NOTE TO USERS

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

David Tai-Wei Lu
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

A scheme for dynamic deployment of bandwidth in opto-electronic switches and agile optical networks

TITRE DE LA THÈSE / TITLE OF THESIS

T. Hall
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Hanan Anis

Changcheng Huang

Gary W. Slater
LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

A scheme for dynamic deployment of bandwidth in opto-electronic switches and agile optical networks

By

David Tai-Wei Lu

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the MASc degree in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information and Technology
University of Ottawa

Your file Votre référence
ISBN: 0-494-11333-2
Our file Notre référence
ISBN: 0-494-11333-2

Acknowledgements

I would like to say the deepest thank you to my thesis supervisor

Professor Trevor Hall

for his knowledge, guidance and help at all times.

I would also like to say thank you to

Dr. Sofia Paredes

for her knowledge, friendship and consultation at all times.

This work was supported by the Natural Sciences and Engineering Research Council (**NSERC**) and industrial and government partners, through the Agile All-Photonic Networks (**AAPN**) Research Network.

Abstract

This thesis investigates a novel scheme of Flexible Bandwidth Provision (FBP) in a three-stage packet switch. The proposed switch architecture is a hybrid optics/electronics Clos-like switch. The electronics domain is used as a memory technology, and the photonics domain is used as a communication technology. The architecture and method are suitable for the relatively slow reconfiguration of current transparent photonic switch technology. The advantage of including both optics and electronics implementation is that packets do not contend over resources, such as shared buses, that are not fundamental to a packet switch. The FBP algorithm is used to change the configuration of the optical centre stage to provide the bandwidth required by the incoming traffic. The switching performance generally constrains by either the data rate or the total port number of the switch.

This thesis will then examine the mapping of FBP onto an Agile All-Photonic Network (AAPN). AAPN proposes an overlaid-star topology with edge-nodes interconnected by the beams of the stars via the core switches located at the hub of each star. The Adaptive Wavelength-Timeslot Allocation (AWTA) algorithm is introduced for the network to adapt the configuration of a core node to meet the demands of changing traffic patterns. If the algorithm is used over more than one core node, it may also provide an option to the edge nodes to choose distinct routes with different propagation delays, which can relieve the routing of delay-critical traffic. The results show that the queuing delay performance is directly proportional to the time between configurations of the core node and the links propagation delays.

Table of Contents

Acknowledgements ii
Abstract iii
Table of Contents iv
List of Figures v
List of Symbols viii
List of Acronyms ix
1 Introduction 1
2 Background 6
2.1 Switch Architectures 6
2.2 Optical Network 10
2.3 Internet Traffic 12
3 Flexible Bandwidth Provision Switch Architecture 14
3.1 Switch Architecture 14
3.2 Flexible Bandwidth Provision 17
3.2.1 Estimated Traffic Matrix 18
3.2.2 Service Rate Matrix 19
3.2.3 Matching Algorithm 22
3.3 OPNET Implementation 22
3.4 Performance Analysis 29
3.4.1 Bernoulli Traffic 30
3.4.2 Pareto Traffic 35
3.5 FBP Switch with IP Traffic 37
4 Agile All-Photonic Network 43
4.1 Agile All-Photonic Network Topology 43
4.1.1 Edge-nodes 45
4.1.2 Core nodes 48
4.2 Mapping the FBP switch architecture onto AAPN 49
4.3 Adaptive Wavelength-Timeslot Allocation (AWTA) algorithm 51
4.4 OPNET Implementation 54
4.5 Performance Analysis 62
4.5.1 AAPN-AWTA with Uniform Poisson Traffic 67
4.5.2 AAPN-AWTA with Non-uniform Poisson Traffic 71
4.5.3 AAPN-AWTA with Uniform Bursty Traffic 74
4.5.4 AAPN-AWTA with Uniform IP Traffic 76
5 Summary and Conclusions 78
5.1 Suggestions for Future Work 82
6 References 84

List of Figures

Figure 1 A 4x4 crossbar switch 7
Figure 2 Output queued switch 7
Figure 3 Virtual output queuing. Q_{pq} is the queue for packets going from input p to output q 8
Figure 4 Output Queued Switch implemented with a Common Memory Switch 9
Figure 5 Clos Network 10
Figure 6 The Evolution of the Photonic Network [20] 12
Figure 7 Three-stage packet switch with optical core for n=6, l=4, N=nl=24, m=8, k=1.333 15
Figure 8 Input sector 0, a_0, for the switch in Figure 7. N=24, n=6, l=4, m=8. $Q_{i,j}$ is the VOSQ for traffic going from input sector i to output sector j 16
Figure 9 Output sector 2 for the switch in Figure 7. N=24, n=6, l=4, m=8. Qq is the dedicated queue for traffic going to output q, $0 \le q \le N-1$ 17
Figure 10 Network Model used to implement and test the FBP switch 23
Figure 11 FBP switch model at the node level. The solid lines represent packet streams and the dotted lines represent the association of a transmitter with a receiver 25
Figure 12 Input sector process model 26
Figure 13 FBP "hub" process model 27
Figure 14 Workstation node Model 28
Figure 15 Process model in the "proc" object of the workstation node 29
Figure 16 Average delay versus load obtained with a speed up of two for Bernoulli traffic arrival with uniform destination with τ = 50, 100, 200 32
Figure 17 Average delay and input queuing delay versus load obtained with a speed up of two for Bernoulli traffic arrivals with uniform destinations for τ = 50 32
Figure 18 Average delay versus load obtained with a speed up of one for Bernoulli traffic arrivals with uniform destinations for τ = 50, 100, 200 33
Figure 19 Average delay versus load obtained with a speed up of two for Bernoulli traffic arrival with non-uniform destination with τ = 50, 100, 200 35
Figure 20 Average delay versus load obtained with a speed up of two for Pareto traffic arrival with non-uniform destination with τ = 50, 100, 200 36
Figure 21 Average delay versus load obtained with a speed up of one for Pareto traffic arrival with uniform destination with τ = 50, 100, 200 37
Figure 22 Sample IP traffic trace in OPNET 38
Figure 23 Typical IP Switch node model 39
Figure 24 IP_dispatch process model 40
Figure 25 logical IP network 41
Figure 26 IP router with FBP functionality 42
Figure 27 A one- star network with eight edge-nodes and one core-node 44
Figure 28 An overlaid two-star network with eight- edge nodes and two-core nodes 45
Figure 29 A one star network with four edge nodes and a single core node with the detail of the source and destination functional blocks shown separately. 47

Figure 30 Detail of the core node showing the stacking of the crossbars within the node 49
Figure 31 AAPN Topology revisited. (a) Edge nodes consist of a source and a destination module, (b) Equivalent three-stage architecture The architecture becomes a three-stage network if the edge nodes are logically split in source (first stage) and destination (third stage) 50
Figure 32 Logical equivalence between the AAPN topology and the FBP switch. (a) Three stage network, (b) The core node consists of a stacking of crossbars, one for each wavelength. 51
Figure 33 Timeline of the Adaptive Timeslot Allocation method for all edge nodes located at the same distance from the core node 53
Figure 34 Timeline of the Adaptive Timeslot Allocation method for edge nodes at different distances from the core node 54
Figure 35 AAPN topology used for the simulations in OPNET 55
Figure 36 Elements of the AAPN Core Node 56
Figure 37 Process model in AAPN core node 57
Figure 38 Modules of the AAPN Edge Node 58
Figure 39 Process Model of the *general processor* in the Edge Node 59
Figure 40 Process Model of the *shared memory* module in the Edge Node 60
Figure 41 AAPN Network implemented to include the features of IP traffic 61
Figure 42 Modules of the edge node variation used for the simulation with IP traffic ... 61
Figure 43 Modules of the IP Traffic Generator node 62
Figure 44 Average delay in edge node 1 versus load obtained with the no propagation delays scenario. The speed up is two and the traffic is Poisson with uniform destinations. Values for τ are 80, 160 and 240 68
Figure 45 Average delay in edge node 1 versus load with $\tau = 80$ obtained for three different scenarios: network with no delay, network with fixed delays and network with various delays. The speed up is two and traffic is Poisson with uniform destinations 69
Figure 46 Average delay in various edge nodes versus load with $\tau = 80$ obtained for network with various delays. The speed up is two and traffic is Poisson with uniform destinations 70
Figure 47 Average delay in edge node 1 versus load with $\tau = 2100$ obtained for two different scenarios: network with no delay and network with long delay. The speed up is two and traffic is Poisson with uniform destination 71
Figure 48 Average delay in edge node 1 versus load obtained with the no propagation delays scenario. The speed up is two and the traffic is Poisson with non-uniform destinations. Values for τ are 80, 160 and 240 72
Figure 49 Average delay in edge node 1 versus load with $\tau = 80$ obtained for three different scenarios: network with no delay, network with fixed delays and network with various delays. The speed up is two and traffic is Poisson with non-uniform destinations 73
Figure 50 Average delay in edge node 1 versus load with $\tau = 2100$ obtained for two different scenarios: network with no delay and network with long delay. The speed up is two and traffic is Poisson with non-uniform destinations 74

Figure 51 Average delay in edge node 1 versus load obtained with the no propagation delays scenario. The speed up is two and the traffic is Bursty with uniform destinations. Values for τ are 80, 160 and 240 75
Figure 52 Average delay in edge node 1 versus load with $\tau = 80$ obtained for three different scenarios: network with no delay, network with fixed delays and network with various delays. The speed up is two and traffic is Bursty with uniform destinations 75
Figure 53 Average delay in edge node 1 versus load with $\tau = 2100$ obtained for two different scenarios: network with no delay and network with long delay. The speed up is two and traffic is Bursty with non-uniform destination 76
Figure 54 Average delay in edge node 1 versus load obtained with the no propagation delays scenario. The speed up is two and the IP traffic with uniform destinations. Values for τ are 80, 160 and 240 77

List of Symbols

N	number of incoming links to switch
k	speed up
kN	number of available interconnections in the central stage
m	number of crossbar switches in the central stage
n	number of inputs/outputs per sector
l	number of sectors
Λ	inter-sector traffic matrix
Λ_e	estimated inter-sector traffic matrix
Λ_c	current inter-sector traffic matrix
$\overline{\Lambda}_e$	normalize estimated inter-sector traffic matrix
S_{min}	minimal inter-sector service rate requirement
S	inter-sector service rate matrix
X	excess matrix
B	bistochastic matrix
σ	adaptation speed variable
τ	inter-configuration period
α	tail index of a Pareto distributed random variable
β	minimum value of a Pareto distributed random variable
s_time	the waiting time for edge nodes to apply the received service matrix
t_time	the waiting time for edge node to start transmitting traffic matrix

List of Acronyms

AAPN	Agile All Photonic Network
AWTA	Adaptive Wavelength-Timeslot Allocation
FBP	Flexible Bandwidth Provision
FELC/VLSI	Ferroelectric Liquid Crystal over Silicon VLSI
ICI	Internal Control Information
IP	Internet Protocol
OADM	Optical Add Drop Multiplexer
OQ	Output Queues
OXC	Optical Cross-connect
PDF	Probability Density Function
SONET	Synchronous Optical Network
SDH	Synchronous Digital Hierarchy
VOSQ	Virtual Output Sector Queues
VIQ	Virtual Input Queues
VOQ	Virtual Output Queues
WDM	Wavelength Division Multiplexing
WC	Wavelength Converter

1 Introduction

This thesis investigates a novel scheme of Flexible Bandwidth Provision (FBP) in a three-stage packet switch, which was introduced by Trevor. J. Hall and Sofia A. Paredes in Dr Paredes's PhD thesis, *Flexible Bandwidth Provision and Scheduling in a Packet Switch with an Optical Core* [1] as well as in [2].

Increasing network traffic keeps setting new, higher demands on networks. The traffic amount doubles every few months [3]. Implementing an optical network in core domains has the potential to solve the problem of the Internet's quick growth. However, there are still quite a few unsolved problems in optical networks. As traffic continues to increase, four important key factors require continuous improvement in order for the Internet to continue to provide good service. The four factors are link speeds, router data throughput, packet forwarding rates, and quick adaptation to routing changes [3]. The recent development of optical fiber technology improves the link speed dramatically. However, except for the first factor, the link speed; the last three factors are all related to the router performance, which indicates that router operation is the key for successful and rapid transmission of dense traffic network in order to increase the data rate. A router is responsible for forwarding the received data packets to the appropriate output links. A typical forwarding process includes header recognition, longest prefix matching (for IP networks), matching with the correct output links and header update. It is essential to improve the router performance in order to keep up with the speed of optical fiber technology.

In the current network environment, packet switches are implemented mostly electronically. Researchers have proposed many schemes to deploy optics in them. Building on research on Spatial Light Modulator (SLM) based optical switches [4][5][6]; the viability of an optoelectronic Clos-like packet switch architecture [7] was demonstrated by the POETS project [8]. Many architectures in the prior art map neatly onto this structure; including the knock-out switch [9], the generalized knockout switch [10], the combined input-output queued switch [11], the Birkhoff-von Neumann switch

[12], the load balanced Birkhoff-von Neumann Switch [13][14], and the Parallel Packet Switch [15]. The various different architectures differ in the detail of their control strategies and the stages of the Clos circuit switch architecture that is buffered. The POETS demonstrator was constrained to packets in bit-parallel format because of the use of Ferroelectric Liquid Crystal over Silicon VLSI technology (FELC/VLSI). Switching packets in bit serial format that is enabled by VCSEL technology was studied in the VIVALDI project [16] and by others [17].

The proposed switch architecture is a hybrid optics/electronics Clos-like switch that includes both the electronics and photonics domains. The electronics domain is used as a memory technology and the photonics domain is used as a communication technology. The architecture and method are suitable for the relatively slow reconfiguration of current transparent photonic switch technology. The advantage of including both optics and electronics implementation is that packets do not contend over resources such as shared buses that are not fundamental to a packet switch. However, the drawback is the cost of repeated optic-to-electronic and electronic-to-optic conversions.

Contention is a fundamental problem in the packet switches that reside in the core routers of a network. Contention occurs when two packets headed to the same egress port arrive at a router at the same time. The common approach to solve contention is the use of a suitable queuing scheme to buffer packets and a scheduling algorithm to avoid collision. Due to the fact that there is no piratical photonic memory with current technology, the proposed scheme uses electronics for the buffering of packets. The low cost of electronic memory devices also offers an advantage in selecting electronic buffering; however, the resulting number of optoelectronic interfaces is a significant factor in the cost.

The first stage includes l centralized shared memory packet switch modules that switch and buffer packets between n ingress ports and m egress ports. The buffers in the input sectors are organized as Virtual Output Sector Queues (VOSQ); i.e., there is one queue for packets destined to each output sector. The centre stage interconnects the first and

third stage and it consists of m crossbar switches of dimension $l \times l$ with no buffering. The third stage includes l centralized shared memory packet switch modules that switch and buffer packets between m ingress ports and n egress ports. The buffers in the output sector are organized as Output Queues (OQ), i.e. one queue for packets destined to each output port. The configuration of the photonic centre stage is found by solving an edge-colouring problem on a bipartite graph which is defined by the incoming traffic. The calculation and reconfiguration only need to be performed if the statistical pattern of the arriving traffic changes. Since a per-timeslot scheduler is not warranted because reconfiguration is only needed to adapt to the slower variations in arrival statistics, a major bottleneck is removed.

The FBP algorithm is used to change the configuration of the optical centre stage to provide the bandwidth required by the incoming traffic. The process ensures that all queues are serviced at a sufficient rate to keep queue lengths finite under the bursty nature of aggregated traffic [18]. The basic principle of FBP algorithm is to bind together several lower rate links to form higher rate logical links and optimize the data transfer. The FBP algorithm improves the switch performance dramatically.

Along the way in researching with the FBP switch, it became apparent that a distributed version of the FBP switch architecture can map into an optical star network. The first and third stages together become an edge node and the centre stage becomes the core node. The new network is a star network which is the fundamental network architecture of the Agile All-Photonic Network (AAPN) proposal [19]. Since the FBP internal switch interconnections are optical links, it is feasible to attempt such mapping, with the main considerations being the effect of propagation delays, which were negligible in the FBP context.

AAPN proposes an overlaid-star topology with edge-nodes interconnected by the beams of the stars via core switches located at the hub of each star. The beams usually consist of two or more fibers that transport wavelength division multiplexed (WDM) traffic between edge-nodes and the bandwidth of each wavelength is shared in time by

assigning varying numbers of timeslots to each flow of data. The core switches operate on a slotted mode to forward incoming traffic. It must be possible to create and destroy paths sufficiently rapidly that they endure for only microseconds.

The Adaptive Wavelength-Timeslot Allocation (AWTA) algorithm is introduced for the AAPN network to adapt the configuration of a core node to meet the demands of changing traffic patterns. If the method is used over more than one core node, it may also provide an option to the edge nodes to choose distinct routes with different propagation delays, which can relieve the routing of delay-critical traffic. In this work, however, only the case of a single core node was studied.

This thesis is structured in the following manner. Chapter 2 presents some basic packet switch architecture concepts, optical network transmission, and a few definitions used in the thesis. Chapter 3 describes the architecture of the three-stage packet switch and elucidates on the concept of Flexible Bandwidth Provision method. It also discusses the implementation of the switch architecture model in OPNET Modeler and shows the performance results in steady state obtained from the simulations. The last part of this chapter describes the motivation to investigate the FBP switch under standard IP network traffic. Chapter 4 describes the concept of the Agile All-Photonic Networks and the Adaptive Wavelength-Timeslot Allocation algorithm. It illustrates the mapping from the FBP switch onto AAPN and discusses the implementation of this network model in OPNET Modeler. The performance analysis obtained from the simulations is presented at the end of the chapter. Chapter 5 draws conclusions and describes remarks about the FBP and AWTA architectures and methods. A short discussion regarding the future improvement of the simulation models is included.

The following is a list of the main contributions presented in this thesis:

- The implementation of the FBP switch model in OPNET Modeler, which provides a test bed suitable for further testing and research on the subject.

- The validation of the performance analysis obtained with the previous implementation of the FBP architecture in the C++ language.
- The implementation of an AAPN single star network topology in OPNET Modeler, also suitable for further testing and research on the subject.
- The development of the Adaptive Wavelength-Timeslot Allocation (AWTA) algorithm, which corresponds to a distributed version of the FBP algorithm with the main consideration of the effect of propagation delays, which were negligible in the FBP context.
- The implementation of the AWTA algorithm and evaluation of its performance by simulation.

2 Background

This chapter presents some basic network concepts, a brief background of packet switch architectures, optical network transmission and a few definitions related to packet switching used in the thesis.

2.1 Switch Architectures

Packet switches transfer the packets to their local destination ports, pre-determined by the routing protocols. *Contention* is a fundamental problem in the packet switches. It occurs when more than one packet headed to the same egress port arrive at a switch at the same time. The common approach to solve contention is the use of a suitable queuing scheme to buffer packets and a scheduling algorithm to avoid collision. Scheduling is the method used to choose which packet to send during each timeslot if there is more than one packet contending for the same output.

Crossbar Switch

Figure 1 shows the basic crossbar circuit. Input ports can directly set up physical connections to different output ports. Input ports can only connect to one output at one time.

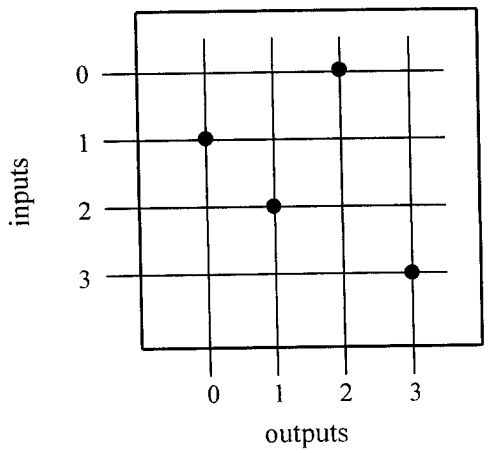

Figure 1 A 4x4 crossbar switch

Output Queued Switch

Figure 2 shows the architecture of an output queued (OQ) switch containing 4 inputs, 4 outputs and 4 output queues. An output queued switch is similar to the crossbar circuit except the output queued switch has buffers for each output ports.

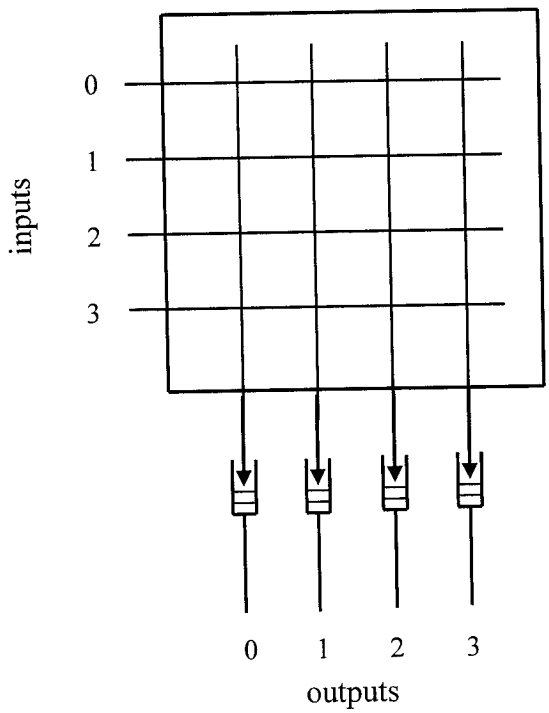

Figure 2 Output queued switch

Virtual Output Queuing

Figure 3 shows the architecture of virtual output queuing (VOQ) switch with N inputs and N outputs. Each input maintains N queues for each output in VOQ switch. Q_{pq} is the queue for packets transmitting from input port p to output port q, $0 \le p \le N-1$ and $0 \le q \le N-1$.

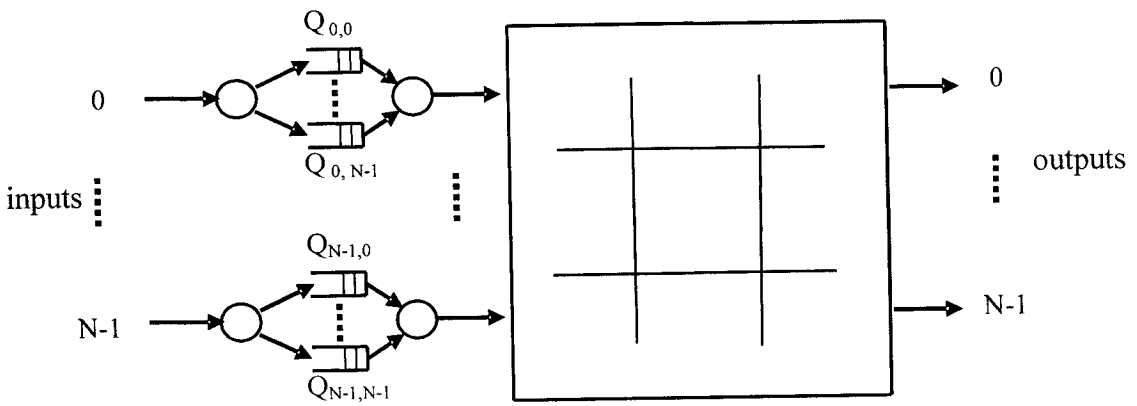

Figure 3 Virtual output queuing. Q_{pq} is the queue for packets going from input p to output q

Shared memory switch

A basic component of packet switch architecture is the common memory switch, also referred as centralized shared memory switch.

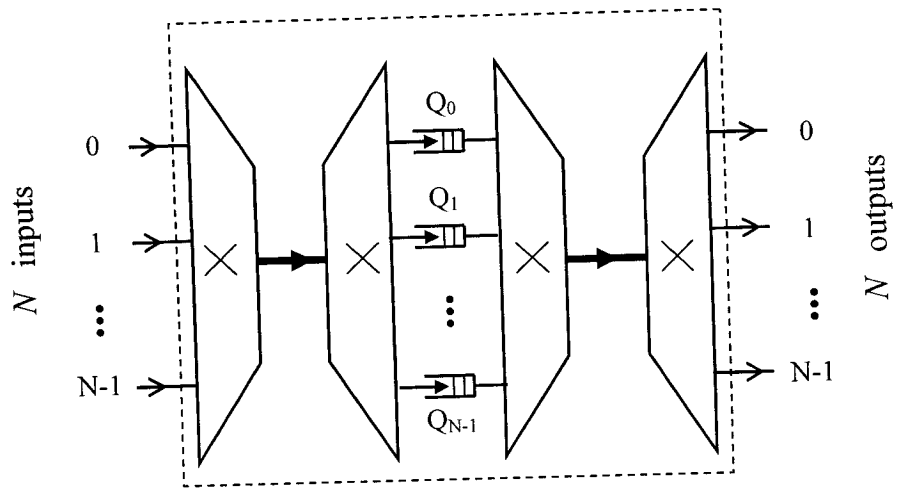

Figure 4 Output Queued Switch implemented with a Common Memory Switch

When a common memory is used as a packet switch, packets must remain in the switch whilst waiting for departure. If there are more incoming packets than the common memory switch can store, those new packets will be discarded. This issue can be solved by increasing the storage to surround larger buffers since the memory in the switch is common. Therefore, the dynamic queues can be effective to reduce packet loss probability and the minimum total size of memory required.

