition svc_compl -> svc_start			
attribute	*value*	*type*	*default value*
name	tr_8	string	tr
condition	!QUEUE_EMPTY	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

transition svc_compl -> idle			
attribute	*value*	*type*	*default value*
name	tr_9	string	tr
condition	default	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline

Appendix H

atm(ap)_acbf_ ms Process Model Report

Process Model Comments

General Process Description:

The acb_fifo_ms queueing process model accepts packets from any number of sources and autonomously forwards them to a single destination module after holding each one for a simulated duration referred to as the packet's service time. The service time may vary from packet to packet and is computed by dividing the packet lengths (measured in bits) by the value of the process model attribute "service rate" (given in bits/sec).

Enqueued packets must wait for the availability of a server resource before entering service, and eventually being forwarded. The number of concurrent servers supported by the queue is configurable via the process model attribute "num_servers". The queue is said to operate with a first-in-first-out (FIFO) discipline since the packets do not begin service until earlier ones have begun service.

Since the packet service times are variable, packets could complete service in a different order than they enetered service (which may not be FIFO); a process model attribute "strict_fifo" allows the queue to be configured in such a way that ackets whose service intervals elapse, cannot fully complete service and be forwarded until earlier arriving packets are themselves forwarded.

ICI Interfaces:

None

Packet Formats:

None

Statistic Wires:

None

Process Registry:

Not Applicable

Restrictions:

---

1. The acb_fifo_ms process model must be contained within a queue module.
2. The source stream index of the output packet stream of the surrounding queue module should be 0.

Process Model Attributes

Attribute **service_rate** properties

Property	*Value*

...
...

Default Value:	9,600	
Data Type:	double	
Attribute Description:	Private	
Auto. assign value:	TRUE	
Units:	bits/sec	
Low Range:	0.0 exclusive	
Comments:	Amount of data that completes service in one second of simulation time.	
Symbol Map:	*Symbol*	*Value*
	9,600	9,600
Allow other values:	YES	

Attribute **num_servers** properties

Property	*Value*	
Default Value:	1	
Data Type:	integer	
Attribute Description:	Private	
Auto. assign value:	TRUE	
Low Range:	1 inclusive	
Comments:	Number of servers supported by the queue.	
Symbol Map:	*Symbol*	*Value*
	1	1
	2	2
	5	5
Allow other values:	YES	

Attribute **strict_fifo** properties

Property	*Value*	
Default Value:	Disabled	
Data Type:	integer	
Attribute Description:	Private	
Auto. assign value:	TRUE	
Comments:	Maintains the FIFO policy of the queue with regard to the release of packets which terminate service out of order. If enabled, the queue holds packets which complete service until all earlier arrivals have also completed service.	
Symbol Map:	*Symbol*	*Value*
	Enabled	1
	Disabled	0
Allow other values:	NO	

...
...

Process Model Interface Attributes

Attribute **begsim intrpt** properties

Property	*Value*	*Inherit*
Assign Status:	hidden	
Initial Value:	enabled	N/A
Default Value:	disabled	YES
Data Type:	toggle	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute specifies whether a 'begin simulation interrupt' is generated for a processor module's root process at the start of the simulation.	
Symbol Map:	NONE	YES

Attribute **endsim intrpt** properties

Property	*Value*	*Inherit*
Assign Status:	hidden	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	toggle	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute specifies whether an 'end simulation interrupt' is generated for a processor module's root process at the end of the simulation.	
Symbol Map:	NONE	YES

Attribute **failure intrpts** properties

Property	*Value*	*Inherit*
Assign Status:	hidden	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	enumerated	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute specifies whether failure interrupts are generated for a processor module's root process upon failure of nodes or links in the network model.	
Symbol Map:	NONE	YES

Attribute **intrpt interval** properties

Property	*Value*	*Inherit*
Assign Status:	hidden	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	toggle double	N/A
Attribute Description:	Private	N/A
Units:	sec.	YES
Comments:		YES
	This attribute specifies how often regular interrupts	

...
...

	are scheduled for the root process of a processor module.	
Symbol Map:	NONE	YES

Attribute **priority** properties

Property	*Value*	*Inherit*
Assign Status:	hidden	
Initial Value:	0	N/A
Default Value:	0	YES
Data Type:	integer	N/A
Attribute Description:	Private	N/A
Low Range:	-32767 inclusive	YES
High Range:	32767 inclusive	YES
Comments:		YES
	This attribute is used to determine the execution order of events that are scheduled to occur at the same simulation time.	
Symbol Map:	NONE	YES

Attribute **recovery intrpts** properties

Property	*Value*	*Inherit*
Assign Status:	hidden	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	enumerated	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute specifies whether recovery interrupts are scheduled for the processor module's root process upon recovery of nodes or links in the network model.	
Symbol Map:	NONE	YES

Attribute **super priority** properties

Property	*Value*	*Inherit*
Assign Status:	hidden	
Initial Value:	disabled	N/A
Default Value:	disabled	YES
Data Type:	toggle	N/A
Attribute Description:	Private	N/A
Comments:		YES
	This attribute is used to determine the execution order of events that are scheduled to occur at the same simulation time.	
Symbol Map:	NONE	YES

Header Block

```
#define QUEUE_EMPTY    (op_q_empty ())
#define SVC_COMPLETION  op_intrpt_type () == OPC_INTRPT_SELF
#define ARRIVAL        op_intrpt_type () == OPC_INTRPT_STRM
```

...
...

```
#define STATUS_FREE      0
#define STATUS_BUSY      1

#define SERVERS_AVAILABLE (num_free_servers > 0)

#define PK_STATUS_WAITING 0.0
#define PK_STATUS_INSVC     1.0
#define PK_STATUS_DONESVC      2.0
```

State Variable Block

```
int*   \server_status_table;
int*        \server_pkid_table;
int         \num_servers;
double  \service_rate;
Objid   \own_id;
int         \num_free_servers;
int         \num_waiting_pks;
int         \strict_fifo;
```

Temporary Variable Block

```
Packet*       acellp_pk;
int          pk_len;
double         pk_svc_time;
int          insert_ok;
int                  i, j, k, a;
int                  num_pks;
int                  server_index;
```

Diagnostic Block

```
printf ("                    configuration:\n");
printf ("                       discipline: %s\n", (strict_fifo) ? "strict fifo" : "fcfs");
printf ("                       # servers:%d\n", num_servers);
printf ("                       svc rate: %g\n", service_rate);
printf ("\n");

printf ("                    Server Table\n");
for (a = 0; a < num_servers; a++)
     {
     if (server_status_table [a] == STATUS_FREE)
          printf ("                       %d) FREE\n", a);
     else{
          printf ("                       %d) BUSY, servicing pk (%d)\n",
               a, server_pkid_table [a]);
          }
     }
printf ("\n");
printf ("                    num free servers:    %d\n", num_free_servers);
printf ("                    num waiting pks:     %d\n", num_waiting_pks);
```

...
...

forced state init

attribute	*value*	*type*	*default value*
name	init	string	st
enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced
initial	enabled	toggle	disabled

enter execs init

```
/* get queue module's own object id */
own_id = op_id_self ();

/* determine number of servers (test for legal value) */
op_ima_obj_attr_get (own_id, "num_servers", &num_servers);
if (num_servers < 1)
    num_servers = 1;

/* get assigned value of server processing rate */
op_ima_obj_attr_get (own_id, "service_rate", &service_rate);

/* see if queue is configured as a strict multi-server */
/* fifo or simply a fcfs fifo. */
op_ima_obj_attr_get (own_id, "strict_fifo", &strict_fifo);

/* allocate a table of status flags - one entry per server */
server_status_table = (int*) op_prg_mem_alloc (sizeof (int) * num_servers);

/* also allocate and init a table mapping servers */
/* to the id's of the packets they are serving */
server_pkid_table = (int*) op_prg_mem_alloc (sizeof (int) * num_servers);
for (i = 0; i < num_servers; i++)
    server_pkid_table [i] = -1;

/* initially all servers are free */
for (i = 0; i < num_servers; i++)
    server_status_table [i] = STATUS_FREE;

/* init the number of free servers */
num_free_servers = num_servers;

/*init the number of waiting pks */
num_waiting_pks = 0;
```

transition init -> arrival

attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB300	color	RGB333
drawing style	line	toggle	spline
path	(See below.)	object list	

...
...

path init -> arrival [0]

attribute	*value*	*type*	*default value*
x	179	double	0.0
y	151	double	0.0

path init -> arrival [1]

attribute	*value*	*type*	*default value*
x	240	double	0.0
y	151	double	0.0

transition init -> idle

attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition	default	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline
path	(See below.)	object list	

path init -> idle [0]

attribute	*value*	*type*	*default value*
x	143	double	0.0
y	179	double	0.0

path init -> idle [1]

attribute	*value*	*type*	*default value*
x	166	double	0.0
y	264	double	0.0

path init -> idle [2]

attribute	*value*	*type*	*default value*
x	235	double	0.0
y	278	double	0.0

path init -> idle [3]

attribute	*value*	*type*	*default value*
x	301	double	0.0
y	278	double	0.0

...
...