Clos Network

A Clos network as shown in Figure 5 is strictly non-blocking if a connection can be made between any pair of ports without having to disturb an existing connection. A Clos network needs to have $m \geq 2n-1$ in order to meet this strictly non-blocking requirement.

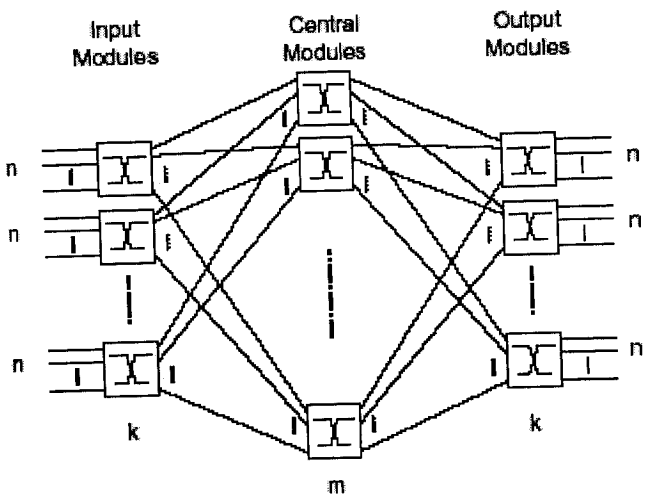

Figure 5 Clos Network

2.2 Optical Network

Although many people believe and expect Optical Networks to be the base of infrastructure for the next generation of data network, there are still many issues that need to be solved. Existing networks have widely adopted the Synchronous Optical Network (SONET), which is the American National Standards Institute standard, Synchronous Digital Hierarchy (SDH), which is the International equivalent of SONET, Wavelength Division Multiplexing (WDM), and DWDM for physical layer transmitting schemes. The Optical Network, however, is still not mature enough to implement Internet traffic straight over the optical network. The main function for current Optical Networks is transmitting signals in bits. It does not perform functions such as buffering, routing, and protecting in the optical domain.

Several components of Optical Network such as Wavelength Converters (WC), Optical Cross-connect (OXC), and Optical Add Drop Multiplexer (OADM) have made the most significant progress so far in the maturation process. Although most of these products are still far from its theoretical requirements, these components have been developed for longer periods of time.

On the other hand, other components of the Optical Network, such as an Optical Router and an Optical Buffer are still far from completion. In fact, only lab demonstrations have been revealed, and no commercial products are available yet. The problems in Network Layer are even worse. Many proposed schemes such as control plane and routing are trying to improve network utilization. However, most schemes have only software simulations for demonstrations.

There are three main schemes for transmitting data through a network. The first method uses Connectionless Packet Switching, for example, IP. The second method uses connection oriented packet switching, such as MPLS and ATM, and the third method is circuit switching, such as Wavelength Multiplexing [20]. In wavelength routed networks, the direct wavelength path can be established by introducing optical cross connect switches at each node. Currently, the main problem with optical networks is not bandwidth or the speed of the optical fiber. The problem is with the routing techniques because it is still impossible to achieve optical header recognition. At the router, it can only read the header in the optical domain and process it in the electrical domain, and then bring it back to the optical domain. This process is called Optical-Electrical-Optical conversion (O/E/O).

A classic approach for IP over Optical Network involves four layers, with each layer having a different scope. The IP layer is responsible for carrying applications and services, the ATM layer is responsible for traffic engineering function, SONET/SDH is responsible for transporting data, and DWDM is responsible for capacity. When a typical package sends from one host to another host, it first needs to transfer data into the IP package, then transfer into an ATM Cell, then transfer into the SONET frame, and lastly transfer into the DWDM fiber, as shown in Figure 6.

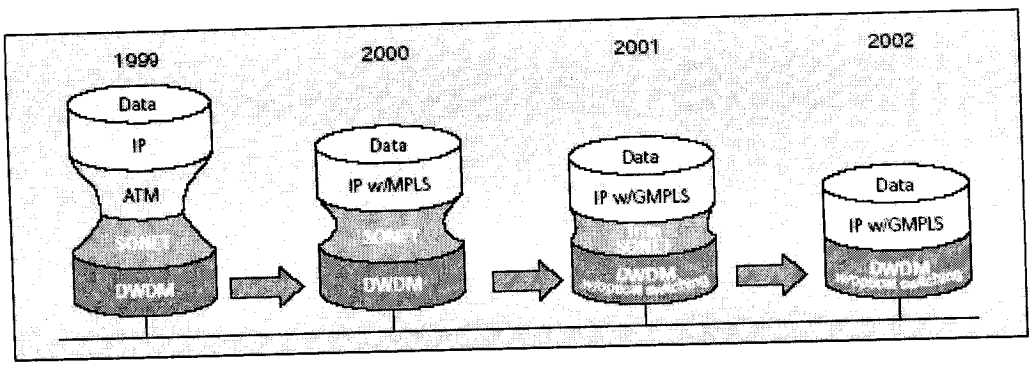

Figure 6 The Evolution of the Photonic Network [20]

Multilayer architectures typically suffer from their many layers. When an additional layer exists in the architecture, more headers need to be added into the package, which causes more overhead and less bandwidth utilization. Each layer needs its own control plane to supervise the layer, and additional layers mean that extra control planes need to be implemented, controlled, and updated. Each layer must have its own topology and communicate with its neighbor layers, requiring more equipment and increasing operation costs. One layer less in the architecture implies less communication between layers. As the evolution of Photonic network continues, architects will try to bypass all unnecessary layers.

2.3 Internet Traffic

A non-stationarity stochastic process is one in which the probability characteristics of the process vary as a function of time.

A traffic matrix is commonly defined as a $N \times N$ matrix [21] that characterizes the traffic load between input and output ports. A port is oversubscribed when the expected rate of packets transmitted for that port exceeds the port capacity. A traffic matrix is defined to be admissible in both the input and output ports are not oversubscribed.

If traffic followed a Poisson or Markovian arrival process, it would have a characteristic burst length. However; it would be smoothed when averaged over a long period of time. Bursty traffic, which has been identified as being self-similar, has no characteristic burst length no matter what scale it is viewed. The traffic is always bursty in all time scales [22].

3 Flexible Bandwidth Provision Switch Architecture

This chapter introduces and describes the packet switch architecture with Flexible Bandwidth Provision. The three-stage packet switch architecture is described, which consists of a reconfigurable optical interconnection as a central stage, surrounded by two electronic buffering stages. The proposed switch architecture is a hybrid optics/electronics Clos-like switch which includes both the electronics and photonics domains. The electronics domain is used as a memory technology and photonics domain is used as a communication technology and the modular architecture accommodates the relatively slow reconfiguration of current transparent photonic switch technology. The Flexible Bandwidth Provision algorithm is used to change the configuration of the optical centre stage to allocate the requested/desired bandwidth according to the incoming traffic. The configuration of the photonic centre stage is found by solving an edge-coloring problem on a bipartite graph which is defined by the traffic. This chapter also discusses the implementation of the switch architecture model in OPNET and shows the performance results in steady state obtained from the simulations.

3.1 Switch Architecture

Since IP packets have variable sizes, it is a normal practice for a switch to segment the variable length packets into convenient fixed size cells. A cell usually refers to a fixed length packet. These new converted cells, in the same way as IP packets, contain a local header and a data payload. The local header reveals the destination port address and the data payload originates from the segmentation of the original IP packet header. The terms *cell* and *packet* will be used interchangeably in the remainder of this work.

It is assumed that the core switch of a router operates in a time slotted basis. The duration of the timeslot is equal to the duration of a cell and those cells on different ports are temporally aligned.

The three-stage switch architecture is shown in Figure 7. It is a simplified diagram with a small number of inputs/outputs ports. The first and third stages are implemented in electronic technology and the second stage is implemented in photonic technology. The first stage encloses l centralized shared memory packet switch modules, also referred to as *input sectors*. An input sector contains l buffers between n ingress ports and m egress ports as shown in Figure 7. The center stage, consists of m crossbar switches of dimension $l \times l$ with no buffers that interconnect the input sectors and outputs sectors. The third stage encloses l centralized shared memory packet switch modules, also referred to as *output sectors*. An output sector switch contains n buffers between m ingress ports and n egress ports as shows in Figure 7.

Overall, the switch has $N = n \times l$ external ingress and egress ports. An assumption is made that all links only transfer one packet per timeslot for the FBP switch. Based on this assumption, the ratio of the internal capacity to the external capacity, also referred to as *speed-up* is equal to the spatial speed-up $k = m/n$.

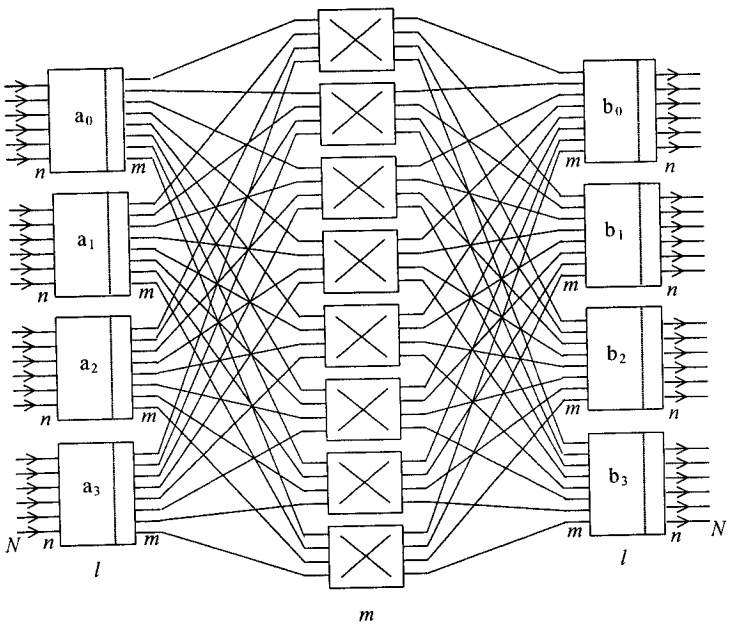

Figure 7 Three-stage packet switch with optical core for n=6, l=4, N=nl=24, m=8, k=1.333

The input sectors could use first-in first-out (FIFO) virtual output queues (VOQ) for packet buffering. The VOQ represents one queue for each output port at each input port. In this work, Virtual Output Sector Queues (VOSQ) are used instead, which correspond to one queue for each output sector within each input sector. Figure 8 shows the internal logical structure of an input sector implemented as a centralized shared memory switch. This diagram corresponds to input sector a_0 of Figure 7. The dual port random access memory is organized as a set of FIFO virtual output sector queues $Q_{i,j}$. The write bus is time divided between de-serialisers (one per input) that together act as the multiplexer shown at the input. The read bus is time divided between serialisers (one per output) that together act as the demultiplexer shown at the output.

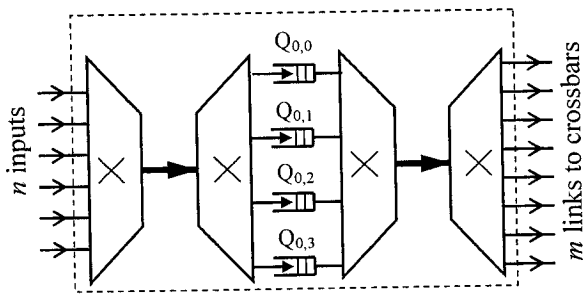

Figure 8 Input sector 0, a_0, for the switch in Figure 7. N=24, n=6, l=4, m=8. $Q_{i,j}$ is the VOSQ for traffic going from input sector i to output sector j.

The output sector contains straight output queues (OQ), which correspond to one queue within an output sector for each of its output ports. Figure 9 shows the internal logical structure of an output sector implemented as a centralized shared memory switch. This diagram corresponds to output sector b_2 of Figure 7. All queues Q_q are FIFO output queues

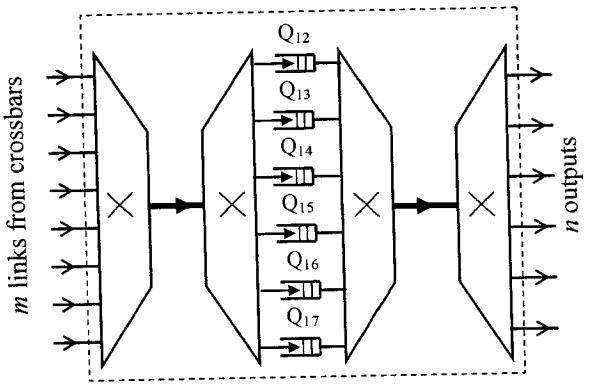

Figure 9 Output sector 2 for the switch in Figure 7. N=24, n=6, l=4, m=8. Qq is the dedicated queue for traffic going to output q, $0 \le q \le N-1$

The use of VOSQs (as opposed to VOQs) reduces the number of queues that must be managed from N^2, as in many widely used architectures, to $l^2 + N$ and makes the inter-sector scheduling policy less complicated. There are overall kN physical paths available in the central stage, which are shared among all the queues. The inter-sector scheduling policy will resolve the issue of which packets will depart amongst those at the head of queues in the input sectors.

FBP functionality can be achieved by varying the number of inter-sector links between the input/output sector pairs by appropriate configuration of the central stage crossbars. The total number of links incident to each sector is m. The configuration of pathways is determined by an edge-coloring problem solved on a bipartite graph that is defined by an inter-sector service rate matrix that may be derived from rate requests or estimated traffic arrivals.

3.2 Flexible Bandwidth Provision

Flexible Bandwidth Provision refers to the redistribution of the number of links internal to the switch, which are bundled together to form higher or lower capacity logical

connections between different end points. The redistribution in this work will occur at various points in time according to a set of rules and goals.

It is essential to have the ability to set up and remove link connections to achieve FBP functionality in a switch. The FBP switches therefore have the capability to manage the existing links and reconfigure their distribution to meet the incoming traffic demand. It involves three steps to reconfigure the FBP switch. First, an estimated *traffic matrix* is used to represent the incoming traffic. Then, a *service rate matrix* is used to represent the new link connections and, finally, use a *decompose* technique to find the components of the service matrix and obtain the appropriate crossbar switches configurations.

3.2.1 Estimated Traffic Matrix

The traffic matrix $\Lambda = [\lambda_{ij}]$, of size $l \times l$ represents the bandwidth requirement between input and output sectors. The traffic matrix is defined for sectors and not for incoming/outgoing links as in other works in packet switching. The row i has a value between $0 \le i \le l-1$ and corresponds to the input sectors. The column j has a value between $0 \le j \le l-1$ and corresponds to the output sectors. The elements λ_{ij} correspond to the number of packets to be transported from input sector i to output sector j.

It is not possible to know the precise real time traffic matrix. Consequently, an estimated traffic matrix can be calculated and used as an input to the scheduler/controller of the switch. The estimation is made with a simple running average calculated after every inter-configuration period, τ timeslots, using an adaptation speed variable, σ, that determines how quickly the estimated matrix will reflect the new patterns of traffic arriving:

$$\Lambda_e \leftarrow \sigma\Lambda_c + (1-\sigma)\Lambda_e \qquad (1)$$

where Λ_e is Estimated Traffic Matrix and Λ_c is the current offered load which is an integer matrix in number of packets that arrived at input sector i and destined for output sector j since the last configuration of the switch.

Below is an example of Λe for l=4, n=4, m=8, σ=0.9 and τ= 20:

$$\Lambda_e = \begin{bmatrix} 10.836 & 0.9 & 8.8128 & 32.859 \\ 1.5309 & 28.091 & 8.73 & 12.6 \\ 1.2249 & 2.8719 & 54.856 & 10.44 \\ 30.51 & 0.2808 & 12.682 & 12.6 \end{bmatrix}$$

The estimated traffic matrix indicates that the input and output sector pair that requires the most bandwidth is i= 2 and j=2 which has 54.856 packets. The input and output sector pair that requires the less bandwidth is input sector 3 and output sector 1.

3.2.2 Service Rate Matrix

The inter-sector service rate matrix, S, represents the actual bandwidth measured in link connections between the input and output sector pairs. All elements in the service matrix are integer numbers which represent the number of interconnection links between each input/output sector pair.

The links assigned by the service matrix must correspond to the estimated traffic matrix, Λ_e, as proportionally as possible. Both rows and columns summations of the service matrix, must be equal to the number of available incident links m, which is the number of crossbar switches in the central stage:

$$\sum_i s_{ij} = m, \sum_j s_{ij} = m$$

In order to obtain the service matrix S, the first step is to normalize Λ_e by the largest sum over rows and columns to form a sub-stochastic matrix $\bar{\Lambda}_e$. Therefore, all the sums over rows and columns will be equal or less than one. In the sample estimate traffic matrix, Λ_e, the maximum sum over rows and columns is 85.082 which is column 2, so:

$$\bar{\Lambda}_e = \frac{\Lambda_e}{85.082} = \begin{bmatrix} .12736 & .01057 & .10358 & .38621 \\ .01799 & .33017 & .10260 & .14809 \\ .01439 & .03375 & .64474 & .12270 \\ .35859 & .00330 & .14906 & .14809 \end{bmatrix}$$

A minimum service matrix can be constructed from the suitably re-scaled version of $\bar{\Lambda}_e$:

$$S_{\min} = \lfloor (m-l)\bar{\Lambda}_e \rfloor + \bar{\bar{1}} \tag{2}$$

where $\bar{\bar{1}}$ is a matrix of size $l \times l$ with all elements equal to one. This matrix corresponds to a set of permanent interconnection links that will provide minimum connectivity between each input and output sector pairs and prevent queues from ever being starved. If starvation of queues is not a concern, then $S_{\min}$ can be calculated simply as:

$$S_{\min} = \lfloor m\bar{\Lambda}_e \rfloor \tag{3}$$

For the sample Λ_e shown, the minimal service matrix calculated with equation (2) will be:

$$S_{\min} = \begin{bmatrix} 1 & 1 & 1 & 2 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 3 & 1 \\ 2 & 1 & 1 & 1 \end{bmatrix}$$

An iterative *rate-filling* algorithm is implemented to distribute the remaining interconnection links. It starts with $S = S_{\min}$, the Excess Matrix, X, is defined as:

$$X = S - \frac{1}{\tau}\Lambda_e \tag{4}$$

where τ is the inter-configuration period and the term $\frac{1}{\tau}\Lambda_e$ corresponds to the average number of packets to be transported every timeslot between sector pairs.

The excess matrix X represents the difference between the current service matrix and the traffic demand. The rate-filling algorithm finds the smallest element in X and provides an additional interconnection at corresponding position in service matrix, S. The algorithm updates the excess matrix X every time an additional link has been assigned to the service matrix. It also ensures that the associated row sum or column sum does not exceed the maximum number of crossbars *m*. The rate-filling algorithm stops updating S when every row and column sum of S is equal to *m*. At this moment, all available interconnection links have already been assigned. For the example being presented, the initial excess matrix will be:

$$X = \begin{bmatrix} .45815 & .95500 & .55936 & .35700 \\ .92345 & .59541 & .56350 & .37000 \\ .93875 & .85640 & .25716 & .47800 \\ .47450 & .98596 & .36586 & .37000 \end{bmatrix}$$

Since the minimum amount in X is 0.25716, the first additional interconnection will be assigned between input sector 2 and output sector 2. By continuing with the same procedure, the final service matrix will be obtained as:

$$S = \begin{bmatrix} 2 & 2 & 1 & 3 \\ 2 & 3 & 1 & 2 \\ 1 & 2 & 4 & 1 \\ 3 & 1 & 2 & 2 \end{bmatrix}$$

S in this particular example indicates that for input sector 0, 2 links are connected to output sector 0, 2 links to output sector 1, 1 link to output sector 2 and 3 links to output

sector 3. As required, there are a total of 8 links connected to each input and output sector.

3.2.3 Matching Algorithm

After the computation of service matrix S has been completed, the regular method for the FBP switch uses an edge-coloring algorithm to decompose the service matrix S into m permutations. However, the decompose procedure is unnecessary for the OPNET simulations constructed in this work. Since the reconfigurable crossbar switches do not exist in this module in OPNET, there is no need to perform the bipartite graph decomposition algorithm to break down the service matrix. The simplified pseudo code for the FBP algorithm is below.

Main FBP Algorithm:

1. Obtain the estimated traffic matrix, Λ_e.
2. Normalized the estimated traffic matrix, Λ_e.
3. Calculate the Service Matrix S.
4. Send updated Service Matrix S to all input sectors for appropriate configuration.

3.3 OPNET Implementation

OPNET Modeler [23] is a software package developed for the modeling and simulation of communications networks. In OPNET, a complete network structure includes various levels of models such as process models, node models, link models, network models, and external system models. A process model, which is located inside a node model, is the lowest level model and is used to describe the operations of processors and queues. The process models can be used for an extensive variety of functions, including communication protocols, algorithms, resource sharing, operating systems, queuing disciplines, specialized traffic generators, and custom statistic collectors. All network operations are defined at the process model level first and then placed on the node model.

A node model may include various processors, queues, packet streams, transmitters and receivers as needed to construct the node structure. Processors and queues define the operations made on packets in the node model and their logic is implemented as finite state machines with binary expressions that determine the transition between states. Packet streams determine the connectivity between processors, queues, transmitters and receivers. Transmitters and receivers define the number and format of input and output ports of the node. A link model is used to connect nodes in the network level. Different node models require their associated link model; for example, an ATM node model would not be compatible with the Ethernet link model. The correct association between link and node is essential.

The network model defines the overall structure of a simulated network system. It is the highest level, where objects like nodes and links in a network are specified, as well as their physical locations, interconnections and configurations. There are also external systems which are models determined by code that is external to OPNET. An external system interacts with the node model synchronously and can exchange data with it; however, neither the external system nor the node model has any implicit knowledge of each other's process data.

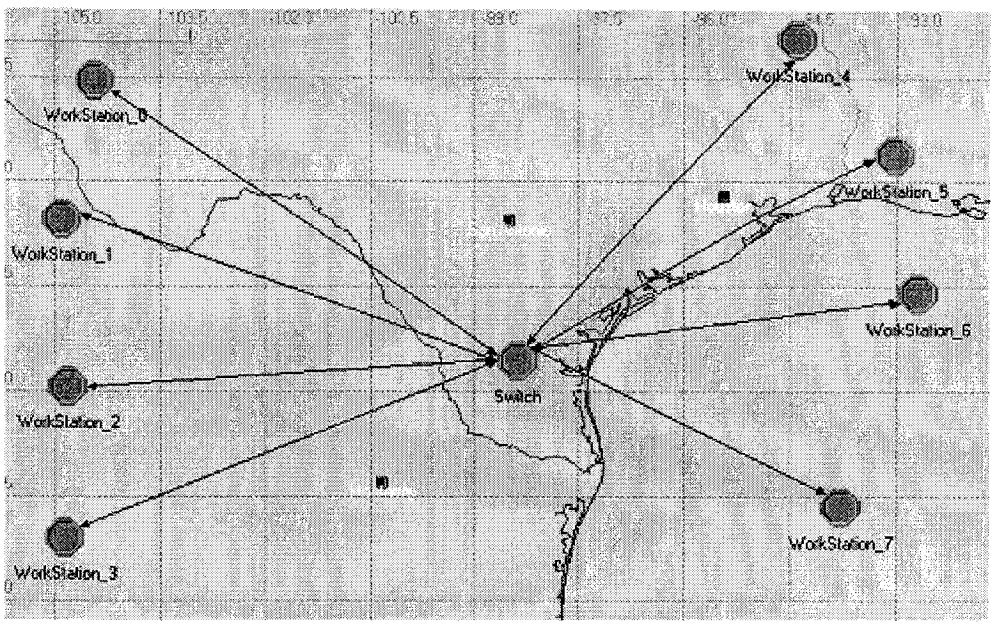

Figure 10 Network Model used to implement and test the FBP switch

A FBP switch module that operates under different traffic types has been built with the first objective of verifying the feasibility of implementing the FBP switch architecture in OPNET. The subsequent objectives were to obtain a comparison with the results obtained with the custom software written in the C++ language to simulate this architecture [1], and to setup a benchmark performance for further implementation and research on this topic.

The network topology consists of 8 workstations and one FBP switch. The switch contains 8 input sectors, 8 output sectors and 1 input/output port per sector that operates eight times faster than the internal streams of the switch and therefore represents 8 sources and 8 destinations, respectively. This implementation corresponds almost in its entirety to the 64x64 architecture used to gather most of the results presented in [1].

Figure 11 shows the FBP switch model at the node level. Each workstation generates traffic based on two distributions that define the packet arrivals and the packet destinations independently. All workstations generate and receive packets simultaneously. The propagation delay between workstations and switch was set to the speed of light because the link connections represent optical fibers in the simulation.