forced state **arrival**

attribute	*value*	*type*	*default value*
name	arrival	string	st
enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced
initial	disabled	toggle	disabled

enter execs **arrival**

```
/* acquire the arriving packet */
/* multiple arriving streams are supported. */
acellp_pk = op_pk_get (op_intrpt_strm ());

/* new packets are stamped to */
/* indicate the packet is waiting for service. */
op_pk_priority_set (acellp_pk, PK_STATUS_WAITING);

/* attempt to enqueue the packet at tail of subqueue 0 */
if (op_subq_pk_insert (0, acellp_pk, OPC_QPOS_TAIL) != OPC_QINS_OK)
   {
   /* the inserton failed (due to to a full queue) */
   /* deallocate the packet */
   op_pk_destroy (acellp_pk);

      /* set flag indicating insertion failed */
      /* this flag is used to determine transition */
      /* out of this state */
   insert_ok = 0;
   }
else{
      /* insertion was successful */
   insert_ok = 1;

      /* increase the number of waiting packets */
      /* which are not currently in service */
      num_waiting_pks++;
   }
```

transition **arrival -> svc_start**

attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition	SERVERS_AVAILABLE &...	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline
path	(See below.)	object list	

...
...

path arrival -> svc_start [0]

attribute	*value*	*type*	*default value*
x	291	double	0.0
y	129	double	0.0

path arrival -> svc_start [1]

attribute	*value*	*type*	*default value*
x	328	double	0.0
y	113	double	0.0

path arrival -> svc_start [2]

attribute	*value*	*type*	*default value*
x	370	double	0.0
y	127	double	0.0

transition arrival -> idle

attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition	default	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline
path	(See below.)	object list	

path arrival -> idle [0]

attribute	*value*	*type*	*default value*
x	266	double	0.0
y	179	double	0.0

path arrival -> idle [1]

attribute	*value*	*type*	*default value*
x	276	double	0.0
y	219	double	0.0

path arrival -> idle [2]

attribute	*value*	*type*	*default value*
x	309	double	0.0
y	248	double	0.0

...
...

unforced state **idle**

attribute	*value*	*type*	*default value*
name	idle	string	st
enter execs	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced
initial	disabled	toggle	disabled

transition **idle -> arrival**

attribute	*value*	*type*	*default value*
name	tr_5	string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline
path	(See below.)	object list	

path **idle -> arrival [0]**

attribute	*value*	*type*	*default value*
x	300	double	0.0
y	267	double	0.0

path **idle -> arrival [1]**

attribute	*value*	*type*	*default value*
x	255	double	0.0
y	227	double	0.0

path **idle -> arrival [2]**

attribute	*value*	*type*	*default value*
x	256	double	0.0
y	176	double	0.0

transition **idle -> svc_compl**

attribute	*value*	*type*	*default value*
name	tr_6	string	tr
condition	SVC_COMPLETION	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline
path	(See below.)	object list	

...
...

path **idle -> svc_compl [0]**

attribute	*value*	*type*	*default value*
x	356	double	0.0
y	284	double	0.0

path **idle -> svc_compl [1]**

attribute	*value*	*type*	*default value*
x	480	double	0.0
y	234	double	0.0

path **idle -> svc_compl [2]**

attribute	*value*	*type*	*default value*
x	519	double	0.0
y	178	double	0.0

forced state **svc_start**

attribute	*value*	*type*	*default value*
name	svc_start	string	st
enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced
initial	disabled	toggle	disabled

enter execs **svc_start**

```
    /* find the first packet which is waiting for service */
    /* these packets are stamped with appropriate prioririty */
    num_pks = op_subq_stat (0, OPC_QSTAT_PKSIZE);
    for (i = 0; i < num_pks; i++)
 5      {
        acellp_pk = op_subq_pk_access (0, i);
        if (op_pk_priority_get (acellp_pk) == PK_STATUS_WAITING)
            break;
        }
10
    /* determine the packet's length (in bits) */
    pk_len = op_pk_total_size_get (acellp_pk);

    /* determine the time required to complete */
15  /* service of the packet */
    pk_svc_time = pk_len / service_rate;

    /* once a packet begins service its priority level is */
    /* changed to reflect this */
20  op_pk_priority_set (acellp_pk, PK_STATUS_INSVC);

    /* find a free server to process the packet */
    for (i = 0; i < num_servers; i++)
        {
```

...
...

```
25      if (server_status_table [i] == STATUS_FREE)
            {
            /* schedule an interrupt for this server */
            /* at the time where service ends. */
            op_intrpt_schedule_self (op_sim_time () + pk_svc_time, i);
30
            /* the server is now busy. */
            server_status_table [i] = STATUS_BUSY;

            /* reduce the number of free servers */
35          num_free_servers--;

            /* decrease the number of waiting packets */
            num_waiting_pks--;

40          /* establish a corespondence between the server */
            /* and the packet which it is serving */
            server_pkid_table [i] = op_pk_id (acellp_pk);

            /* stop search for free server */
45          break;
            }
        }
```

transition **svc_start -> idle**

attribute	*value*	*type*	*default value*
name	tr_7	string	tr
condition		string	
executive		string	
color	RGB300	color	RGB333
drawing style	line	toggle	spline
path	(See below.)	object list	

path **svc_start -> idle [0]**

attribute	*value*	*type*	*default value*
x	377	double	0.0
y	177	double	0.0

path **svc_start -> idle [1]**

attribute	*value*	*type*	*default value*
x	335	double	0.0
y	240	double	0.0

forced state **svc_compl**

attribute	*value*	*type*	*default value*
name	svc_compl	string	st

...
...

enter execs	(See below.)	textlist	
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced
initial	disabled	toggle	disabled

enter execs **svc compl**

```
/* Determine which server is completing service. */
/* This is passed as the code associated with */
/* the service completion interrupt. */
server_index = op_intrpt_code ();

/* find the packet which was being served */
num_pks = op_subq_stat (0, OPC_QSTAT_PKSIZE);
for (i = 0; i < num_pks; i++)
    {
    acellp_pk = op_subq_pk_access (0, i);
    if (op_pk_id (acellp_pk) == server_pkid_table [server_index])
        {
        /* the packet completing service is number 'i' */

        /* server is free again. */
        server_status_table [server_index] = STATUS_FREE;
        num_free_servers++;

        /* if this is not the first packet in the subqueue */
        /* then pulling it out would, in a strict sense, not */
        /* model a fifo. Two methodologies are supported here: */
        /* the "strict" methodolgy accurately maintains fifo */
        /* discipline; the second maintains only a first-come- */
        /* first-serve discipline with service completion times */
        /* not necessarily tracking the order of service start times. */
        if (!strict_fifo)
            {
            /* remove it from the queue */
            acellp_pk = op_subq_pk_remove (0, i);

            /* forward the packet on stream 0, */
            /* causing an immediate interrupt at dest. */
            op_pk_send_forced (acellp_pk, 0);
            }
        else{
            if (i == 0)
                {
                /* all contiguous packets which have completed service */
                /* and are at the head of the queue may be removed and */
                /* forwarded. Count how many such packets there are. */
                for (j = 1; j < num_pks; j++)
                    {
                    acellp_pk = op_subq_pk_access (0, j);
                    if (op_pk_priority_get (acellp_pk) != PK_STATUS_DONESVC)
                        break;
                    }

                /* the number of packets to be pulled is 'j' */
                /* send all but the last one quietly */
                for (k = 0; k < j - 1; k++)
```

```
...
...
                    {
                    /* pull the head off from the queue and send it quietly */
                    acellp_pk = op_subq_pk_remove (0, OPC_QPOS_HEAD);
                    op_pk_send_quiet (acellp_pk, 0);
                    }

                /* send the last one forced, to cause an interrupt */
                acellp_pk = op_subq_pk_remove (0, OPC_QPOS_HEAD);
                op_pk_send_forced (acellp_pk, 0);
                }
            else{
                /* the service of this packet can be considered complete */
                /* but the packet can not be forwarded until ones ahead */
                /* of it are completely serviced as well. The packet is */
                /* tagged as 'blocked' by setting its priority to -1.0 */
                op_pk_priority_set (acellp_pk, PK_STATUS_DONESVC);
                }
            }

        /* stop the search */
        break;
        }
    }
```

transition svc_compl -> svc_start			
attribute	*value*	*type*	*default value*
name	tr_8	string	tr
condition	num_waiting_pks > 0	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline
path	(See below.)	object list	

path svc_compl -> svc_start [0]			
attribute	*value*	*type*	*default value*
x	489	double	0.0
y	127	double	0.0

path svc_compl -> svc_start [1]			
attribute	*value*	*type*	*default value*
x	441	double	0.0
y	114	double	0.0

path svc_compl -> svc_start [2]			
attribute	*value*	*type*	*default value*
x	405	double	0.0
y	124	double	0.0

...
...

transition **svc_compl -> idle**

attribute	*value*	*type*	*default value*
name	tr_9	string	tr
condition	default	string	
executive		string	
color	RGB300	color	RGB333
drawing style	spline	toggle	spline
path	(See below.)	object list	

path **svc_compl -> idle [0]**

attribute	*value*	*type*	*default value*
x	486	double	0.0
y	169	double	0.0

path **svc_compl -> idle [1]**

attribute	*value*	*type*	*default value*
x	453	double	0.0
y	213	double	0.0

path **svc_compl -> idle [2]**

attribute	*value*	*type*	*default value*
x	355	double	0.0
y	253	double	0.0