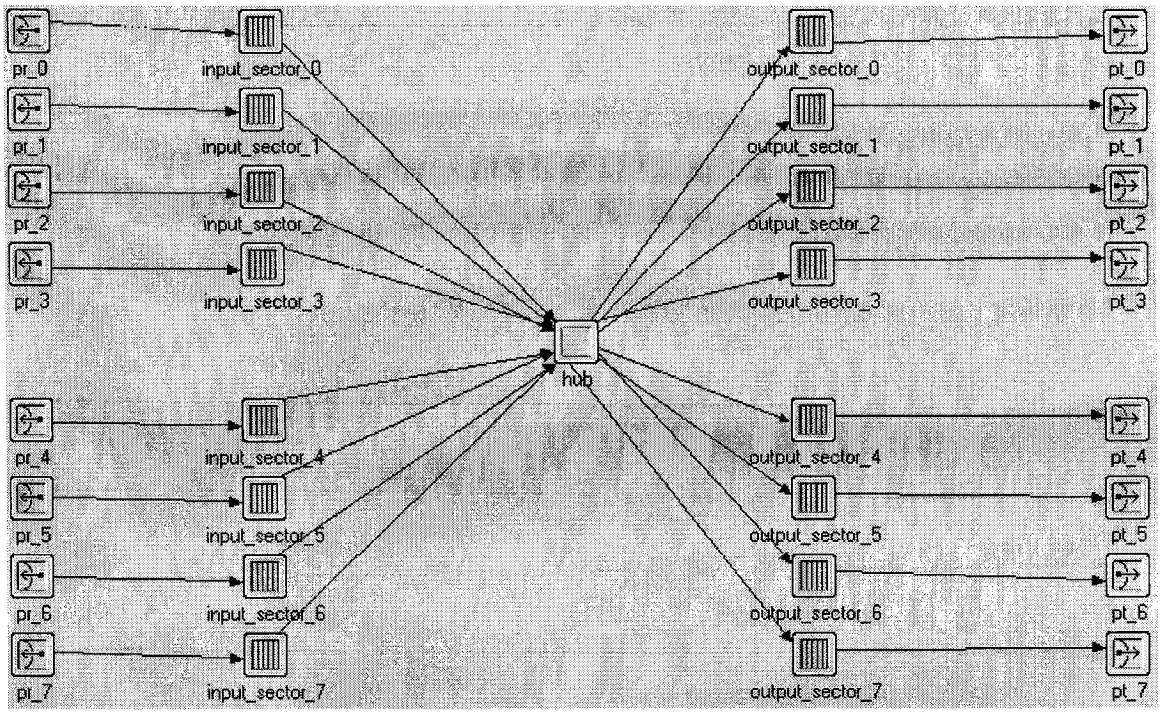

Figure 11 FBP switch model at the node level. The solid lines represent packet streams and the dotted lines represent the association of a transmitter with a receiver

Each input sector contains 8 VOSQs and each output sector contains 8 OQs. Both input and output sectors use finite state machines to define the queuing processes. The "*hub*" logically represents all 16 crossbar switches in the architecture. Given that is it not necessary to implement each of these switches separately to simulate the architecture at the network level, the hub is therefore a simple processor that receives and sends packets according to the *service matrix* calculated. Since the reconfigurable crossbar switches do not exist in this module in OPNET, there is no need to perform the bipartite graph decomposition algorithm to break down the service matrix.

In order to implement the FBP scheduling algorithm, each input sector transmits an *Internal Control Information* (ICI) packet to the hub to update the traffic matrix after every inter-configuration period, τ, defined in timeslots in this model. The hub processor calculates the appropriate service matrix and transmits ICI packets back to each input sector every τ seconds for them to update the service rates of the VOSQs. The input

sectors then release packets according to the updated service matrix for the following timeslots.

Figure 12 shows the state machine used for all input sectors. When a packet arrives, the first action is to obtain the destination address and then, based on this address, it is decided which output sector the packet belongs to. The packet is then queued at the FIFO queue associated with this output sector. Every timeslot, the queues will release a number of packets from their head (if there are any queued). The service matrix determines the number of packets that will be released from each of them in one timeslot. The released packets will be sent to the "hub" (as shown in Figure 11). Every τ seconds (one inter-configuration period), the traffic matrix information is updated by measuring the length of the VOSQs in each sector and this information is sent to the "hub" module for subsequent calculation of the service matrix.

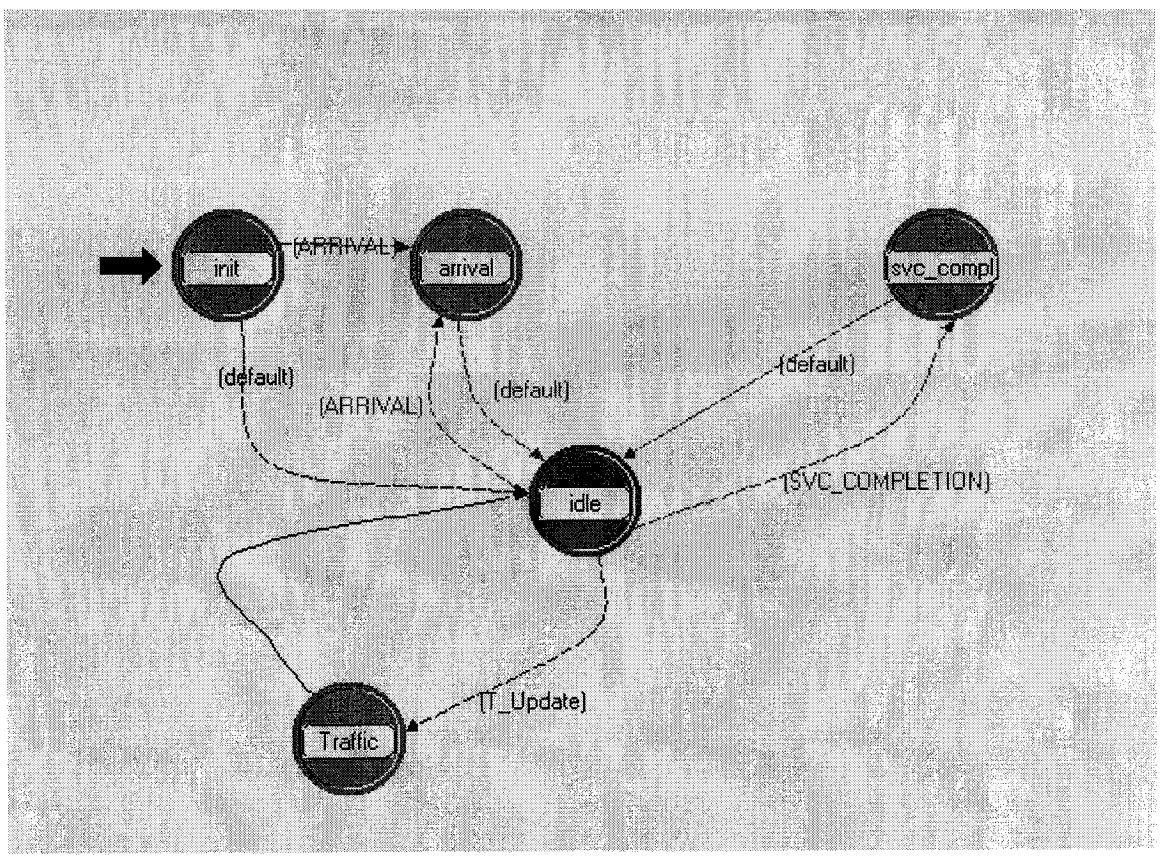

Figure 12 Input sector process model

Figure 13 shows the process model for the "hub" located inside the switch node model. This process performs the service matrix calculation, switch reconfiguration and ICI packet operations every τ seconds. At all other times, the hub is a simple "receive and send" based device that just examines the destination sector of the packet and forwards it accordingly. In practice, the central stage of the FBP switch does not perform any header processing, but this inaccuracy is irrelevant for the simulations presented in this work since it is concerned only with the performance at the network level. The "compute" state acts as the global controller for the network. It computes a new service matrix every τ timeslots. The "update" state collects the traffic matrix from the input sectors and delivers the new service matrix back to them. Both traffic matrix and service matrix transactions are made with control packets between the input sectors and the global controller. The "send" state simply forward incoming traffic packet into the desired edge node based on the destination address stored in header information.

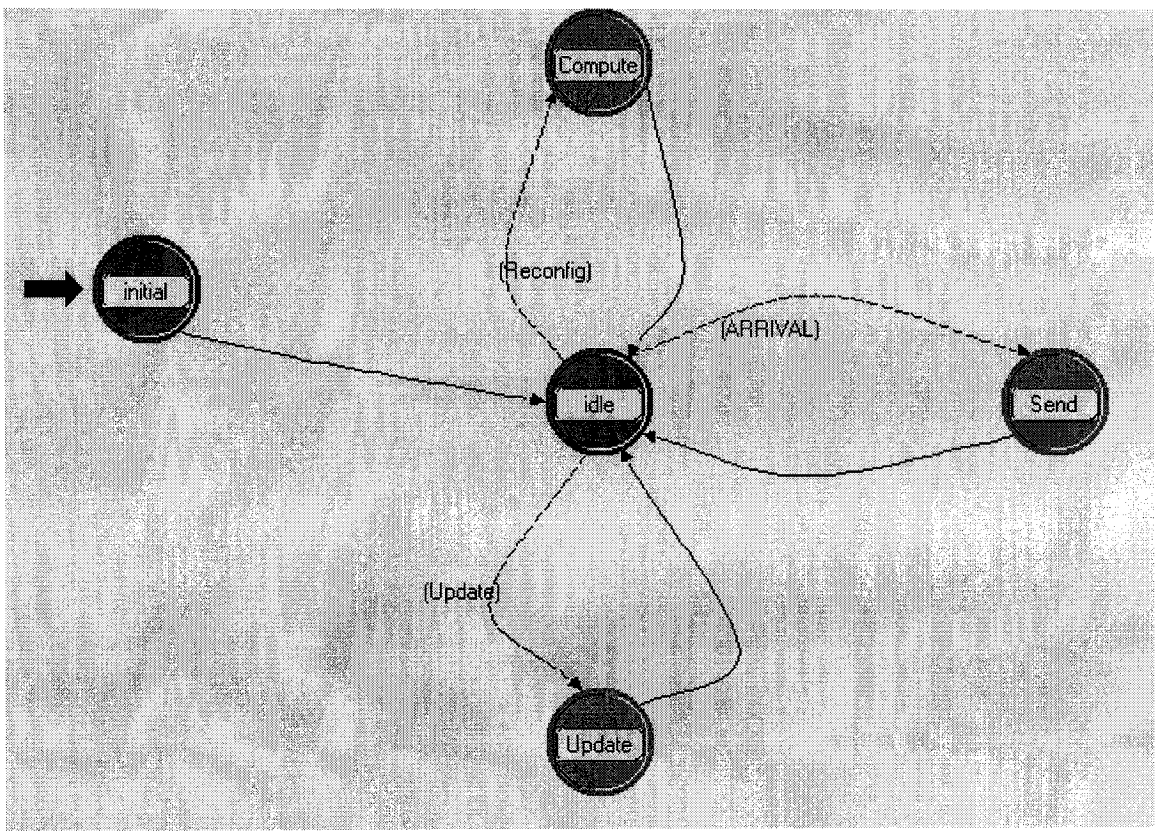

Figure 13 FBP "hub" process model

Figure 14 shows the workstation node model used for all workstations. It contains four objects: the packet generator (the "src" processor); the packet processor ("proc" processor) and the input and output ports ("rcv" and "xmt", respectively). The "src" process model generates packets based on a desired distribution for the interarrival times such as uniform or Exponential. The "proc" process model assigns packet destination address and determines end to end delay. The "xmt" and "rcv" objects are point-to-point transmitter and receiver, respectively, that connect the workstation to the FBP switch.

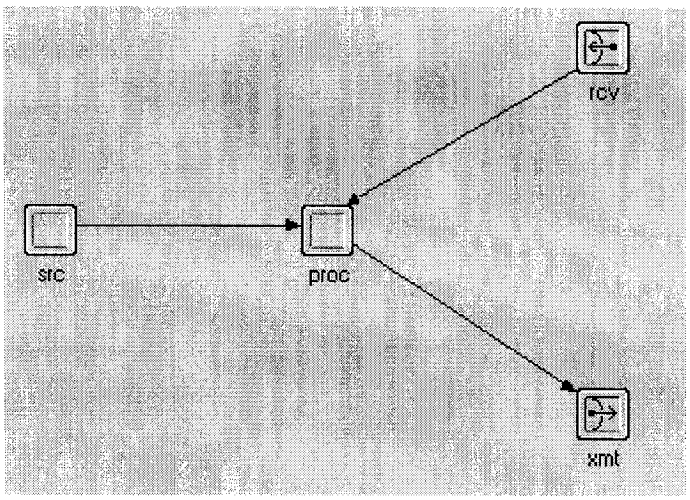

Figure 14 Workstation node Model

Figure 15 shows the process model used in the "proc" object at node level. The main functions of this model include assigning a destination address to the newly generated packets and determining the end-to-end delay for packets received at this node. The "rcv" state performs the computation of the end-to-end delay for every incoming packet. The end-to-end delay is measured as the time at which the packet reaches it destination node minus the time at which the packet was created. The SRC_ARRVL and RCV_ARRVL expressions correspond to the conditions used to verify whether the current packet was newly generated at the traffic generator or received from the FBP switch. Every T seconds (defined by the user in timeslots), the "T_update" state changes the bistochastic matrix used to generate non-uniform destinations for the generated packets and in this way a non-stationary arrival process is obtained.

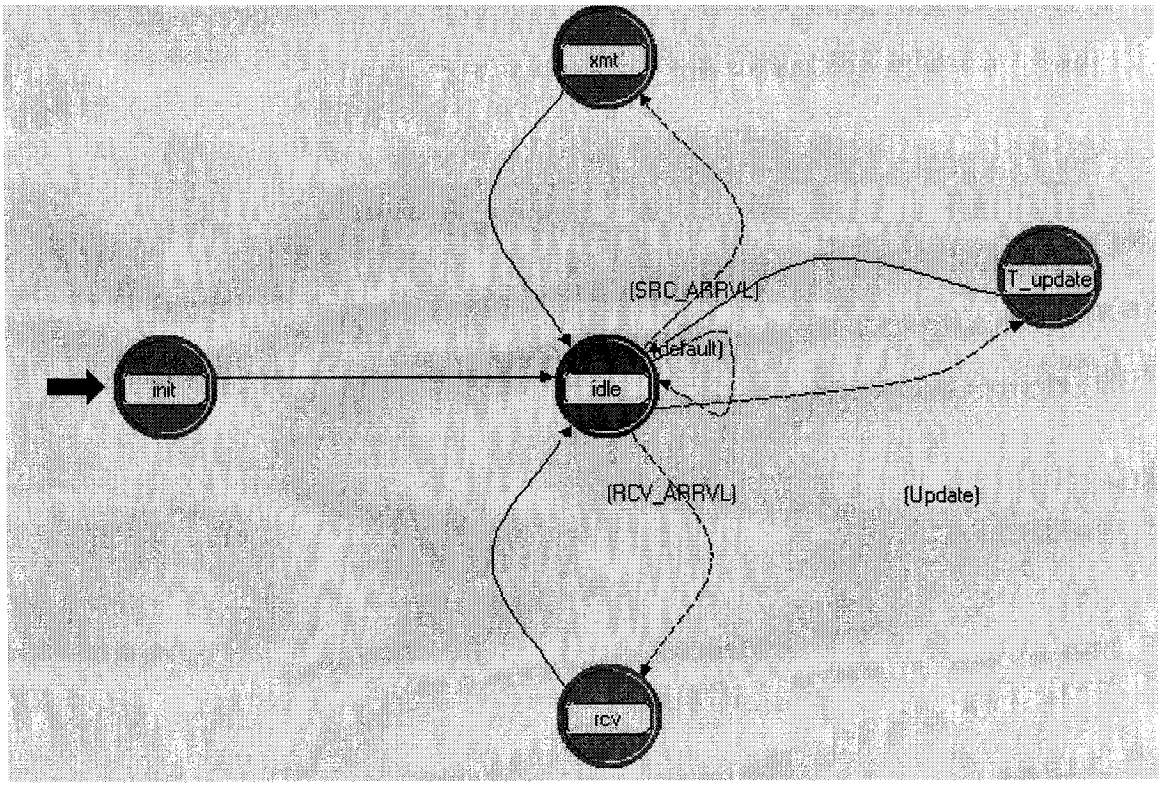

Figure 15 Process model in the "proc" object of the workstation node

3.4 Performance Analysis

The switch architecture and FBP method have already been demonstrated with self-similar and Poisson traffic under various test environments in [1]. The aim was to show how the number of paths provided between sector pairs with the FBP algorithm adapts to the changing traffic statistics to obtain 100% throughput and how the delays are kept close to the ideal output queuing switch performance as a consequence of this adaptation. As mentioned previously, the aim in this part of this work is to validate the previous results obtained with custom code and to establish a standard platform for the FBP switch to be later embedded in a more complex network environment in OPNET.

It is important to verify the differences in performance between the modules in custom C code and OPNET. One of the major differences in performance analysis is that the delay

measurement in the custom code considers all the output ports of the switch, which generates an average number from all sectors.

The simulated switch used for the performance analysis is a 64×64 switch with $M = 16, l = 8$, unless otherwise stated. Every sector has eight external ports and the internal speed up of the switch is $k = 2$. Each input sector has l queues which correspond to the number of output sectors and each output sector contains n queues for the corresponding workstations.

The queuing delay is the sum of both the input stage and output stage queuing delays. The queuing delay in the OPNET model has been measured separately for different sectors. Each sector has an independent queuing delay value. The results presented in this thesis are obtained from the average of five different simulations run with different random seeds. A random seed is an integer variable that can be chosen by the user and is the basis number used by the random number generator in OPNET.

The steady state response of the switch architecture with FBP is chosen to examine the switch performance. The steady state response refers to when the switch reaches stability (which is usually achieved in this context by running a simulation for an extend period of time).

3.4.1 Bernoulli Traffic

In this section, the traffic generated is modeled as a set of Bernoulli trials for each source. In this model, the probability that a packet will arrive at the incoming link at any one timeslot is ρ and the probability that no packet will arrive is $1 - \rho$. The aggregation of N Bernoulli trials yields a binomial probability distribution. The Poisson distribution is the limit of the Binomial distribution as N approaches infinity and the expected value stays unchanged. Therefore, the combination of all the arrival processes from all sources is a Poisson process. An Exponential distribution of the packets' inter-arrival times

generates Poisson packet arrivals at one source. Therefore, in this section, the Bernoulli packet arrival model uses the Exponential distribution from the OPNET *Distribution Package* [24] for the inter-arrival times.

3.4.1.1 Uniform Destinations

In this section, the probability distribution used to assign destinations to the generated packets is uniform (the OPNET built-in distribution from the Distribution Package is used) in order to evenly distribute traffic to all destinations. All destinations have a probability equal to 1/N since there are *N* destinations.

Figure 16 shows the average delay obtained with the FBP algorithm for a 64×64 switch with an internal speed up of two for Bernoulli traffic arrivals with uniform destinations and different values of inter-configuration period τ. The traffic estimation, $\overline{\Lambda}_e$, is calculated only with the length of the VOSQs. All sectors (and all outputs) have the same steady state delay performance as expected due to the uniform destinations distribution.

An increase of the value of τ will have only the slightest impact to the queuing delays because the uniform destinations distribution remains during the whole simulation and therefore the configuration of the central stage is not necessary a second time. The total queuing delay for a load of 90% is around 6 timeslots for all the values of inter-configuration period shown in the figure, which is consistent with the results shown in [2].

It is important to note that OPNET does not operate on a timeslot basis. Since time is measured in seconds; all data presented in this section has been converted to *timeslots* based on the basic time unit (the time taken to transmit one packet) in order to facilitate the comparison to the custom-code implementation of the switch.

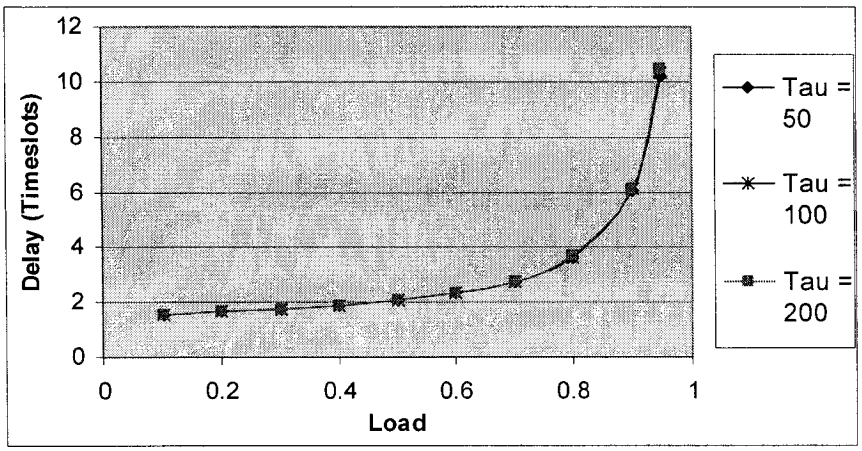

Figure 16 Average delay versus load obtained with a speed up of two for Bernoulli traffic arrival with uniform destination with $\tau = 50, 100, 200$

Most of the queuing takes place in the output stage when the internal speed-up of the switch is two (and for short inter-configuration periods). Figure 17 shows the total average delay (which includes the delays in both input and output sectors) and the input sector delay for an inter-configuration period equal to 50 timeslots. The figure clearly shows that the queuing delay in the output sectors dominates the overall delay. The input sector queuing delay for a load of 90% is around 0.6 timeslots only and the total queuing delay for a load of 90% is around 6 timeslots.

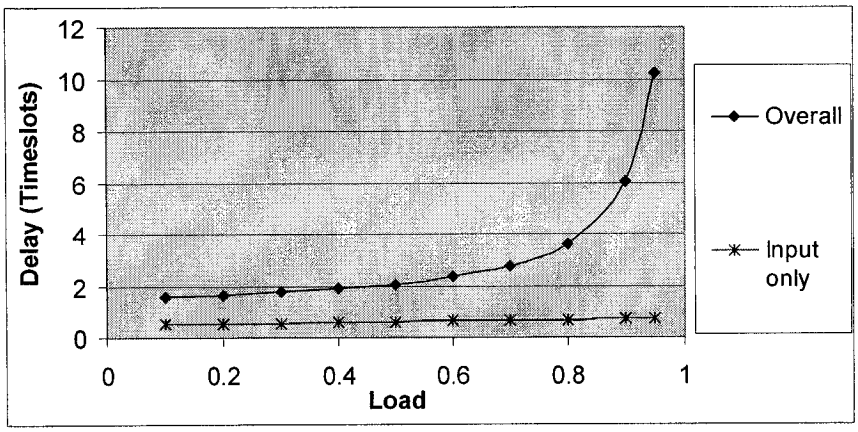

Figure 17 Average delay and input queuing delay versus load obtained with a speed up of two for Bernoulli traffic arrivals with uniform destinations for $\tau = 50$

Figure 18 shows the average delay obtained with the FBP algorithm for a speed up of one and different values of inter-configuration period τ. The traffic estimation, $\overline{\Lambda}_e$, is calculated only with the lengths of the VOSQs. Most of the queuing takes place in the output stage for short inter-configuration periods. All sectors have similar delay performance as expected due to the uniform destinations distribution. Changing the value of τ will have only a minimal impact to the queuing delays since the traffic is uniformly distributed. The total queuing delay increases dramatically for loads over 60% traffic load, which indicates that this is the maximum throughput of the switch when the internal speed up is one.

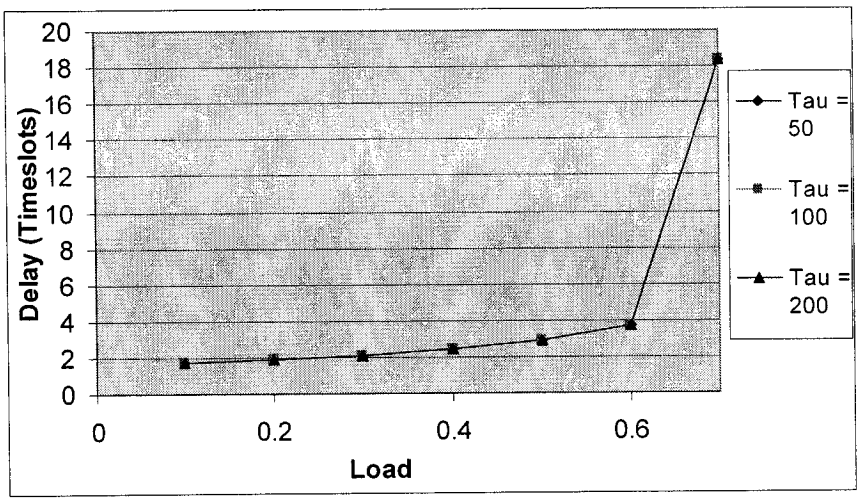

Figure 18 Average delay versus load obtained with a speed up of one for Bernoulli traffic arrivals with uniform destinations for τ = 50, 100, 200

3.4.1.2 Non Uniform Destinations

Since it is not characteristic of real traffic to have uniformly distributed destinations (balanced traffic), non-uniform probability distributions are used in this section to assign destinations to the generated packets and therefore produce Bernoulli traffic that is unbalanced and non-stationary. For this purpose, a random bistochastic matrix B of size $N \times N$ is calculated:

$B=[b_{pq}]$

$0 \le p \le N-1$, $0 \le q \le N-1$

$\sum_{p} b_{pq} = 1, \sum_{q} b_{pq} = 1$

The element b_{pq} is the probability that a packet from incoming port p will be destined to outgoing port q. Each row corresponds therefore to the probability distribution of destinations for source p. An example of a bistochastic matrix with N = 8 is:

$B =$

0.0358	0.0000	0.0533	0.1423	0.0003	0.2371	0.0002	0.5311
0.0000	0.2286	0.0018	0.3291	0.0142	0.2022	0.2006	0.0235
0.2345	0.2152	0.1373	0.0369	0.0303	0.0080	0.3377	0.0000
0.3441	0.2063	0.0002	0.2072	0.0695	0.1312	0.0415	0.0000
0.0008	0.0074	0.0408	0.0050	0.4669	0.0004	0.2171	0.2617
0.0065	0.0005	0.2310	0.0317	0.2170	0.3379	0.0405	0.1349
0.2410	0.0091	0.5347	0.0007	0.0008	0.0446	0.1624	0.0067
0.1373	0.3328	0.0009	0.2471	0.2010	0.0387	0.0000	0.0421

Every T timeslots, a new bistochastic matrix is created to update the destinations. This method generates non-stationary traffic that is admissible since both the row sums and the column sums are equal to one. The traffic matrix does not have long range dependence and shows no prolonged periods of inadmissibility. The expected traffic matrix is, however, uniform over very long periods of time.

Figure 19 shows the average delay obtained with the FBP algorithm for a 64×64 switch with a speed up of two for Bernoulli traffic arrivals with non-uniform destinations and different values of inter-configuration period τ. The traffic estimation, $\overline{\Lambda}_e$, is calculated as specified in equation 1. For the higher loads, the average delay increases as the inter-configuration period τ increases. This is the result of the VOSQs building up until a new configuration is applied, which will change the service rates of the VOSQs and therefore

empty them. The queues will build up larger if reconfiguration is less frequent and therefore the delay in the input stage increases. Most of the queuing takes place in the output stage for short inter-configuration periods. The only difference between Figure 19 and Figure 16 is packets destination assigning. The switch performance shows the same results as for traffic with uniform destination (Figure 16) when the interconfiguration period is small (τ=50) and a higher delay when the time between successive reconfigurations is longer (τ=100,200).

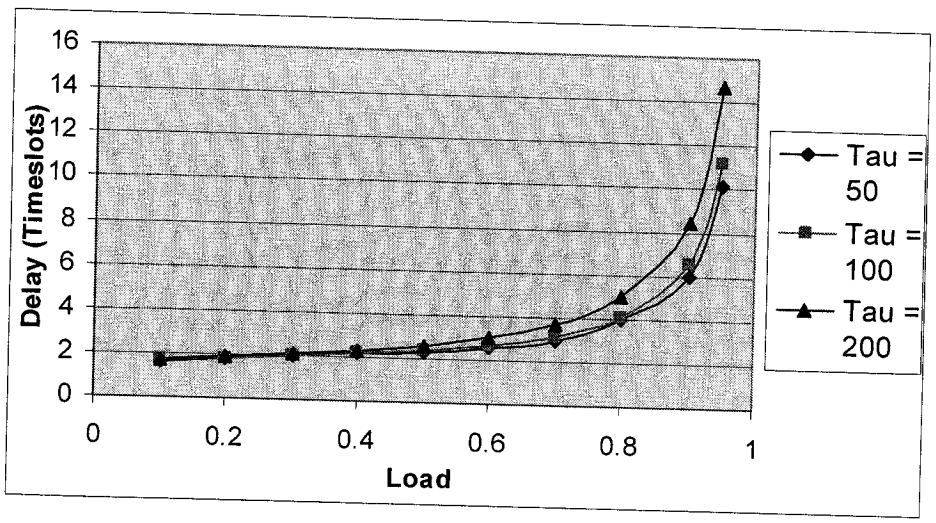

Figure 19 Average delay versus load obtained with a speed up of two for Bernoulli traffic arrival with non-uniform destination with τ = 50, 100, 200

3.4.2 Pareto Traffic

In this section, all traffic packets are generated using the built-in *bursty source* model in OPNET. This model defines the source as having an ON and an OFF state for generating packets, with both of these states being Pareto distributed random variables. The aggregation of these sources yields a process with long range dependence and a bursty traffic trace [18] [24]. The ON state corresponds to the time when packets are generated and the OFF state corresponds to the silent time; i.e. when the source generates no packets. The average estimate of the traffic matrix generated by this distribution is

inadmissible in short periods of time; however, the distribution yields a stationary process over long periods of time.

Figure 20 shows the average delay obtained with the FBP algorithm for a 64 x 64 switch with a speed up of two for Pareto traffic arrivals with non-uniform destinations and different values of inter-configuration period τ. The variable alpha is set to 1.4 and beta is set to 1.5 for all Pareto traffic trace simulations. The interarrival rate is set to 100% traffic load all time and therefore it is the various lengths of the ON and OFF states which change the traffic load as desired. The traffic estimation, $\overline{\Lambda}_e$, is calculated with the packet length of all queues. Most of the queuing takes place in the output stage for short inter-configuration periods as in the previous experiments. The total queuing delay for a load of 90% is around 7.5 timeslots for τ equal to 100. The queuing delay for Pareto traffic arrivals with non-uniform destination distribution is 1.5 timeslots higher than the queuing delay for Bernoulli traffic arrivals with non-uniform destinations distribution for τ equal to 100.

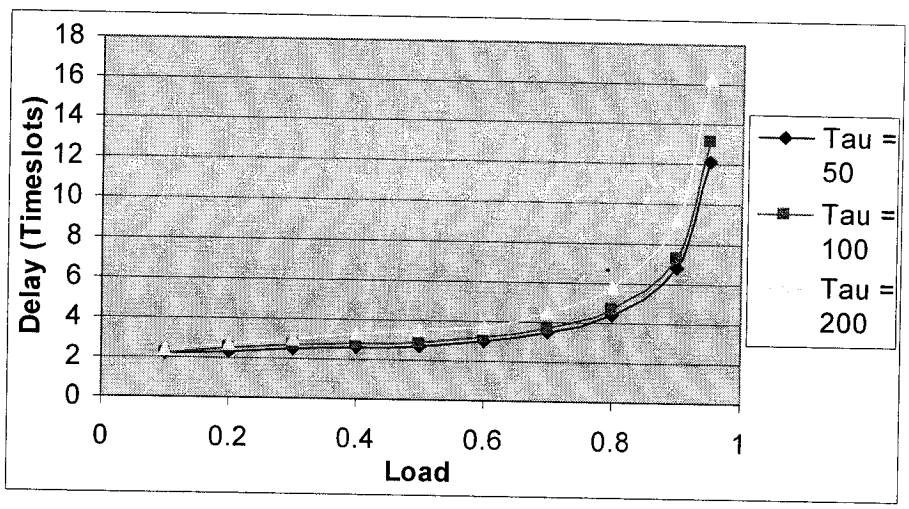

Figure 20 Average delay versus load obtained with a speed up of two for Pareto traffic arrival with non-uniform destination with τ = 50, 100, 200

Figure 21 shows the average delay obtained with the FBP algorithm for a 64 x 64 switch with an internal speed up of one for Pareto traffic arrivals with non-uniform destinations and different values of inter-configuration period τ. The traffic estimation, $\overline{\Lambda}_e$, is

calculated with the packet length in the all queues. All sectors have similar delay performance due to the uniform destinations distribution. Because of the values used for the generation of traffic, the arrivals process turned out very slow compared to the operation of the switch and this is the reason why changing the interconfiguration period does not have a big impact in the overall delay. However, it can be seen that a small increase in traffic load (e.g. from 10% to 20%) has a larger impact on the delay when comparing to the measures obtained for an internal speed up greater than one..

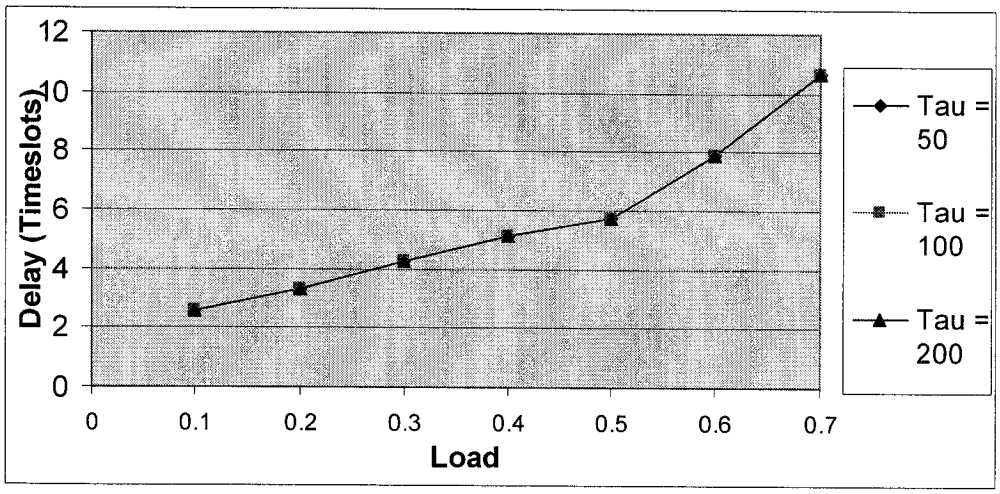

Figure 21 Average delay versus load obtained with a speed up of one for Pareto traffic arrival with uniform destination with τ = 50, 100, 200

3.5 FBP Switch with IP Traffic

The FBP switch module has been successfully built in OPNET. The next step is to introduce this module into an IP network router. This step is a critical one in order to achieve the goal of evaluating the performance of the FBP switch under more realistic network and traffic models.

The IP traffic model in OPNET includes multiple profiles that contain various applications and each application can involve several distributions such as Poisson or Exponential. An application is defined first by user to decide the operation parameters

such as database access and web browsing. A profile describes user activity over a given period of time. A profile may include many different applications. An OPNET IP traffic trace is shown in Figure 22. .

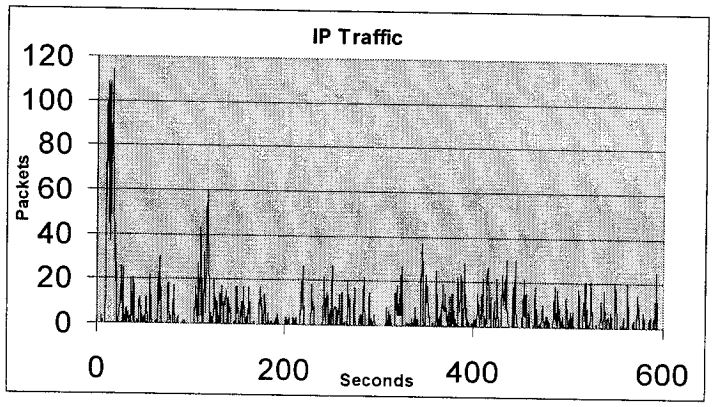

Figure 22 Sample IP traffic trace in OPNET

The main challenge of this implementation is that the existing IP router built-in model in OPNET uses several external system functions to perform various tasks such as IP look-up and transmission of packets. Each system function only has partial data access and has no knowledge about other information in the rest of the functions. Figure 23 shows a typical IP router model in OPNET. The blue lines are the upload packet streams and the red lines are the download packet streams. It can transmit different layers packets such as layer 2, Mac address packets, through Ethernet links and layer 3, IP address packets, through point to point links. The ARP0 and MAC0 processor models convert layer 2 packets into layers 3 packets when the switch gets packets from the receivers and convert the Layer 3 packets to Layer 2 packets for transmission. The IP process model is the central process of the switch and all packets traversing the switch have to pass it as the diagram shows. All packets are first sent to the IP process model and it will read the header information from each packet to determine the routing protocol such as OSPF, TCP or IP. Then the IP process model transmits the packet upwards to the appropriate routing protocol process model. A longest prefix matching will be performed for the destination address in the protocol process model to determine the correct egress port for

each packet. It will then transmit the packet back to the IP process model and use an external function to process the packet and send it to the appropriate egress port.

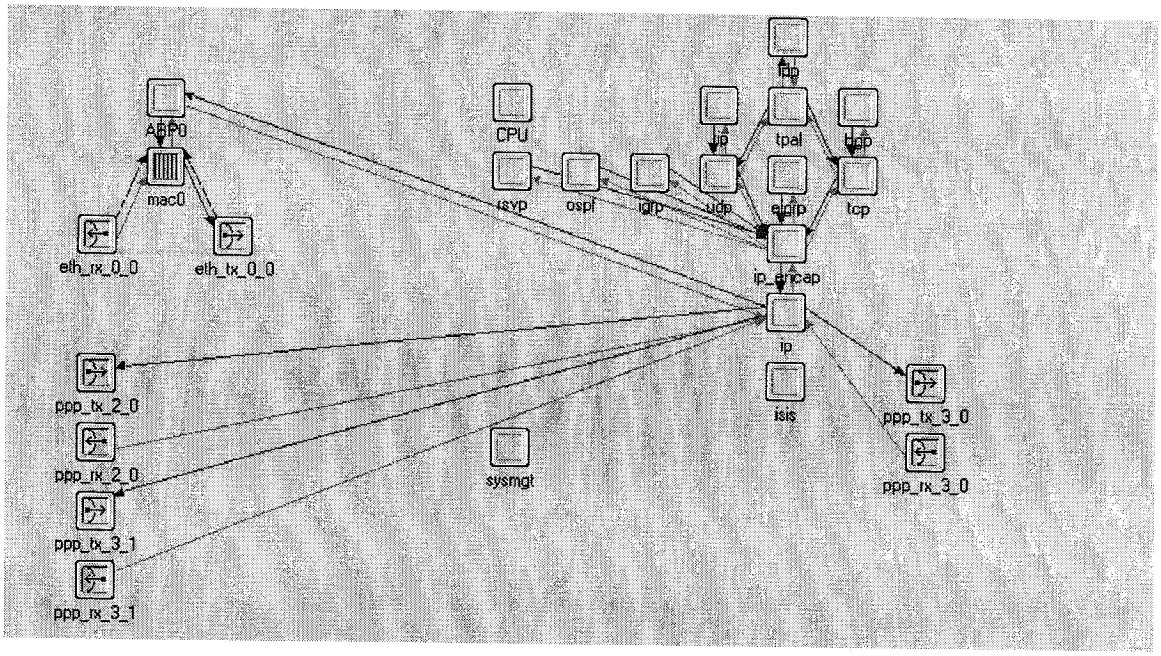

Figure 23 Typical IP Switch node model

Figure 24 shows the "IP_dispatch" process model which is used in the IP switch node model as the IP processor. The process built by the OPNET team as a standard internet traffic process also invokes numerous external function files for packet processing. It can be configured for many different purposes such as IP routers, MPLS routers, gateways, workstations, and etc. The process performs a self-verification to determine current node configuration due to the complexity of the process. Once the traffic arrives, it first initializes the IP address package, global address table, routing table, gateway, firewall and many other parameters in order to ensure that the router is prepared to handle the incoming traffic. It then initializes the redistribution of routing information between the routing protocols configured on this node and generates a network topology and a global routing table. It then composes a link interface table for all ingress and egress ports at this node connected to the IP process.

The process gains various packet statistics such as destination address, the routing protocol to be used, offset number, time to live and etc. from the packet header. The processes itself also determines various information such as the number of routing protocols running on this interface, running custom routing protocol on this interface or not, process the default route if it is specified and etc. stimulatingly. Many operations are executed by external processes..

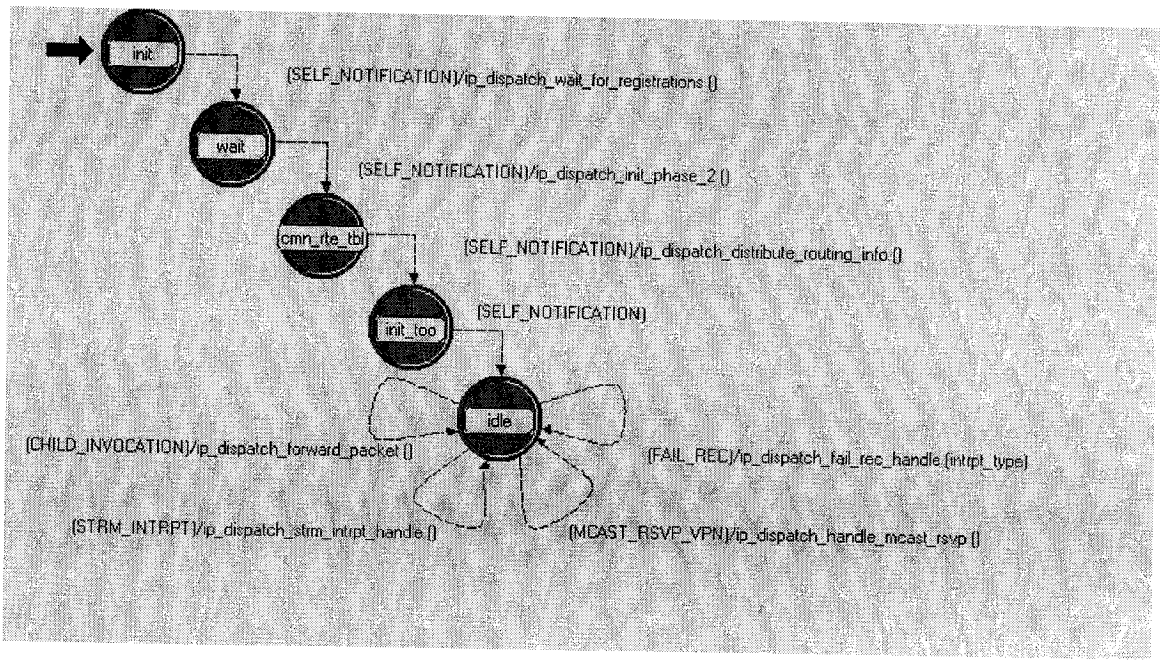

Figure 24 IP_dispatch process model

A preliminary implementation of the IP router with FBP functionality has been built in a small IP network in OPNET. It has been very challenging to direct the packets to the desired sector/port given the limited information that can be extracted from the built-in IP processes. The IP process model has access to the routing table but it cannot modify the table itself. Therefore, a small fixed routing table has been used in an external function to ensure that the packets are transmitted to the desired paths. The IP network implemented based on this approach is shown in. Figure 25

There are two types of traffic running in this network. Layer 2 traffic packets run between the workstations, the MAC switch and the local IP router. Layer 3 traffic

packets run between the local IP router, the IP cloud and the FBP router. The FBP router only receives IP traffic for this simulation network.

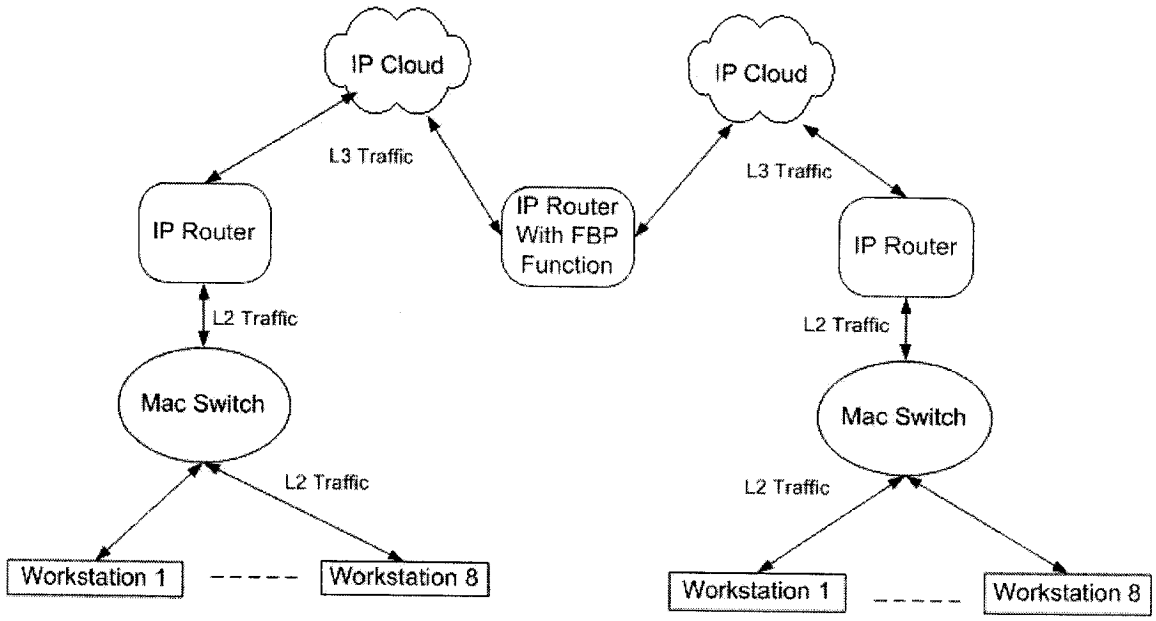

Figure 25 logical IP network

Figure 26 shows a block diagram for the FBP router. It uses all the typical IP router model processes such as the "IP_encap" and, in addition, the input sector, hub and output sector process models. All the three latter process models have the same functions and performances as describe in the previous section 3.3. In this implementation, however, each input sector contains only 2 VOSQs and each output sector contains only 2 OQs. The hub represents 4 centre crossbars in this simulation. The input traffic packets will go through the exact the same processes as in OPNET's built-in IP router until the packets are ready to be sent to the appropriate output ports. The IP process model then reroutes the packets into either input sector 1 or input sector 2 based on the source address of the packet.

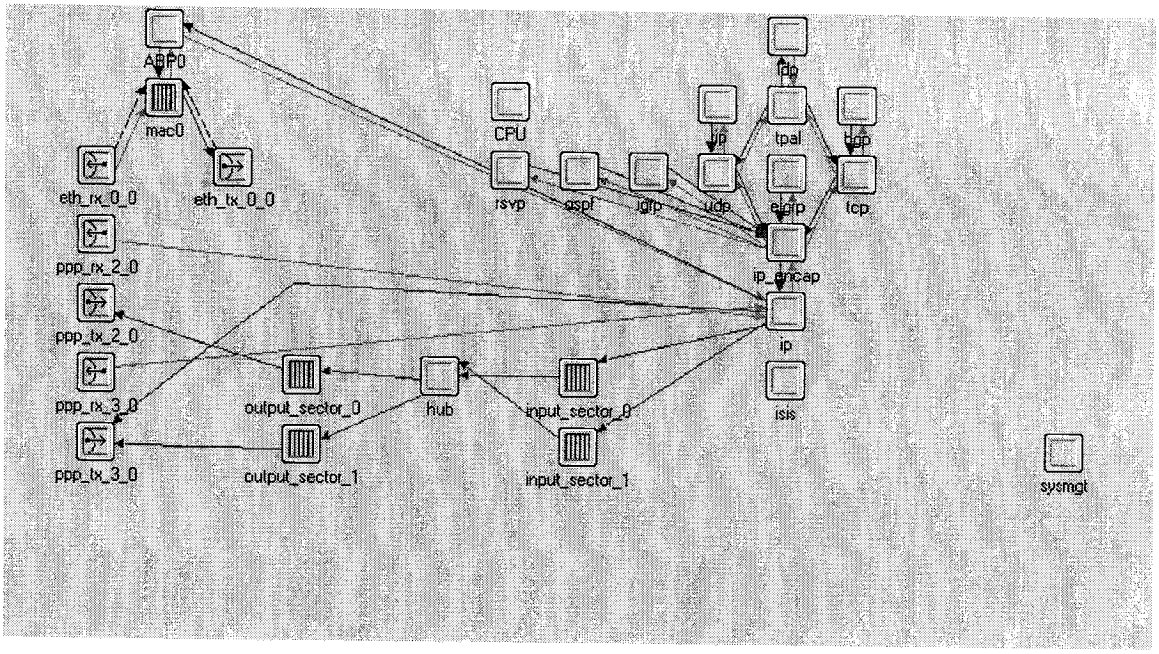

Figure 26 IP router with FBP functionality

Work on the IP router with FBP functionality was postponed due to software license issues. The OPNET Modeler version 9.1 is no longer available and the new OPNET modeler version 10.5 has a different IP built-in model. All the old FBP models previously built to operate with the IP built-in models in version 9.1 are not compatible with the new version 10.5 and therefore it has not ben possible to obtain simulation results and performance analysis for this thesis.

4 Agile All-Photonic Network

Continuing the research with the FBP switch, it is possible to extend the concept of the FBP switch architecture to map it onto an optical star network. The first and third stages together become the edge nodes and the centre stage becomes the core node. The resulting star network is analog to the fundamental architecture of the Agile All-Photonic Networks (AAPN) research effort [19] and, for this reason, it is feasible to attempt such a mapping.

This chapter first discusses the architecture of the AAPN research project. Next, the mapping from the FBP switch to AAPN network architecture and the implementation of an AAPN network in OPNET is described. Finally, the performance analysis of the bandwidth provision method in AAPN is presented.

4.1 Agile All-Photonic Network Topology

AAPN proposes an overlaid-star topology with edge-nodes interconnected by the beams of the stars via core switches located at the hub of each star. The beams usually consist of two or more fibers that transport wavelength division multiplexed (WDM) traffic between edge-nodes and the bandwidth of each wavelength is shared in time by assigning varying numbers of timeslots to each flow of data. The core switches operate on a slotted mode to forward incoming traffic. It must be possible to create and destroy paths sufficiently rapidly that they endure for only microseconds.

A bandwidth sharing scheme must introduced for the network to adapt its configuration to meet the demands of changing traffic patterns. It may also provide an option to edge nodes to choose distinct routes with different propagation delays, which can relieve the routing of delay-critical traffic.

Figure 27 and Figure 28 provide illustrative examples of a one-star and a two-star AAPN network, respectively. The total number of edge nodes is much larger in practice (usually hundreds to one thousand). All edge nodes are not necessarily connected to all core nodes; however, it is essential to make at least one star connected to all edge nodes and all edge nodes connected at least to one star. The number dimensions and placement of core and edge nodes is determined by demographic and economic factors.

The ratio of the internal network capacity to the external network capacity is known as speed-up. Buffers are in general required at the destination edge-node when the speed-up is greater than unity.

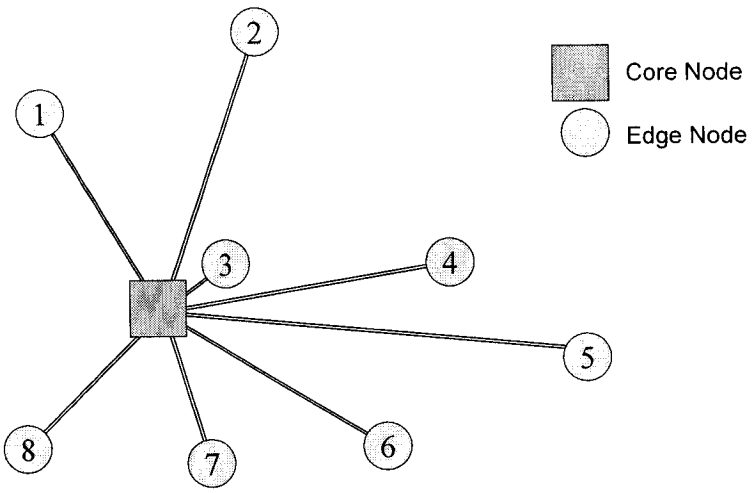

Figure 27 A one- star network with eight edge-nodes and one core-node

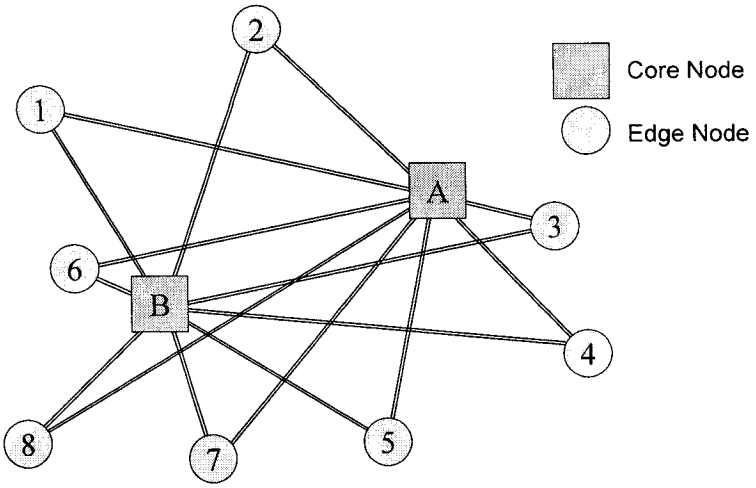

Figure 28 An overlaid two-star network with eight- edge nodes and two-core nodes

4.1.1 Edge-nodes

Each edge-node may be decomposed into its source and destination functions as shown in Figure 29. All source-destination pairs are essentially independent; however, it is feasible that two or more pairs share memory to facilitate self-directed traffic that has thereby no need to be routed via a core node.

Each edge node preserves a distinct buffer for each destination edge node. Consequently, all packets that have the same destination will be queued in the same buffer for a particular edge-node via a particular core node. There could be more than one source connected to a particular edge node but the traffic is organized according to the destination, regardless of the source. Each edge-node maintains a separate clock but all clocks are synchronized for all core nodes via a suitable protocol. It is also required that

the edge switches perform accounting and statistical information gathering. As shown Figure 29, the traffic information needs to be updated.

The edge nodes use a schedule to release packets from the buffers associated to a particular core node. The edge node has the ability to adjust its schedule to compensate for propagation times from the edge node to the core node. Figure 29 shows a symmetrical arrangement edge node. It illustrates a star network with four edge nodes, each containing four buffers. The edge node uses WDM to forward traffic to the core node. Each destination edge node maintains a distinct buffer for queuing received packets from a particular edge node via a particular core node.

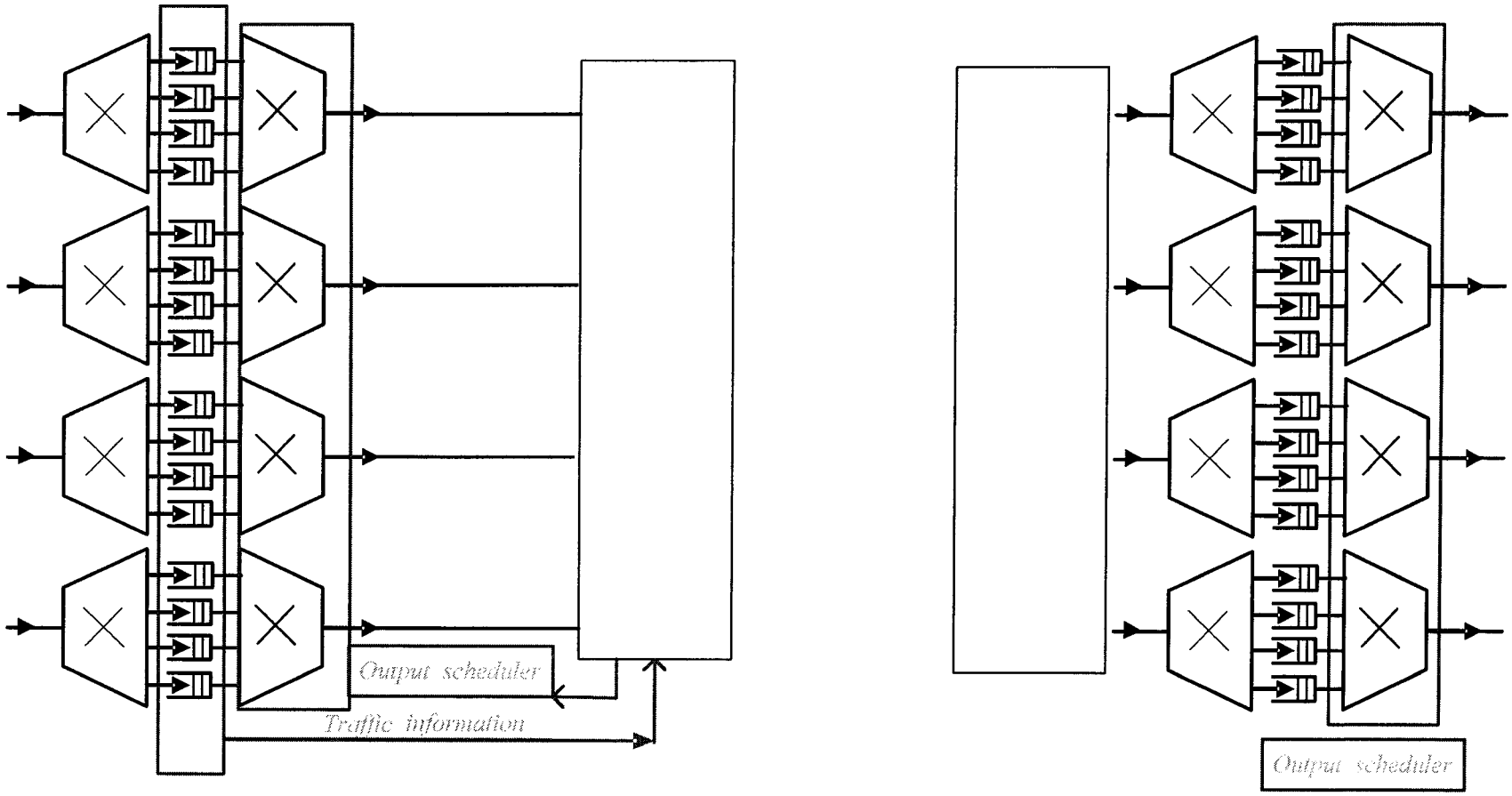

Figure 29 A one star network with four edge nodes and a single core node with the detail of the source and destination functional blocks shown separately.

4.1.2 Core nodes

Figure 30 illustrates the stacking of crossbars within the core node's switch; there is one for each wavelength. All incoming packets from the source nodes are dispatched to the destination nodes without any queuing involved in the core node. The de-multiplexers receive the incoming optical signals from the source edge nodes and split them into their various wavelength channels. Subsequently, each wavelength is space-switched by a dedicated crossbar. The multiplexers merge all separate wavelengths again and forward the optical signal to the edge nodes.

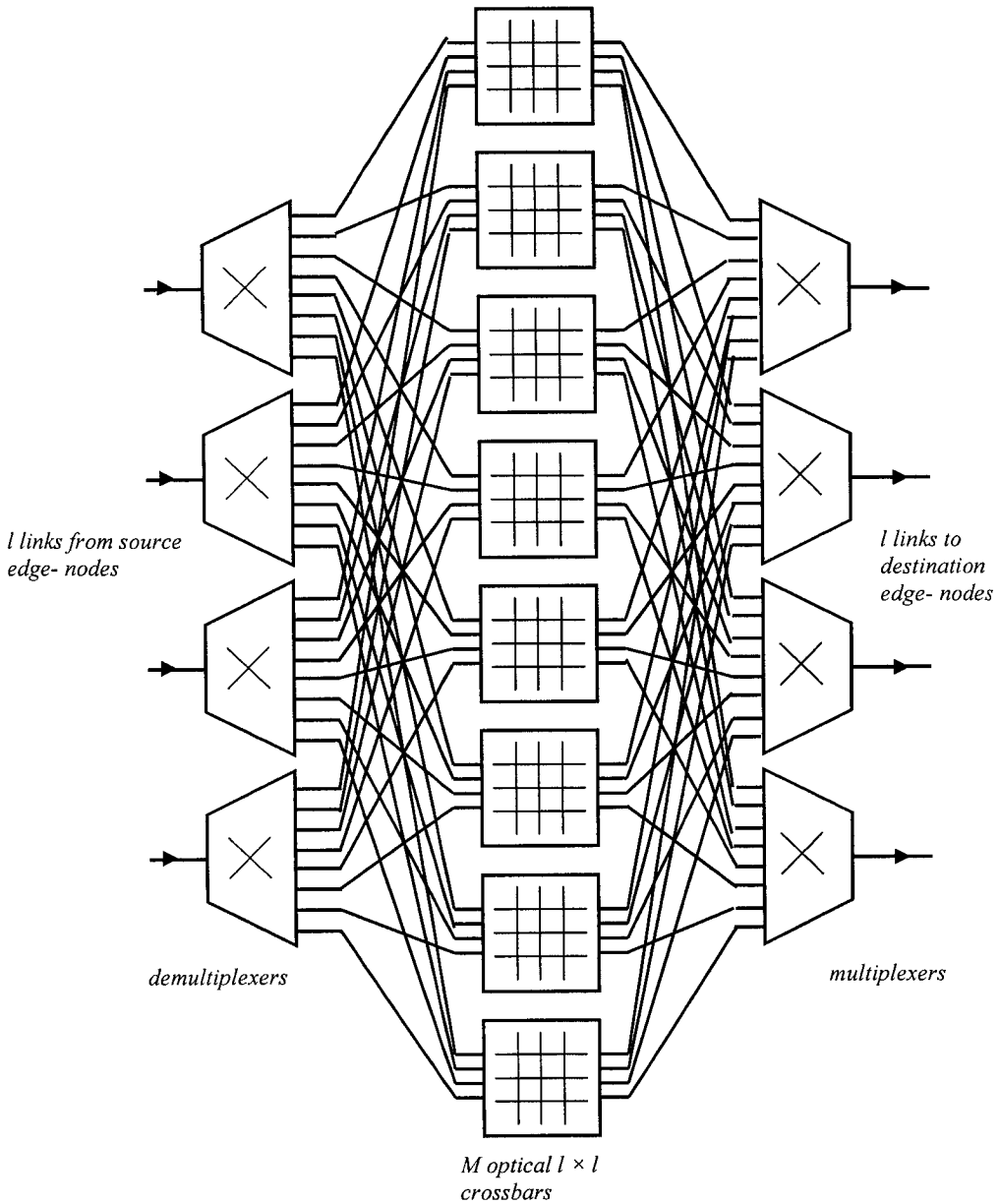

Figure 30 Detail of the core node showing the stacking of the crossbars within the node

4.2 Mapping the FBP switch architecture onto AAPN

As mentioned previously, the AAPN is a star network topology as shown in Figure 31(a). Each edge node consists of a source module (shown in gray) which receives ingress traffic and a destination module (shown in white) which forwards egress traffic. The

connections between the edge node modules and the core node are in the optical domain with unidirectional transmission once the source and destination modules have been (logically) separated.

With this view in mind, the edge node could be logically split into two parts as shown in Figure 31(b). The gray circles still represent the source modules of the edge nodes and the gray circles represent the destination modules of the edge nodes. The source edge-nodes transmit traffic to the core network and the destination edge nodes receive traffic from the core network. The connection between edge nodes and core node remain in the optical domain and unidirectional.

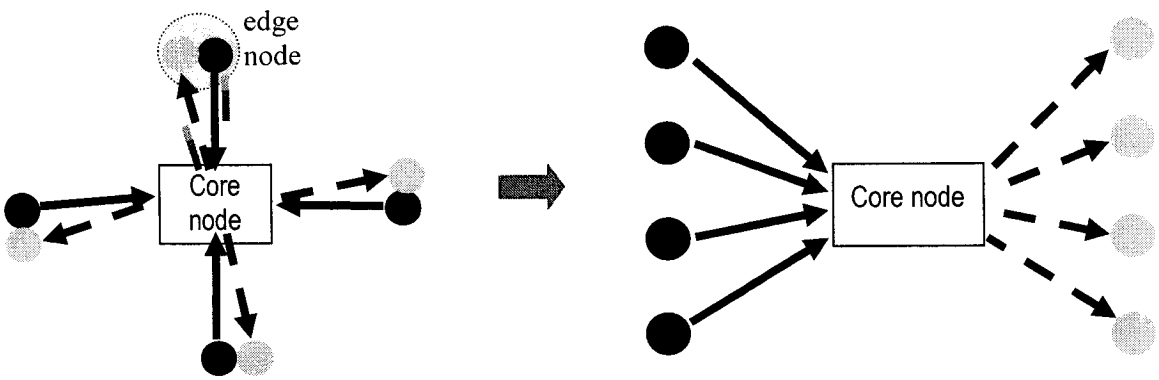

Figure 31 AAPN Topology revisited. (a) Edge nodes consist of a source and a destination module, (b) Equivalent three-stage architecture The architecture becomes a three-stage network if the edge nodes are logically split in source (first stage) and destination (third stage)

Substituting the core node with its constituent elements (a stacking of crossbars, one for each wavelength) as shown in Figure 32(b), the source edge-nodes connect to the demultiplexers and the multiplexers connect to the destination edge-nodes. All connections remain optical and unidirectional. Since the source edge-nodes maintain one queue for each destination node, this architecture is identical to the FBP switch architecture, with the source nodes being equivalent to the input sectors and the destination nodes being equivalent to the output sectors. The first and third stages

correspond to the sets of edge nodes (source and destinations) and the center stage corresponds to the core node.

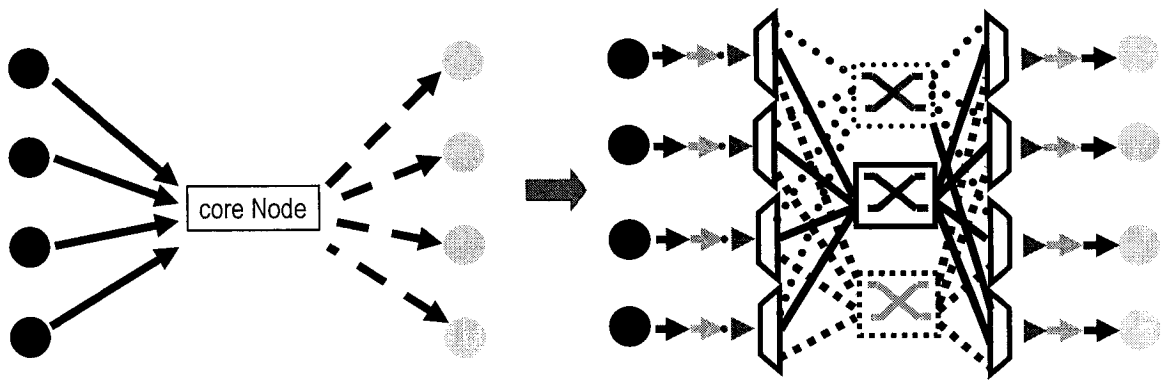

Figure 32 Logical equivalence between the AAPN topology and the FBP switch. (a) Three stage network, (b) The core node consists of a stacking of crossbars, one for each wavelength.

Given that the two architectures are logically identical, it is inviting to use and test the same bandwidth allocation method, FBP, in the AAPN architecture. In the context of AAPN, the algorithm has been renamed *Adaptive Wavelength-Timeslot Allocation* (AWTA). Several considerations need to be taken into account to obtain an adequate mapping of FBP onto AWTA; including the new traffic sources, link speeds, number of channels available and, most importantly, the propagation delays (in transmission between the edges and the core), which were negligible at the switch level.

4.3 Adaptive Wavelength-Timeslot Allocation (AWTA) algorithm

The Adaptive Wavelength-Timeslot Allocation algorithm is introduced in this section as the bandwidth sharing scheme that adapts the configuration of an AAPN to meet the demands of changing traffic patterns. The scheme is a variant of the FBP method with a few adjustments as it will be described below.

In AWTA, the traffic matrix is constructed from the queues information at each edge node. Once every inter-configuration period, τ, all edge nodes send the traffic

information to the global controller located in the core node.. The minimum τ is two times the maximum core-edge link propagation delay in the network plus the time it takes to calculate the service matrix in AWTA. The global controller computes the service matrix as described in the FBP architecture section after it receives all traffic matrix information from all edge nodes. After the computation of the new service matrix, the core node transmits this new service matrix back to each edge node. This brings up the propagation delay and synchronization issues for the network.

The AWTA algorithm may assign either a full wavelength (analogous to a full path of the FBP switch) or only a part of it, with the wavelength bandwidth being divided in timeslots. In the first case, there are a total m wavelengths to be assigned. A varying number of wavelengths between source-destination pairs are assigned to provide different bandwidth. In the second case, the bandwidth of one wavelength is divided into m timeslots per frame and these are the ones assigned in different numbers to provide variable bandwidth. In the first case, the core node has to change configuration once every τ seconds; while in the second case, the core node has to change configuration m times every frame and change the arrangement of slots every τ seconds. A combination of both methods may be used if there is more than one wavelength available. The frames would contain larger timeslots and the switch therefore has to change configuration less often.

Figure 33 illustrates the timeline of scheduling and control operations for a complete inter-configuration period. It is essential that all the edge nodes apply the new service matrix rates at the appropriate time and that they synchronize their transmission with the core node in such a way that the new arrangement of timeslots in the TDM frame corresponds to the new configuration of the core node at the precise moment when the optical signal reaches the core node. Deficient synchronization leads to packet mis-routing and/or loss. The proper transmission times depend on the propagation delay between the core node and the respective edge node. It is also essential for the core node to receive all traffic matrices from the edge nodes just before the computation starts.

Inaccurate traffic matrix information leads to imprecise service matrix calculation, which results in larger queuing delays.

The "service time", *s_time* defined in timeslot is the waiting time for edge nodes to apply the received service matrix. The purpose of it is to make sure that all traffic based on the new service matrix arrives at the core node at the same time. The "traffic time", *t_time*, defined in timeslots is the waiting time for the edge node to start transmitting the traffic matrix information to the core node. The purpose of it is to make sure that the core node receives the correct traffic matrix for service matrix computation. Figure 33 illustrates the timing diagram for a network with equal link propagation delays between the core node and all edge nodes (*prop q*). The "service times" and "traffic times" for all edge nodes have the same values.

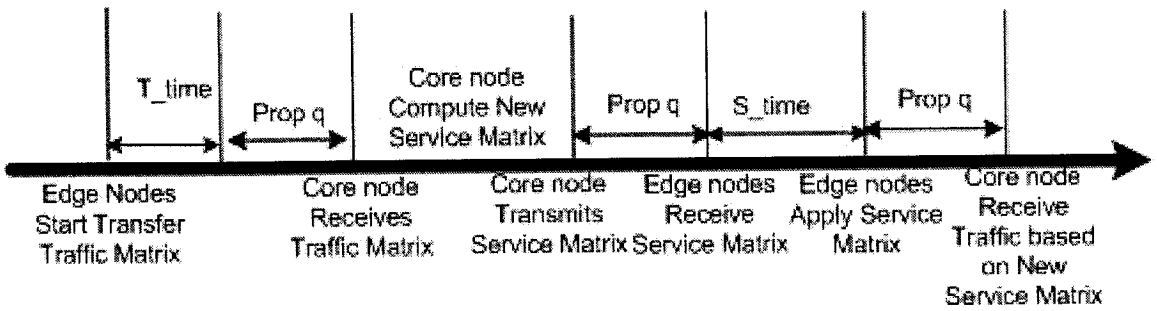

Figure 33 Timeline of the Adaptive Timeslot Allocation method for all edge nodes located at the same distance from the core node

Figure 34 illustrates the timing diagram for a network with two different link propagation delays. This is an example with one core node and two edge nodes, I and J. Edge node I has the link propagation delay, *prop I,* and edge node J has the link propagation delay, *prop J.* Since the value of *prop J* is larger than *prop I,* the traffic time for edge node I, *t_time_I*, has a higher value than *t_time_J* in order for the traffic matrix from both edge nodes to arrive at the core the at the same time. It is essential to provide correct traffic information for the computation of the service matrix at the core node. In the same way, the *s_time_I* is also larger than *s_time_J* in order for the traffic based on the new service matrix to arrive at the core node at the same time. It is critical for the core node to

receive the new arrangement of packets at the right time to avoid miss-routing and collisions. The core node uses M crossbars to transmit traffic and the crossbars are configured as dictated by the decomposition of the service matrix. If the edge nodes release packets in an arrangement different to the crossbar configuration in core node, the packets will be mis-routed.

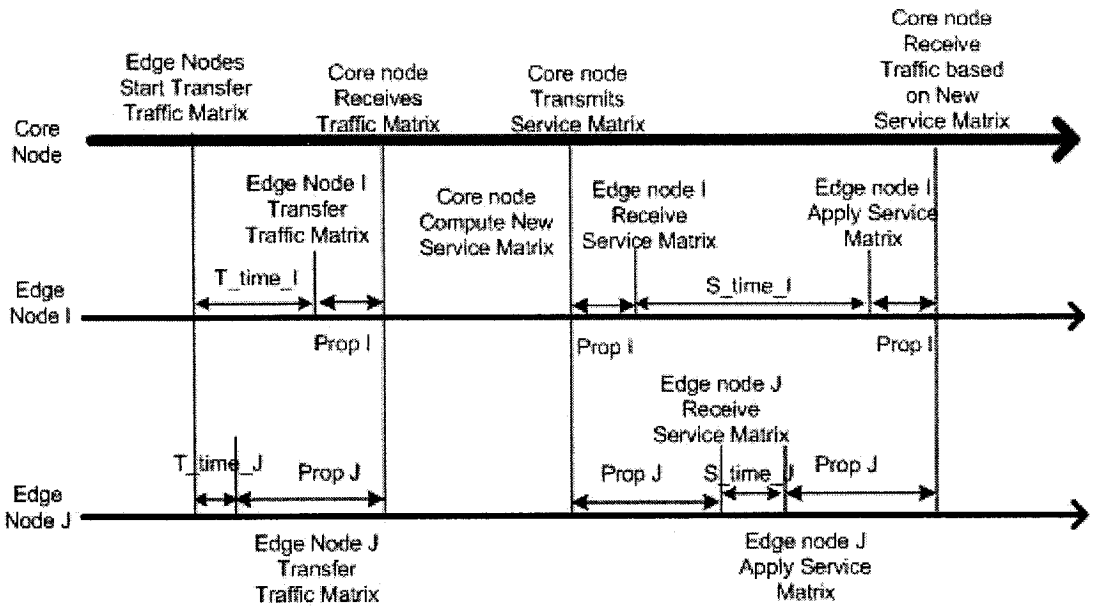

Figure 34 Timeline of the Adaptive Timeslot Allocation method for edge nodes at different distances from the core node

4.4 OPNET Implementation

After the initial AAPN proposal, there have been a few methods suggested for resource allocation, including Optical Burst Switching for example. It is indispensable to setup up some benchmark parameters for comparing different methods. The fundamental topology for AAPN is a single star network with a single wavelength channel; which means there is only one very fast crossbar switch at the core node. The selected test network contains one core node and 8 edge nodes. The connections between core node and edges nodes carry one wavelength with data link rate 10 Gbps. The timeslot is set to

10 ms for all scenarios. After finalizing the AAPN test parameters, the next step is the implementation of it in OPNET.

Figure 35 illustrate an AAPN single star network topology as described in the above paragraph. There are 8 edge nodes and one core node in this star network, with only one wavelength per link. For the sake of making the simulations faster, all packets are destroyed once they are received at the core node, resulting in links that do not need to be bidirectional. This simplification is possible since we are concerned only with queuing delays in the source modules of the edge nodes and not with the queuing delays at the destinations.

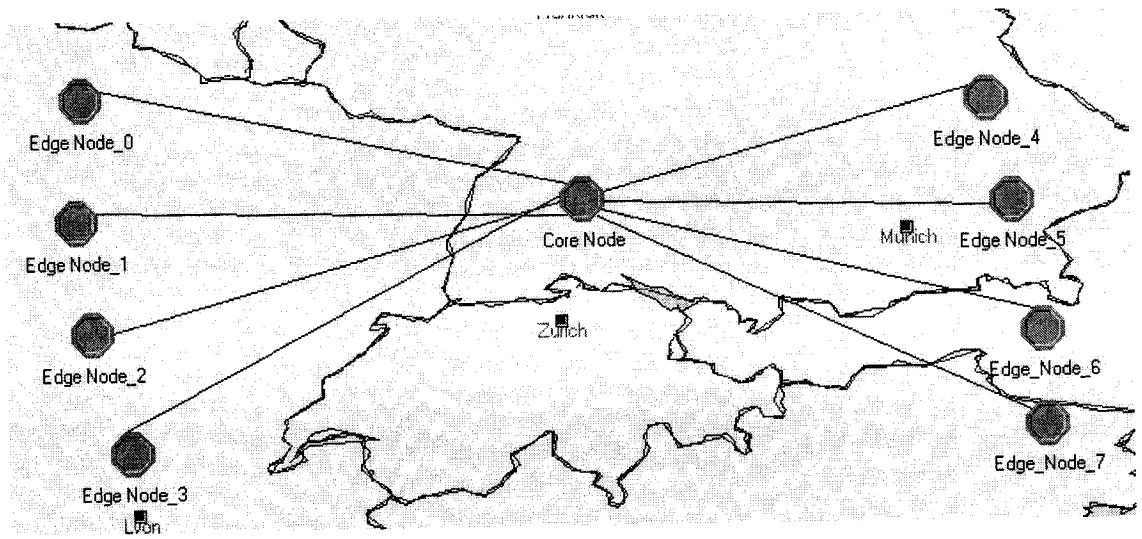

Figure 35 AAPN topology used for the simulations in OPNET

Figure 36 shows the architecture of AAPN core node. It contains 8 inputs/output ports and one process node, the core switch. The main functions for this node process are to forward packets to the appropriate edge node based on the header information and to perform the service matrix computation. Each source edge node is connected to an input port and each destination edge node is connected to an output port of the core node. The process model, *core switch*, logically represents all l multiplexers, l demultiplexers and m l x l crossbars for the core switch. The blue lines are packet streams and the yellow lines represent the logical pair for input and output ports. The global controller is also located

at the *core switch* process model and it is in charge of receiving the traffic matrix from the edge nodes, computing the service matrix and forwarding the new service matrix information to the edge nodes.

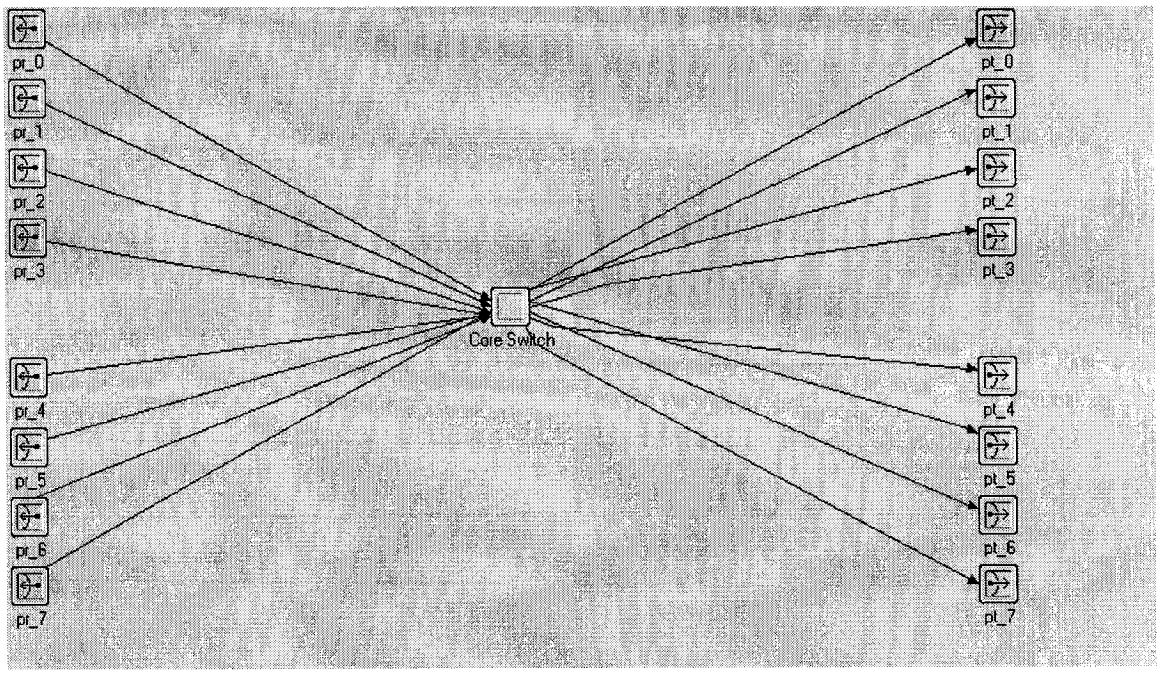

Figure 36 Elements of the AAPN Core Node

Figure 37 illustrates the state machine that constitutes the process model of the core node. This process performs the service matrix calculation, switch reconfiguration and ICI packet operations every τ seconds. At all other times, the hub is a simple "receive and destroy" based device that just examines the header of the packets received to determine whether it is a control or a data packet. In the case of a control packet, the information will be used to compute the service matrix. In the case of a data packet, the packet is destroyed. The "compute" state acts as the global controller for the network and it computes the new service matrix. The "update" state collects the traffic matrix information from the edge nodes and delivers the new service matrix to the edge nodes. Both the traffic matrix and service matrix transactions use control packets between edge nodes and core node.

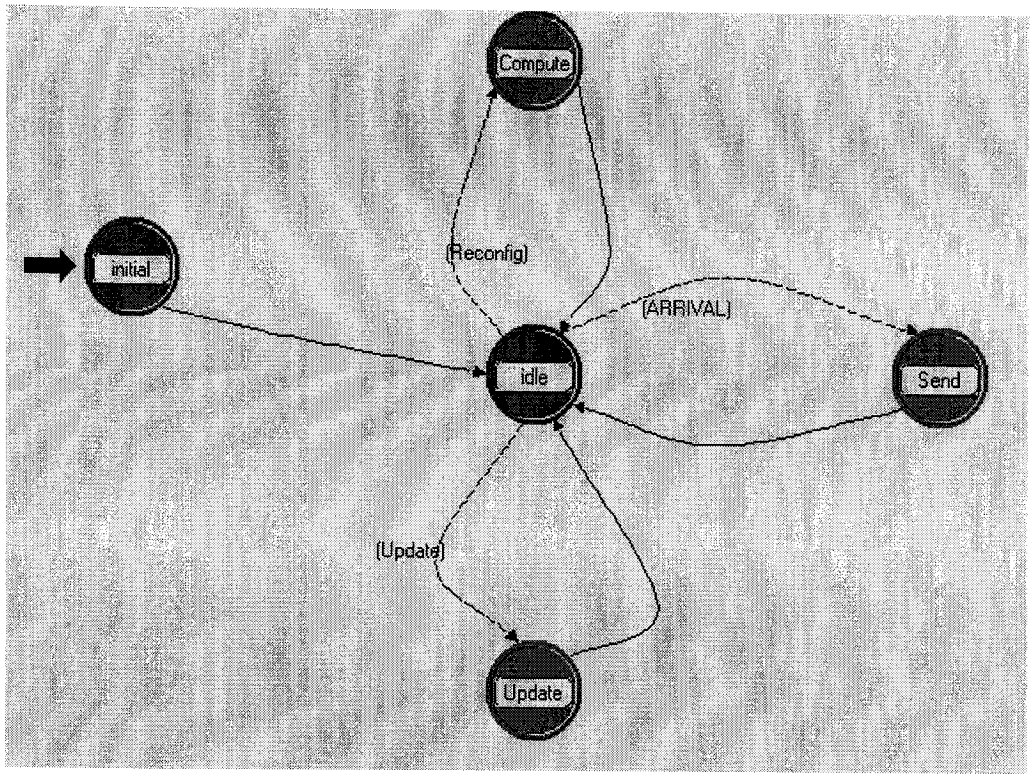

Figure 37 Process model in AAPN core node

Figure 38 illustrates the modules of the AAPN edge node, which performs the actions not only of an AAPN edge node, but also generates the traffic. The "src" process model generates packets based on a desired interarrival distribution. The "proc" process model assigns destination address for each packet based on the desired destination distribution.

The edge node receives control packets from the core node, which correspond to the information about future configuration (service matrix) and destroys them after updating the scheduling operations. The edge node, as said above, forwards control packets which contain the traffic matrix to the core node. The *shared memory* module contains the different queues for each destination edge node. It queues packets in buffers, updates the traffic matrix, sends the information to the global controller and releases packets according to the current service matrix.

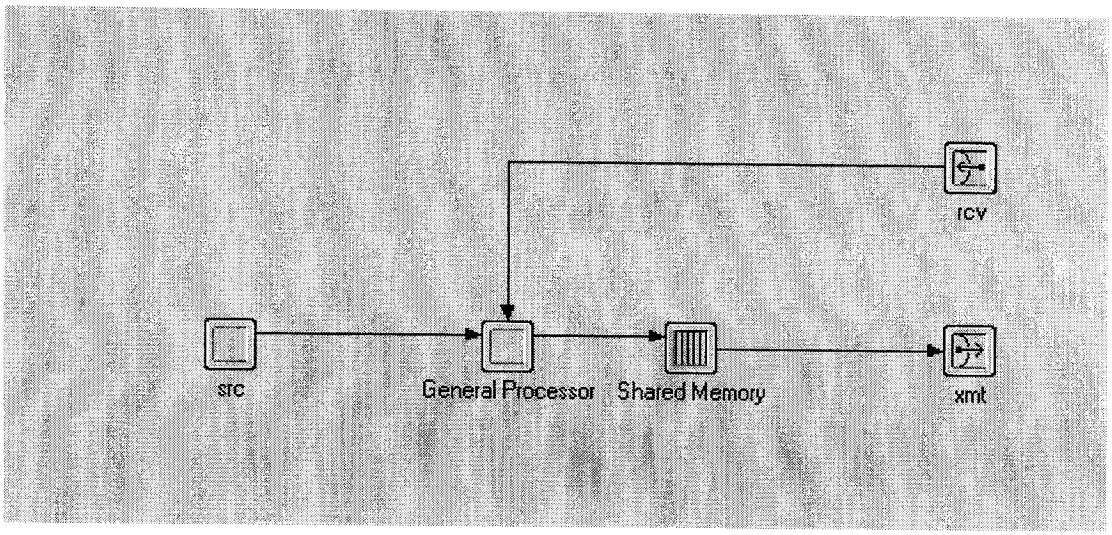

Figure 38 Modules of the AAPN Edge Node

Figure 39 shows the state machine of the *general processor* process model in the AAPN Edge Node. The main functions of this model include assigning a destination address to newly generated packets. The SRC_ARRVL and RCV_ARRVL expressions correspond to the conditions used to verify whether the current packet was newly generated at the traffic generator or received from the core node. The "T_update" state computes a new bistochastic matrix for non-uniform destinations traffic every T timeslots.

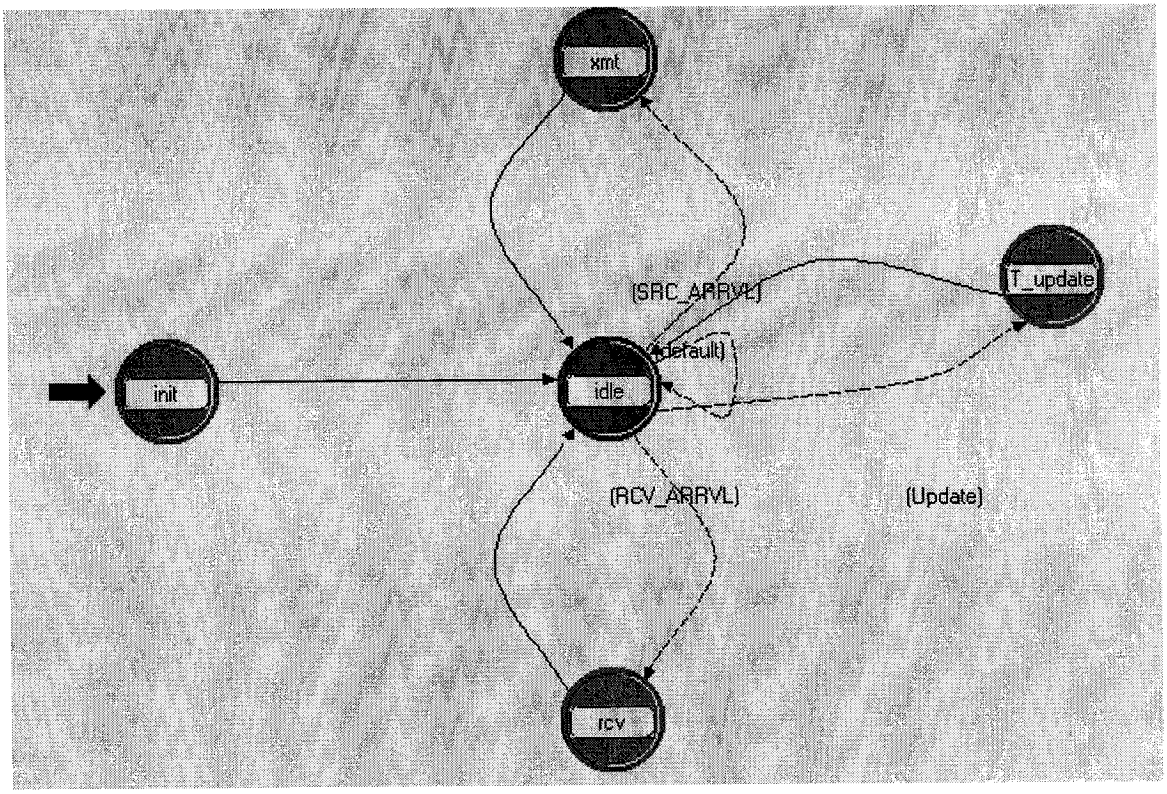

Figure 39 Process Model of the *general processor* in the Edge Node

Figure 40 shows the state machine of the *shared memory* process model. At the "arrival" state, it obtains the destination address from the incoming packet header and buffers the packet in the appropriate FIFO queue, one for each destination edge node. The "traffic" state transmits the traffic matrix information to the core node every inter-configuration period τ. The "service" state receives the service matrix from the core node and updates the service rate of the different queues according to the received service matrix. Every l timeslots, queues will release l packets according the service matrix. The service matrix determines the numbers of packets released from each queue in one frame.

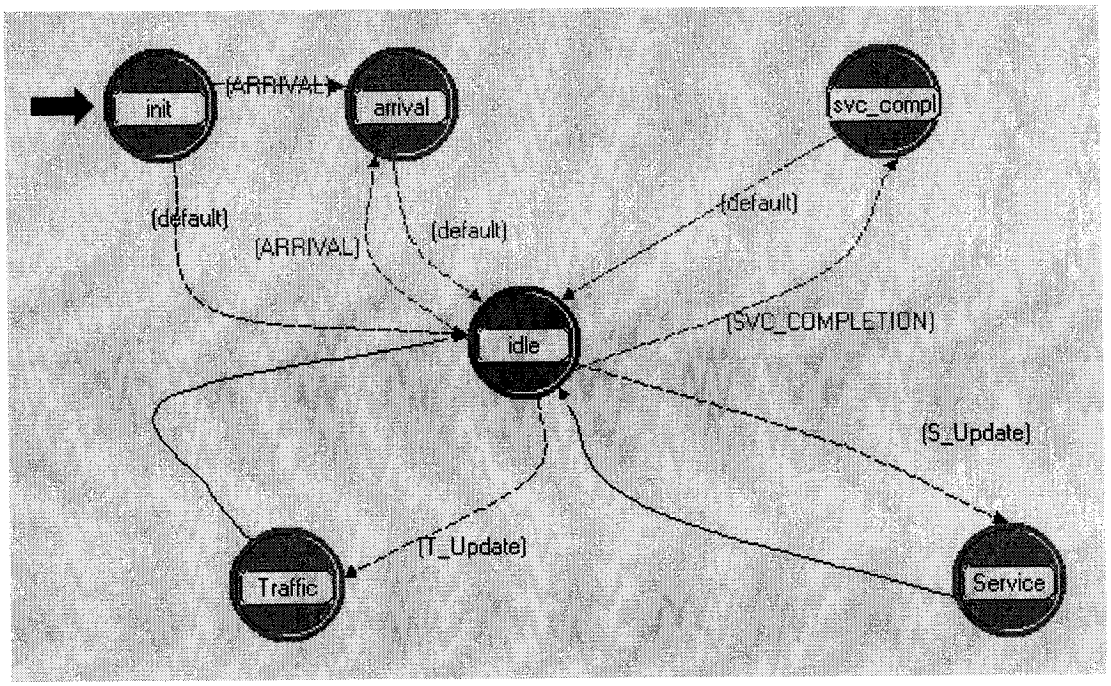

Figure 40 Process Model of the *shared memory* module in the Edge Node

One of the benchmark tests for AAPN is a simulation network that includes some features of IP traffic. Figure 41 shows an OPNET network model built for this purpose. The network model appears slightly different than others; however, the actual AAPN topology remains unchanged. The 4 different packet sizes used in the AAPN IP traffic simulation network are 40 bytes, 552 bytes, 576 bytes and 1500 bytes with 46 %, 18%, 18% and 18% of traffic load, respectively. As shown in Figure 41, the node models "40", "552", "576" and "1500" represent packet generators with packet size 40 bytes, 552 bytes, 576 bytes and 1500 bytes, respectively. All packets are generated at these node models and then forwarded to the each edge nodes. The edge nodes consequently do not contain traffic generators anymore but use 4 point to point receivers that are connected to the *general processor* process model, who will aggregate all the packet sources as shown in Figure 42. The remaining processes works in the same way as described above.

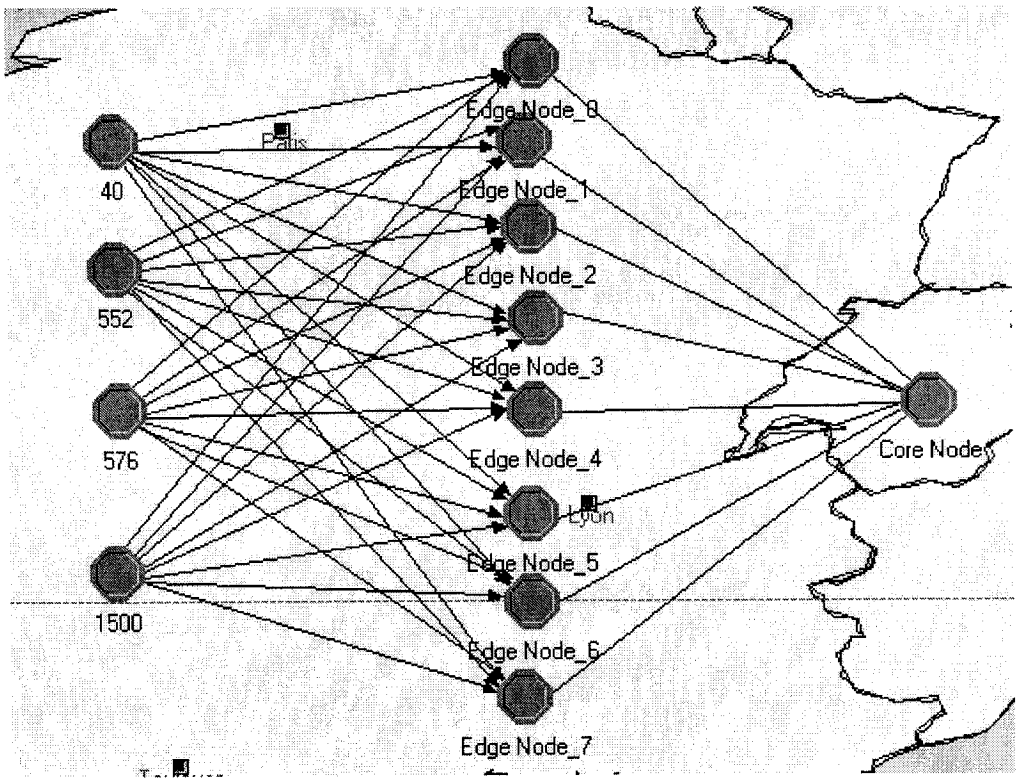

Figure 41 AAPN Network implemented to include the features of IP traffic

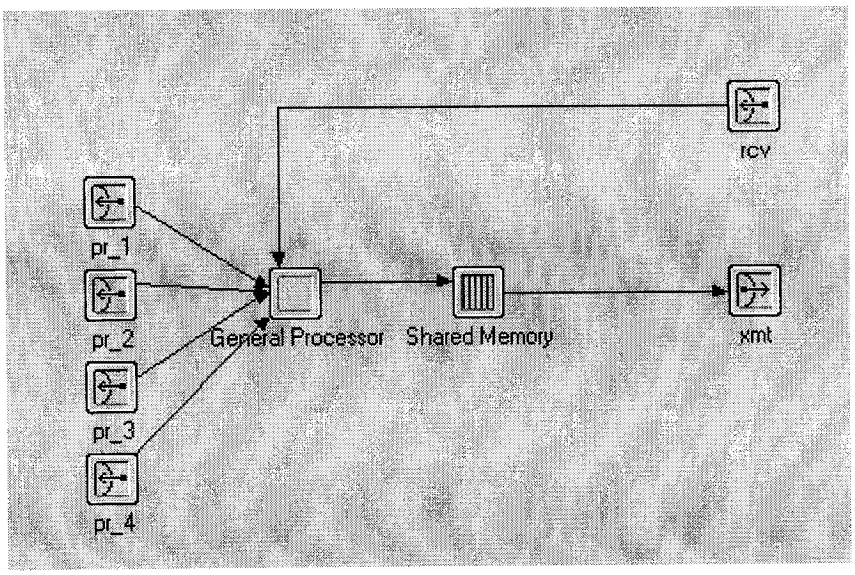

Figure 42 Modules of the edge node variation used for the simulation with IP traffic

Figure 43 shows the modules of the IP traffic generator node model. All IP node models "40", "552", "576" and "1500" use the same model but with their respective packet size

as a parameter. The "src" module generates 8 times more packets than the desired load based on a desired interarrival distribution and the *distributor* process model evenly distributes the generated traffic to the eight edge nodes. Since the IP traffic generator generates traffic 8 times faster then desired load, each edge node will receive only 1/8 of the packets generated after being evenly distributed. The correct traffic load is therefore being produced.

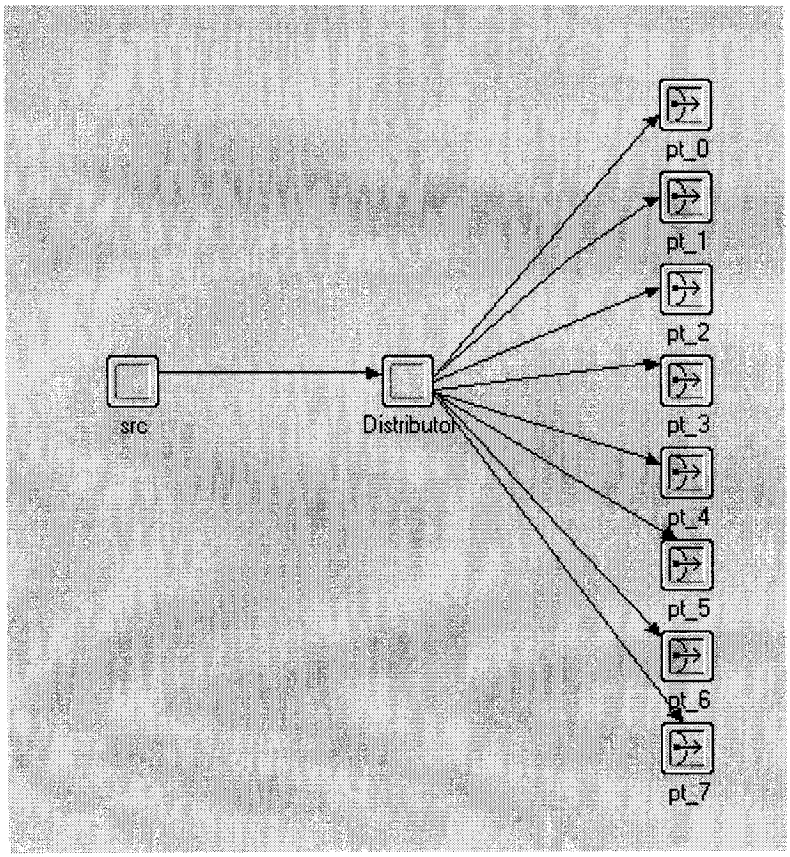

Figure 43 Modules of the IP Traffic Generator node

4.5 Performance Analysis

A benchmark test network was implemented for AAPN simulation. The long term purpose of it is to compare the Adaptive Wavelength-Timeslot Allocation method to other schemes such as Optical Burst Switching (OBS) and to validate the network

performance. The comparison is beyond the scope of this thesis and therefore will not be discussed. All simulation results presented here are related only to AAPN with AWTA simulated under various network environments. The service matrix computation time is assumed to be zero for all scenarios.

The test network contains one core node and eight edge nodes. The connections between core node and edge nodes are unidirectional links (since packets are being destroyed at the core) that carry one wavelength and the data link rate is 10 Gbps. The frame size is 1 ms and each timeslot is 10 μs for all networks. Five different network scenarios have been proposed:

- zero propagation delay between core and edge nodes.
- a metro network with all edge nodes located at the same distance from the core node and with propagation delay equivalent to 10 timeslots link delay between the core and (every) edge node.
- a metro network with different distances between edge and core nodes, these distances being equivalent to 0, 1, 2, 3, 7, 8, 9 and 10 timeslots delay between the core and each of the eight edge nodes, respectively.
- a long haul network with fixed propagation delay set to 1000 timeslots delay, which corresponds to 2000 km.
- a long haul network with different distances between the core and edge nodes; the link propagation delays equivalent to 0, 100, 200, 300, 700, 800, 900 and 1000 timeslots.

There are also four different traffic patterns proposed to test these scenarios. The first traffic pattern is uniform Poisson traffic which uses Poisson traffic arrivals with uniform destinations distribution. The uniform Poisson traffic uses the same model as described in the FBP section. The second traffic pattern is non-uniform Poisson traffic, which uses Poisson traffic arrivals with non-uniform destination distribution.

In the non-uniform destinations experiment, the destination distribution uses a number of non-uniform distributions created in OPNET's *Probability Density Function (PDF)*

Editor. The data contained in the distributions were based on bi-stochastic matrices. The following are five bi-stochastic matrices $\boldsymbol{B}$, $B = [b_{ij}]$, that were used to modulate the destinations of the packets. For each matrix, the element $\boldsymbol{b_{i,j}}$ is the probability that a packet generated at edge node $\boldsymbol{i}$ will be destined for edge node $\boldsymbol{j}$; therefore, the traffic load between certain edge node source-destination pairs (determined by the largest numbers in B) will be higher than other pairs.

B1 =

0.0358	0.0000	0.0533	0.1423	0.0003	0.2371	0.0002	0.5311
0.0000	0.2286	0.0018	0.3291	0.0142	0.2022	0.2006	0.0235
0.2345	0.2152	0.1373	0.0369	0.0303	0.0080	0.3377	0.0000
0.3441	0.2063	0.0002	0.2072	0.0695	0.1312	0.0415	0.0000
0.0008	0.0074	0.0408	0.0050	0.4669	0.0004	0.2171	0.2617
0.0065	0.0005	0.2310	0.0317	0.2170	0.3379	0.0405	0.1349
0.2410	0.0091	0.5347	0.0007	0.0008	0.0446	0.1624	0.0067
0.1373	0.3328	0.0009	0.2471	0.2010	0.0387	0.0000	0.0421

B2 =

0.0328	0.0038	0.0629	0.0946	0.1228	0.4345	0.0069	0.2415
0.0624	0.0919	0.2288	0.0848	0.0064	0.0228	0.0121	0.4908
0.3187	0.1254	0.0345	0.0349	0.1126	0.0061	0.3678	0.0000
0.0002	0.1248	0.1080	0.0001	0.5786	0.0000	0.0753	0.1131
0.0000	0.0190	0.3684	0.0323	0.1444	0.2158	0.1999	0.0201
0.4478	0.0007	0.1191	0.2038	0.0228	0.0545	0.0536	0.0977
0.0947	0.2586	0.0723	0.3799	0.0002	0.0063	0.1868	0.0011
0.0434	0.3757	0.0059	0.1695	0.0121	0.2601	0.0976	0.0356

B3 =

0.2917	0.0000	0.2677	0.0215	0.0007	0.1355	0.2828	0.0001
0.2692	0.0252	0.0009	0.2844	0.2474	0.0000	0.1513	0.0216
0.2824	0.0046	0.1413	0.2659	0.1521	0.0293	0.1233	0.0010

0.0000	0.0023	0.2662	0.1504	0.0458	0.0009	0.0044	0.5298
0.1555	0.0000	0.1673	0.1254	0.0000	0.2837	0.2680	0.0000
0.0003	0.5539	0.1510	0.0017	0.0016	0.0208	0.0045	0.2662
0.0000	0.1293	0.0000	0.0008	0.2865	0.5297	0.0261	0.0276
0.0009	0.2847	0.0056	0.1498	0.2659	0.0001	0.1395	0.1536

B4 =

0.0046	0.3542	0.4728	0.0008	0.0000	0.0001	0.0000	0.1674
0.3379	0.0073	0.0001	0.2757	0.0494	0.0000	0.0000	0.3296
0.0215	0.1674	0.0046	0.3296	0.0000	0.2556	0.0071	0.2142
0.0000	0.1741	0.0246	0.0069	0.1890	0.3511	0.2542	0.0001
0.0000	0.2749	0.4971	0.1897	0.0074	0.0261	0.0005	0.0042
0.0037	0.0000	0.0000	0.0208	0.2549	0.3606	0.3540	0.0061
0.0486	0.0000	0.0008	0.1755	0.3296	0.0001	0.1671	0.2784
0.5837	0.0221	0.0000	0.0010	0.1697	0.0064	0.2171	0.0000

B5 =

0.4816	0.0072	0.0031	0.2134	0.0936	0.0000	0.0008	0.2004
0.0374	0.0002	0.1368	0.0349	0.2181	0.0415	0.4918	0.0393
0.0012	0.1370	0.1031	0.0383	0.0020	0.4980	0.2132	0.0072
0.0235	0.0433	0.2176	0.1361	0.0047	0.0394	0.0397	0.4956
0.0470	0.2504	0.0027	0.5283	0.1568	0.0076	0.0059	0.0015
0.2528	0.4988	0.0383	0.0000	0.0431	0.1398	0.0269	0.0002
0.1373	0.0205	0.4984	0.0078	0.0004	0.0404	0.0795	0.2158
0.0192	0.0426	0.0000	0.0412	0.4813	0.2333	0.1424	0.0400

The parameter T, which is the duration of each destinations distribution, was adopted from the previous part of the project for non-uniform traffic distribution. The C++ code would generate a new bistochastic matrix for destinations distribution every T timeslots. However, this method could not be implemented in OPNET because the PDF distribution

needs to be predefined. Therefore, the destinations distribution will randomly switch between these 5 matrices once every T timeslots.

Since there are only a few matrices, the average over long periods of time (t>>T) is not uniform as intended; therefore, a compensating matrix $B_4^{'}$ was created to make the average uniform:

$$B_4^{'} = \left(Norm\left[\overline{\overline{MaxB_4}} - B4\right]\right)$$

where $\overline{\overline{MaxB_4}}$ is a $l \times l$ matrix with all elements equal to the max element in B_4, "*Norm*' is the operation of normalizing the matrix.

$B_4^{'} =$

0.1580	0.0620	0.0294	0.1591	0.1593	0.1593	0.1593	0.1133
0.0665	0.1573	0.1593	0.0836	0.1457	0.1593	0.1593	0.0688
0.1534	0.1133	0.1580	0.0688	0.1593	0.0891	0.1573	0.1005
0.1593	0.1115	0.1526	0.1575	0.1074	0.0628	0.0895	0.1593
0.1593	0.0839	0.0228	0.1072	0.1573	0.1522	0.1592	0.1582
0.1583	0.1593	0.1593	0.1536	0.0893	0.0602	0.0621	0.1577
0.1460	0.1593	0.1591	0.1111	0.0688	0.1593	0.1134	0.0829
0	0.1532	0.1593	0.1590	0.1127	0.1579	0.0997	0.1593

The new matrix $B_4^{'}$ is therefore also a bistochastic matrix since the row and column sums all are equal to one.

The third traffic pattern is uniform bursty traffic which uses Pareto traffic arrivals with uniform destinations distribution. The last traffic pattern is the IP traffic pattern that uses the four typical packet sizes of IP traffic (with uniform destinations distribution) as described previously.

4.5.1 AAPN-AWTA with Uniform Poisson Traffic

Figure 44 shows the average queuing delay obtained at edge node 1 of the AAPN with AWTA with the no-link-propagation-delay scenario. The speed up is two and different values of inter-configuration period have been used. If there are no link propagation delays between the edge nodes and the core node in the simulated network, the scenario is equivalent to the FBP switch alone, in which there is no propagation delay between the sectors and the optical core crossbar switches. The FBP performance analysis considers the queuing delay as sum of input sectors and output sectors. The AAPN performance analysis considers only the input queuing delays.

The traffic model for Figure 44 is Bernoulli arrivals with uniform destinations distribution. The traffic estimation, $\overline{\Lambda}_e$, is calculated only with the lengths of the queues in the source edge nodes. Both *t_time* and *s_time* are equal to 1 timeslot even though there are no link propagation delays. Sometimes, the traffic matrix does not get transmitted until the next timeslot. This is a safe margin to make sure the service matrix is computed based on the correct traffic information. From the simulation it could be observed that all edge nodes have identical queuing delay. The increase in delay is proportional to the inter-configuration period τ. Both the inter-configuration period and the load are the two major factors for the increase in delay. The three different values of τ resulted in noticeable different queuing delays only for traffic loads above 40%. The queuing delay is around 460 μs for τ equal to 160 timeslots at 90% traffic load.

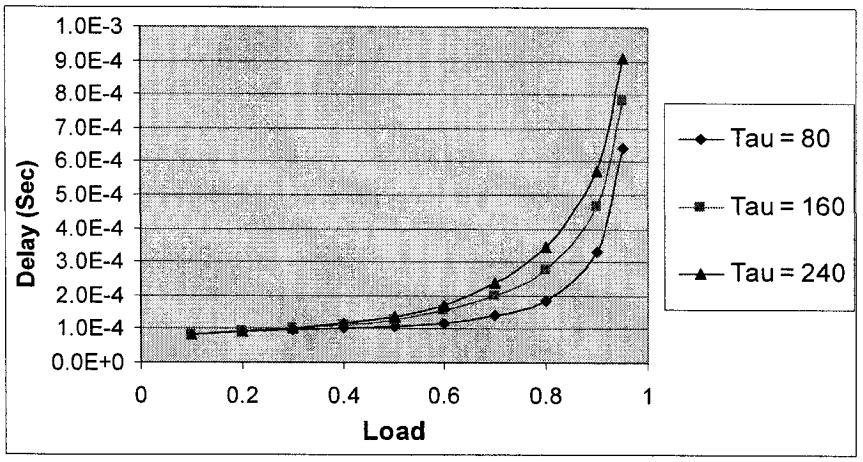

Figure 44 Average delay in edge node 1 versus load obtained with the no propagation delays scenario. The speed up is two and the traffic is Poisson with uniform destinations. Values for τ are 80, 160 and 240

Figure 45 shows the average delay versus load obtained at edge node 1 of AAPN with AWTA for three different scenarios: no link propagation delay, fixed link propagation delay and various link propagation delays. The speed up is two, τ is 80 timeslots and the traffic is Poisson with uniform destinations. The traffic estimation, $\overline{\Lambda}_e$, is calculated only with the queue length in the each queues. Both t_time and s_time are equal to 1 and 15 timeslots for network with no propagation delay and fixed propagation delay respectively. The metro network scenario has link propagation delays equivalent to 0, 1, 2, 3, 7, 8, 9 and 10 timeslots, respectively, between core and edge nodes. Therefore, *t_time* corresponds to 16, 15, 14, 13, 9, 8, 7, 6 timeslots; while *s_time* is equal to 32, 31, 30, 29, 25, 24, 23, 22 timeslots, respectively.

All three queuing delays have very similar performances for the various distances because the link propagation delays have only relatively small differences compared to the inter-configuration period. The largest value for link propagation delay is 10 timeslots and the smallest value is zero. The 10 timeslots different has only a slight effect in the queuing delay performance. The queuing delay for metro fixed link propagation delay network is slightly longer than queuing delay for metro various link propagation delays network (which is also slightly higher than queuing delay for the no

propagation delays scenario as expected). The service matrix is compute based on more recent traffic matrices for the no propagation delay scenario compared to the others, which results in a small improvement in queuing delay performance. The queuing delay remains around 330 μs for τ equals to 80 timeslots at 90% traffic load.

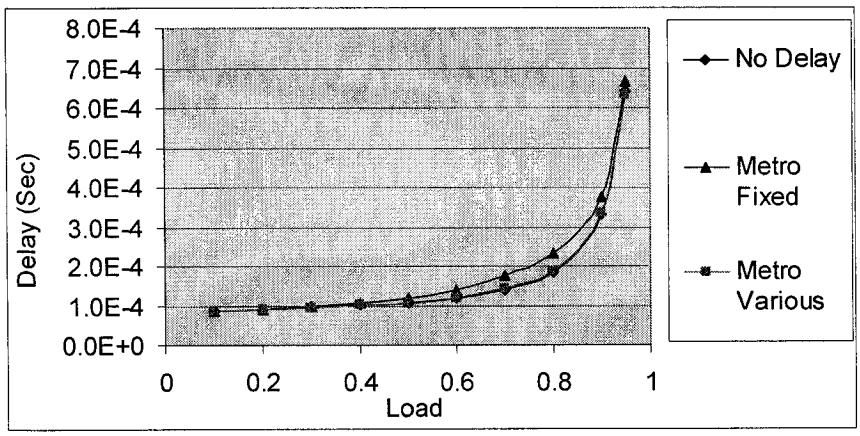

Figure 45 Average delay in edge node 1 versus load with τ = 80 obtained for three different scenarios: network with no delay, network with fixed delays and network with various delays. The speed up is two and traffic is Poisson with uniform destinations

Figure 46 shows the average delay versus load obtained at various edge nodes of the AAPN with AWTA for a network with various link propagation delays. The speed up is two, τ is 80 timeslots and the traffic is Poisson with uniform destinations. This figure shows again the queuing delay effect due to the link propagation delay. A larger link propagation delay leads to larger queuing delay. The queuing delay is not very different for different edge nodes because the maximum difference in link propagation delay is only 10 timeslots (a small number compared to the inter-configuration period).

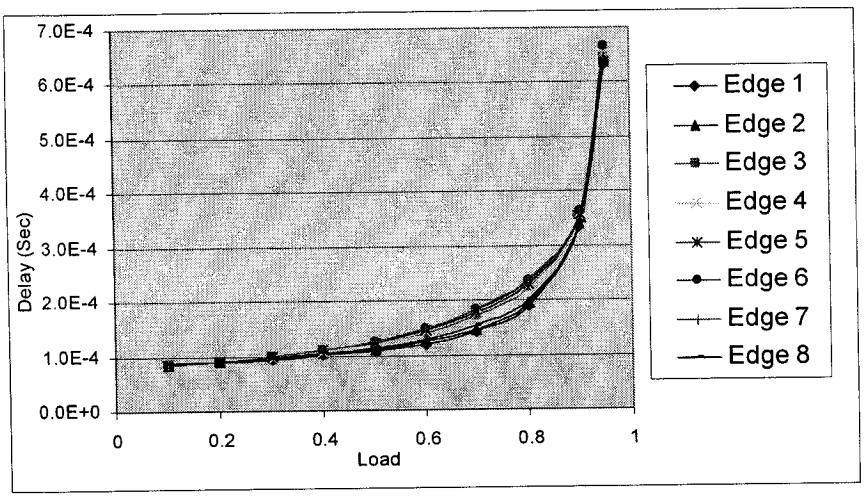

Figure 46 Average delay in various edge nodes versus load with $\tau = 80$ obtained for network with various delays. The speed up is two and traffic is Poisson with uniform destinations

Figure 47 shows the average delay versus load obtained at edge node 1 of AAPN with AWTA for no propagation delay (a collapsed network) and long propagation delay (a long haul network) with a speed up of two. The traffic is Poisson arrivals with uniform destination and $\tau = 2100$ timeslots. The traffic estimation, $\overline{\Lambda}_e$, is calculated only with the queue length in the each queues. The values of *t_time* and *s_time* are equal to 5 and 2000 timeslots for the no propagation delay and fixed long propagation delays, respectively. As expected, the long haul network suffers from higher queuing delays in the source edge nodes.

The queuing delay with no propagation delay is 1.5 ms at 90% traffic load with $\tau = 2100$ timeslots compared to only 460 μs with $\tau = 160$ timeslots in Figure 45, which indicates once more that the inter-configuration period, τ, is one of the main factors for edge node queuing delay. There is roughly 1 ms queuing delay difference between the collapsed network and the long-haul network at 90% traffic load. The collapsed network computes a new service matrix based on the information collected 5 timeslots before computation time instead of 2000 time slots in the case of the long-haul network. In the latter case, the received traffic matrices could be considered "outdated" when they arrive at the core node, which results in an "outdated" service matrix and consequently a much longer

queuing delay at edge nodes. Figure 47 clearly shows the impact of the long propagation delays between edge and core nodes. Both queuing delays are however very similar for traffic load below 50%.

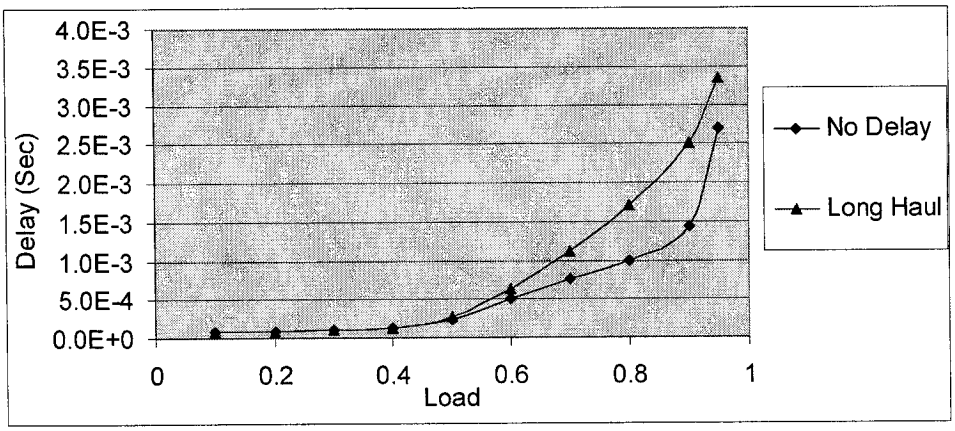

Figure 47 Average delay in edge node 1 versus load with $\tau = 2100$ obtained for two different scenarios: network with no delay and network with long delay. The speed up is two and traffic is Poisson with uniform destination

4.5.2 AAPN-AWTA with Non-uniform Poisson Traffic

Figure 48 shows the average delay obtained at edge node 1 of AAPN-AWTA for the no propagation delay scenario with speed up of two and Bernoulli traffic arrivals with non-uniform destinations distribution and different values of inter-configuration period τ. Both *t_time* and *s_time* are equal to 1 timeslot and the duration of a destinations probability distribution, T, is equal to 1000 timeslots. The queuing delays are different for the various edge nodes because of the non-uniform destinations distribution. The queuing delays shown in Figure 48 are the largest reported so far because the largest element in the bi-stochastic matrix B4 is $b_{1,8}$ which is a 0.5837 (58.37%). This indicates that more than half of the packets generated at edge node one (the one being measured) are assigned to edge node 8, which results in an ever increasing queuing delay in edge node 1, particularly for the queue of packets destined to edge node 8. This happens because a maximum of 9/16 of the traffic load (56.25%) can be carried between any one

source-destination pair given that the no starvation policy is used and at least one timeslot is assigned to each pair for every frame. The queuing delay is around 2.5 ms for τ equal to 160 timeslots at 90% traffic load. This value is much higher than the one obtained for Poisson arrivals with uniform destinations distribution.

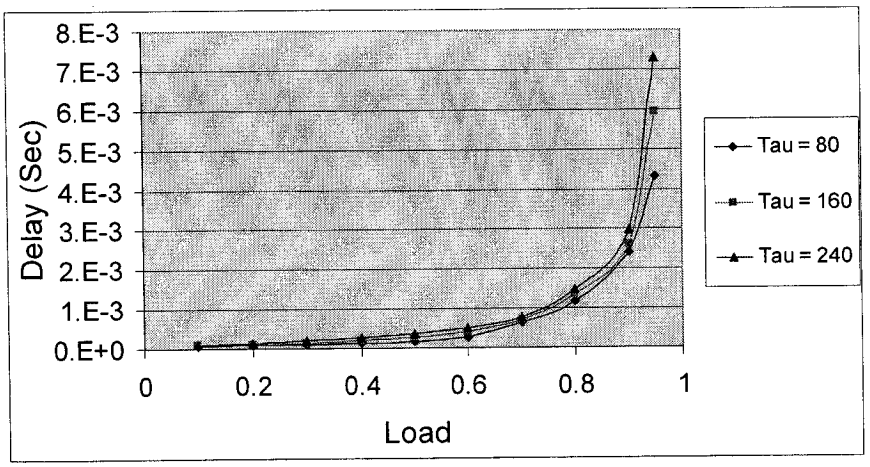

Figure 48 Average delay in edge node 1 versus load obtained with the no propagation delays scenario. The speed up is two and the traffic is Poisson with non-uniform destinations. Values for τ are 80, 160 and 240

Figure 49 shows the average delay versus load obtained at edge node 1 of AAPN-AWTA for three different scenarios: no link propagation delay, fixed link propagation delay and various link propagation delays. The speed up is two, τ is 80 timeslots and the traffic is Poisson with non-uniform destinations. The traffic estimation, $\overline{\Lambda}_e$, is calculated with equation 1. The values of *t_time* and *s_time* are equal to 1 and 15 timeslots for network with no propagation delay and fixed propagation delay, respectively. The metro network scenario has link propagation delays equivalent to 0, 1, 2, 3, 7, 8, 9 and 10 timeslots, respectively, between core and edge nodes. Therefore, *t_time* corresponds to 16, 15, 14, 13, 9, 8, 7, 6 timeslots; while *s_time* is equal to 32, 31, 30, 29, 25, 24, 23, 22 timeslots respectively.

All three queuing delays have very similar performances under the three different scenarios because the link propagation delays correspond only to small variations

compared to the value of τ. The delay for "metro fixed" is slightly higher than the other two because the average delay of all links is higher. The queuing delay remains around 2.5 ms for τ equals to 80 timeslots at 90% traffic load.

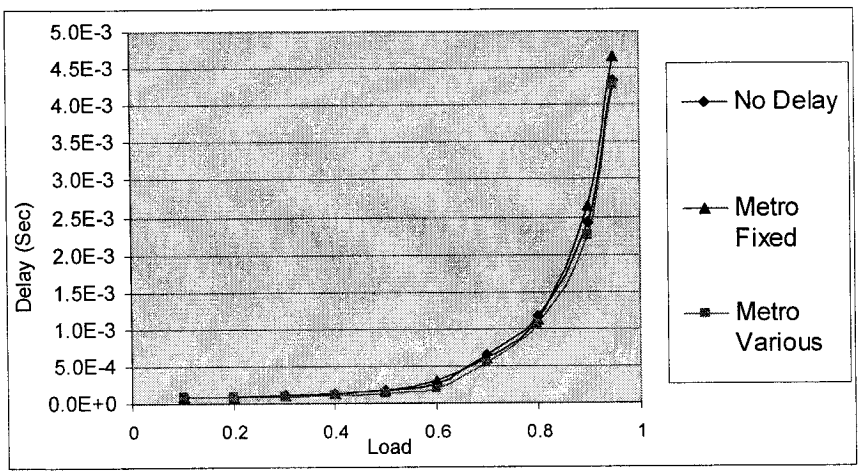

Figure 49 Average delay in edge node 1 versus load with τ = 80 obtained for three different scenarios: network with no delay, network with fixed delays and network with various delays. The speed up is two and traffic is Poisson with non-uniform destinations

Figure 50 shows the average delay versus load obtained at edge node 1 of AAPN with AWTA with no link propagation delay and long link propagation delay networks with a speed up of two for Poisson traffic arrival with non-uniform destination with τ = 2100 timeslots. The traffic estimation, $\overline{\Lambda}_e$, is calculated with the equation 1 listed in chapter 3. The duration of a destination probability distribution, T, is equal to 1000 timeslots. Both *t_time* and *s_time* are equal to 5 and 2000 timeslots for network with no propagation delay and fixed long propagation delays respectively. The increase in delay is related to the inter-configuration period τ. Both reconfiguration period and the different levels of load are the two major factors for the increase of delay.

The queuing delay with no link propagation delay network is 3 ms at 90% traffic load with τ = 2100 timeslots compared to only 2.5 ms with τ = 160 timeslots in Figure 49. It validates inter-configuration period, τ, is one of the main factors for edge nodes queuing delay. There is roughly 300 μs queuing delay difference between for network with no

link propagation and long link propagation delay at 90% traffic load. The network with no link propagation delay computes newer service matrix based on the information 5 timeslots before computation time instead of 2000 timeslots for network with long link propagation delays. The received traffic matrixes are outdated at the core node for network with long link propagation delays, which leads to a much longer queuing delay at the edge nodes. Figure 50 clearly shows the impact for the link propagation to edge node queuing delay.

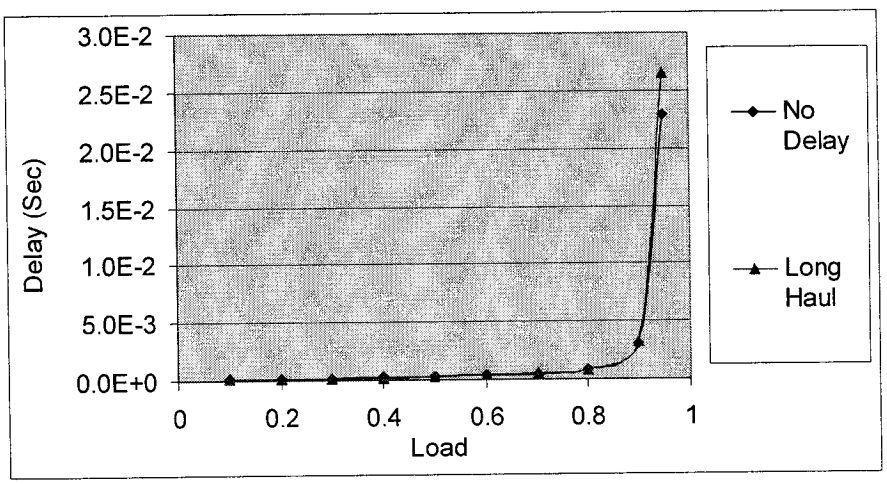

Figure 50 Average delay in edge node 1 versus load with τ = 2100 obtained for two different scenarios: network with no delay and network with long delay. The speed up is two and traffic is Poisson with non-uniform destinations

4.5.3 AAPN-AWTA with Uniform Bursty Traffic

Figure 51 shows the average delay obtained at edge node 1 of AAPN-AWTA for the no propagation delay scenario with speed up of two for Bursty traffic arrival uniform destination distribution and different values of inter-configuration period τ. The values of *t_time* and *s_time* are equal to 1 timeslot. All edge nodes have identical queuing delay because the traffic is balanced (uniform destinations) but increasing the value of τ does have a larger impact on delay because of the traffic fluctuations associated with bursty traffic.

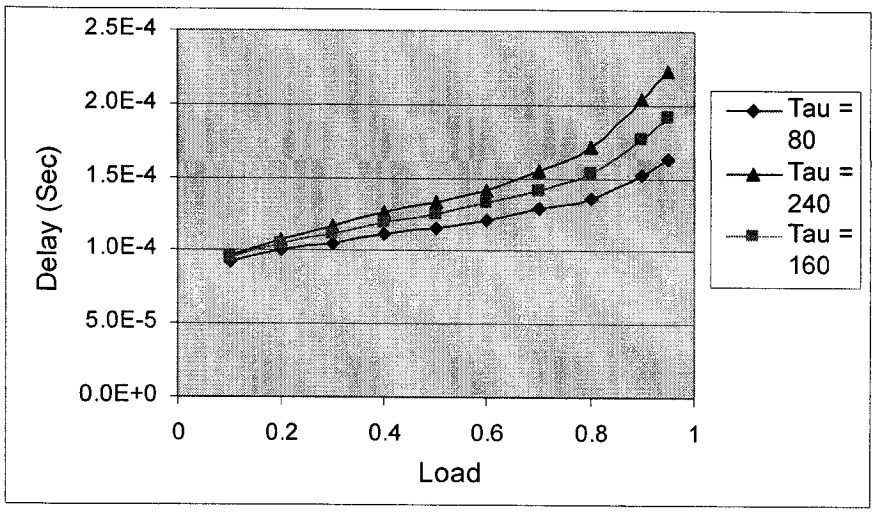

Figure 51 Average delay in edge node 1 versus load obtained with the no propagation delays scenario. The speed up is two and the traffic is Bursty with uniform destinations. Values for τ are 80, 160 and 240

Figure 52 shows a similar result to figure 45 but for bursty traffic. If the distances are small compared to the inter-configuration period, the variation in queuing delay is minimal.

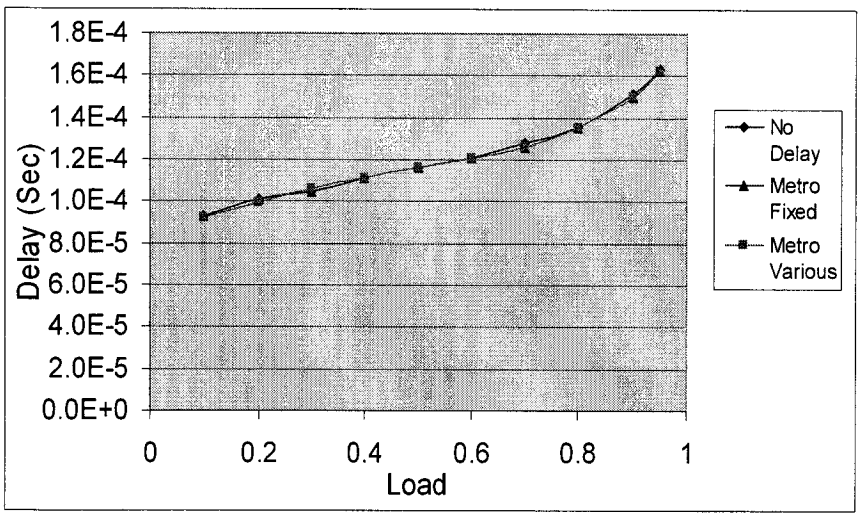

Figure 52 Average delay in edge node 1 versus load with τ = 80 obtained for three different scenarios: network with no delay, network with fixed delays and network with various delays. The speed up is two and traffic is Bursty with uniform destinations

Figure 53 shows a similar result to figure 47 but for bursty traffic. For a fixed value of τ, the longer distances have a big impact in the resulting queuing delay.

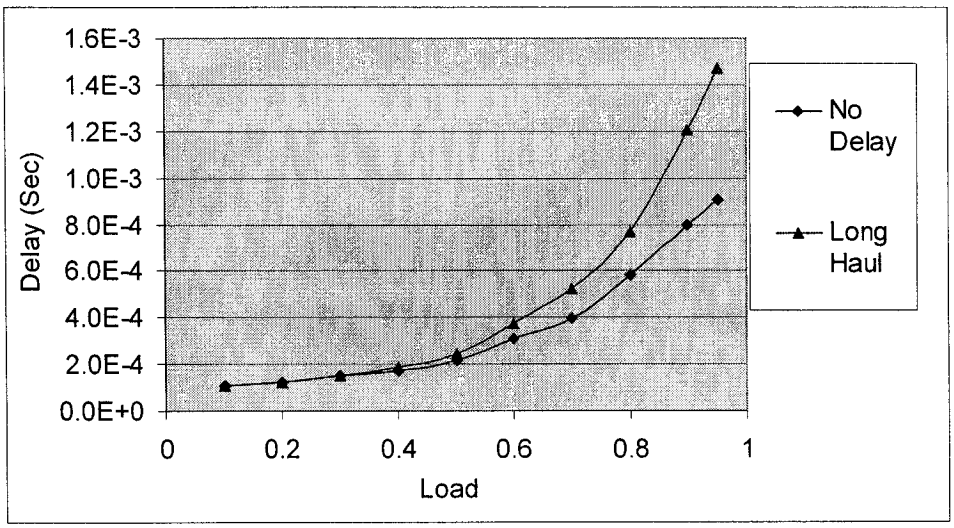

Figure 53 Average delay in edge node 1 versus load with τ = 2100 obtained for two different scenarios: network with no delay and network with long delay. The speed up is two and traffic is Bursty with non-uniform destination

4.5.4 AAPN-AWTA with Uniform IP Traffic

Figure 54 shows the average delay obtained at edge node 1 of AAPN with AWTA with no link propagation delay network with speed up of two for IP traffic arrival uniform destination distribution and different values of inter-configuration period τ. The traffic estimation, $\overline{\Lambda}_e$, is calculated with the equation 1 listed above. Both *t_time* and *s_time* are equal to 1 timeslots and no propagation delays for all links. All edge nodes have identically queuing delay. The increase in delay with respect is proportional to inter-configuration period τ. Both reconfiguration period and the different levels of load are the two major factors for the increase of delay. The queuing delay is around 50 μs for τ equals to 160 timeslots at 90% traffic load. The IP traffic simulations have much lower

5 Summary and Conclusions

A scheme for dynamic deployment of bandwidth that can be applied at two different levels of the network hierarchy has been described in this thesis. The scheme was implemented and tested first in the context of opto-electronic switches and then in the context of agile optical networks. It has been shown that the method succeeds in deploying the required bandwidth in both levels of the network.

In the context of opto-electronic switches, a three-stage packet switch architecture has been described, which consists of a reconfigurable optical interconnection as a central stage, surrounded by two electronic buffering stages. The proposed switch architecture is a hybrid optics/electronics Clos-like switch that includes both the electronics and photonics domains and makes the most of each technology's strengths: the electronics domain is used as a memory technology and the photonics domain is used as a communication technology. The Flexible Bandwidth Provision (FBP) algorithm is used to change the configuration of the optical centre stage to allocate the requested or desired bandwidth according to the incoming traffic. The scheme is such that it accommodates the relatively slow reconfiguration of current transparent photonic switch technology

The switch architecture and FBP method has already been demonstrated with self-similar and Poisson traffic under various test environments in previous work [1], where it was shown how the number of paths provided between sector pairs with the FBP algorithm adapts to the changing traffic statistics to obtain 100% throughput and how the delays are kept close to the ideal output queuing switch performance as a consequence of this adaptation. The first goal of this work was to implement the FBP switch architecture in a different platform (OPNET Modeler) that would be more suitable for further testing under more realistic traffic traces and within a whole network environment. It was important as well to validate the performance measures obtained previously with the implementation in C++ code.

The implementation in OPNET used for the performance analysis consists of a 64×64 switch with 16 crossbar switches in the central stage and 8 sectors in both the input and output stages. Each sector has eight external ports and therefore the internal speed up is two. The performance analysis for steady state response of the switch architecture with FBP shows that the queuing delay of packets in the switch increases with the inter-configuration period, τ (the time elapsed between successive configurations). The queuing delays in the output sectors dominate in the overall delay, except in the case of unit internal speed-up ($k = 1$). Three traffic models, Bernoulli arrivals with uniform destinations, Bernoulli arrivals with non-uniform destinations and Pareto distributed ON-OFF arrival processes with uniform destinations, have been simulated and tested with the FBP switch. The switch performance shows same results for traffic with uniform destinations regardless of the value of inter-configuration period τ (Figure 16). Nevertheless, the average delay with non-uniform destinations increases with the inter-configuration period τ. This indicates that, with a small speed-up, the switch needs to adapt only to slower persistant changes in traffic patterns and not to short fluctuations. The queuing performance measured from OPNET simulations confirms the results generated by C++. A major difference in the performance analysis between the two implementations was that the queuing delay measurement in C++ code considers the average queuing delay from all sectors, while the delay measurement in OPNET considers it sector by sector. Both results demonstrate similar switch behavior under the corresponding traffic models. For an internal speed-up of two, 100% throughput is achievable with no apparent restrictions.

In the context of agile optical networks, namely the AAPN topology, the bandwidth sharing scheme Adaptive Wavelength-Timeslot Allocation (AWTA) is introduced. This scheme is a mapping of FBP onto the next higher level of the network hierarchy and it corresponds to the distributed version of FBP where propagation delays have an impact in the scheduling.

AAPN proposes an overlaid-star topology with edge-nodes interconnected by the beams of the stars via core switches located at the hub of each star. The beams usually consist

of two or more fibers that transport wavelength division multiplexed traffic between edge-nodes and the bandwidth of each wavelength is shared in time by assigning varying numbers of timeslots to each flow of data. The core switches operate on a slotted mode to forward incoming traffic. It must be possible to create and destroy paths sufficiently rapidly that they endure for only microseconds.

In the AWTA scheduling algorithm, the traffic matrix is constructed from the queues information at each edge node. All edge nodes send the traffic information to the global controller in the core node every inter-configuration period, τ. Because of the propagation delays involved, the minimum τ must be at least two times the maximum core edge link propagation delay in the network (two allow for the two-way signaling of information) plus the time it takes to calculate the service matrix. The global controller computes the service matrix as described in the FBP architecture section after it receives all the traffic matrix information from all edge nodes. After the computation of the new service matrix, the core node transmits this new service matrix back to each edge node. Synchronization among the core node and all the edge nodes is essential in order for the core node to receive the new schedule at the exact time when the new configuration has been applied and in this way avoid collisions. A timing diagram has been presented in which propagation delays and synchronization between edge and core nodes have been considered.

The network simulated in this work contains one core node and 8 edge nodes. The connections between core node and edges nodes carry one wavelength at a link rate of 10 Gbps. The frame size is 1 ms and each timeslot is 10 μs in all scenarios. Five different network scenarios have been implemented and tested. The first network corresponds to a zero propagation delay between core and edge nodes, which is effectively the equivalent to the FBP switch (a collapsed AAPN). The second network is a metro network with fixed propagation delay equal to 10 timeslots between core and edge nodes. The third network is a metro network with different distances between core and edge nodes which are 0, 1, 2, 3, 7, 8, 9 and 10 timeslots delay in sequence. The fourth network is a long haul network with fixed propagation delay equal to 1000 timeslots delay. The last

scenario is a long haul network with different distances between core and edge nodes which are 0, 100, 200, 300, 700, 800, 900 and 1000 timeslots delay in sequence.

Four different traffic patterns have been used to test these network scenarios. The first traffic pattern is *Uniform Poisson* traffic which uses Poisson traffic arrivals with a uniform distribution of packet destinations. The second traffic pattern is *Non-uniform Poisson* traffic, which uses Poisson traffic arrivals with a non-uniform distribution of destinations. Both Poisson traffic models are the same as described in the FBP section of this work, with bi-stochastic matrices used to modulate the packets destinations in the latter model. The third traffic model is bursty traffic that uses an aggregation of ON-OFF Pareto-distributed traffic sources with uniform destinations distribution. The last traffic pattern implemented is a packet size distribution that corresponds to the statistics of IP traffic measured in real networks. The 4 different packet sizes used in the AAPN IP traffic simulation network are 40 bytes, 552 bytes, 576 bytes and 1500 bytes with 46 %, 18%, 18% and 18% of traffic load respectively. Packets destinations are uniformly distributed in this model. The results presented in this section correspond to the case in which a full timeslot is assigned to each packet, even if the packet is a lot shorter than the timeslot. For this reason, the queuing delays are larger than in previous tests since there is a big amount of wasted bandwidth.

The performance analysis shows that the network with the AWTA scheduling method succeeds in allocating the necessary bandwidth to the different traffic flows and that a 100% throughput is achieved for all traffic traces and network scenarios when there is a small network speed-up. The queuing delay in the edge nodes is proportional to the inter-configuration period, τ; i.e. the time between successive reconfigurations of the core node. This result is consistent with the results obtained in the FBP section. As expected, the queuing delay in those edge nodes closer to the core is smaller than that in the edges which are further away. In this thesis it has been shown that it is possible to map the FBP scheduling algorithm to the next upper layer in the network hierarchy by extending the algorithm with just a few considerations regarding propagation delays and synchronization.

5.1 Suggestions for Future Work

There are number of interesting research topics to be further developed as a continuation of this work:

- Test the FBP switch embedded in a full TCP/IP network environment. It is of great relevance to confirm that the switch architecture and scheduling algorithm perform well under network processes (e.g. TCP retransmissions) that are not explicitly modeled by the traffic models here presented.

- Perform an implementation and performance analysis of the AWTA allocation algorithm in a multiple star and multiple wavelength network environment in the AAPN context (the examined network in this thesis considers so far only a single star network with only one wavelength per link). The AWTA scheduling algorithm would be used to assign not only a different number of timeslots but also different wavelengths and even different routes; opening up many possibilities for the improvement of overall network performance.

- A slot aggregation process should be implemented in the case of IP traffic for both FBP and AAPN simulations. Both FBP and AWTA are designed for fixed size packets and the slot aggregation process is needed to be able to accommodate variable size packets traffic such as IP traffic. There is already a large research effort in this area being undertaken in the Photonic Network Technology Laboratory [1], the research group within which this work was produced.

- Compare the performance of AWTA in AAPN with other bandwidth allocation/scheduling methods in AAPN, such as Optical Burst Switching (OBS) as described in the AAPN research proposal [19]. The same topological and traffic parameters should be used to obtain a valid comparison of the various methods. There is

work being performed in the area of AAPN-OBS within the Photonic Network Technology Laboratory as well.

6 References

[1] Sofia A. Paredes, "Flexible Bandwidth Provision and Scheduling in a Packet Switch with an Optical Core", PhD thesis (University of London, 2005).

[2] S A Paredes, T J Hall. "Flexible Bandwidth Provision and Scheduling in a Packet Switch with an Optical Core ". OSA Journal of Optical Networking, vol. 4, no. 5, pp. 260-270, May 2005.

[3] M. Waldbvogel, G. Varghese, J. Turner and B. Plattner, "Scalable High Speed Prefix Matching," ACM Transactions on Computer System, vol. 19, no. 4, pp. 440-482. November, 2001.

[4] A. G. Kirk, W. A. Crossland, T. J. Hall, "A compact and scalable free-space optical crossbar," in *Proc. of the Third IEE International Conference on Holographic Systems, Components and Applications*, Edinburgh, IEE conf. publ., vol. 342, pp. 137-141, September 1991.

[5] W. A. Crossland, I. G. Manolis, M. M. Redmond, K. L. Tan, T. D. Wilkinson, M. J. Holmes, T. R. Parker, H. H. Chu, J. Croucher, V. A. Handerek, S. T. Warr, B. Robertson, I. G. Bonas, R. Franklin, C. Stace, H. J. White, R. A. Woolley, G. Henshall, "Holographic optical switching: the ROSES demonstrator," *J. Lightwave Technol.*, vol. 18, no. 12, pp. 1845-1854, December 2000.

[6] H. J. White, C. P. Barrett, M. J. Birch, J. R. Brocklehurst, N. A. Brwonjohn, W. A. Crossland, A. B. Davey, D. M. Monro, D. T. Neilson, J. A. Nicholls, G. M. Proudley, B. Roberston, R. W. A. Scarr, M. Snook, C. Stace, M. R. Taghizadeh, D. Vass, A. C. Walker, "Optically Connected Parallel Machine: Design, Performance and Application," *IEE Proc. – Optoelectronics*, vol. 146, no. 3, pp. 125-136, June 1999.

[7] R. W. A. Scarr, J. R. Collington, W. A. Crossland, M. P. Dames, "Highly parallel optics in ATM switching networks," *IEE Proc. - Optoelectronics*, vol. 144, no. 2, pp. 53-60, April 1997.

[8] T. J. Hall, "Vivaldi: variations on a theme of optical crossbars," in E. Marom, N. A. Vainos, A.A. Friesem, J. W. Goodman, (Eds.) Unconventional Elements for

Information Storage, Processing and Communications, (NATO-Series, Kluwer Academic Publishers, 2000), pp. 241-246.

[9] Y. S. Yeh, M. G. Hluchyj, A. S. Acampora, "The Knockout Switch - A Simple, Modular Architecture for High-Performance Packet Switching," IEEE *J. Select. Areas Commun.*, vol. 5, no. 8, pp. 1274-1283, October 1987.

[10] K. Y. Eng, M. J. Karol, Y. S. Yeh, "A Growable Packet (ATM) Switch Architecture: Design Principles and Applications," *IEEE Trans. Commun.*, vol. 40, no. 2, pp. 423-430, February 1992.

[11] S. T. Chuang, A. Goel, N. McKeown, B. Prabhakar, "Matching Output Queueing with a Combined Input/Output-Queued Switch," *IEEE J. Select. Areas Commun.*, vol. 17, no. 6, pp. 1030-1039, June 1999.

[12] C. S. Chang, W. J. Chen, and H. Y. Huang, "Birkhoff-von Neumann input-buffered crossbar switches for guaranteed-rate services," *IEEE Trans. Commun.*, vol. 49, no. 7, pp. 1145-1147, July 2001 .

[13] C. S. Chang, D. S. Lee, Y. S. Jou, "Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering," *Computer Communications*, vol. 25, no. 6, pp. 611-622, April 2002.

[14] C. S. Chang, D. S. Lee, C. M. Lien, "Load balanced Birkhoff-von Neumann switches, part II: multi-stage buffering," *Computer Communications*, vol. 25, no. 6, pp. 623-634, April 2002.

[15] S. Iyer, N. McKeown, "Analysis of the parallel packet switch architecture," *IEEE/ACM Trans. Networking*, vol. 11, no. 2, pp. 314-324, April 2003.

[16] Y. Wang, W. A. Crossland, R. W. Scarr, "Modelling for optically interconnected packet switches," in Proc. SPIE, vol. 4213, pp. 44-55, 2000.

[17] C. W. Wilmsen, C. J. Duan, J. R. Collington, M. P. Dames, W. A. Crossland, "Vertical cavity surface emitting laser based optoelectronic asynchronous transfer mode switch," *Optical Engineering*, vol. 38, no. 7, pp. 1216-1222, July 1999.

[18] W. E. Leland, M. S. Taqqu, W. Willinger, D.V. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Trans. Networking*, vol. 2, no. 1, pp. 1-15, February 1994.

[19] G. V. Bochman et. al. "22nd The Agile All-Photonic Network: An Architectural outline," *Proc.* Biennial Symposium on Communication, pp.217-218, 2004.

[20] A. Banerjee, J. Drake, J. Lang, B. Turner, K. Kompella, and Y. Rekhter, "Generalized Multiprotocol Label Switching: An Overview of Routing and Management Enhancements" IEEE Communication Magazine vol. 39, no.1, pp. 144-150, January 2001.

[21] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," IEEE Transactions pm Communication, vole.47, no.8, pp. 1260-1267, August 1999.

[22] M. E. Crovella, A. Bestavros, "Self-similarity in World Wide Web Traffic: Evidence and Possible Causes," IEE/ACM Transactions on Network, vol.5, no.6, pp. 835-846, December 1997.

[23] OPNET Technology Inc, (April, 2005). *OPNET [WWW].* Available: http://www.OPNET.com/

[24] OpNet Modeler Release 10.5 PL1, Simulation Kernel Reference Manual, OpNet Technologies Inc., March 2004.

[25] Vern Paxson, "Fast Approximation of Self-Similar Network Traffic," Technical Report, Lawrence Berkeley Laboratory and EECS Division; University of California, Berkeley. April 20, 1995.