Characterisation of a Wavelength Division Multiplexing Multi-Ring Network

By
Christophe Jelger
(BEng Honours)

Thesis submitted to the University of Wales in
Candidature for the Degree of Master of Philosophy

Supervisor: Professor Jaafar M. H. Elmirghani

Department of Electrical & Electronic Engineering
University of Wales Swansea
September 2001

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ……..(candidate)

Date ……..

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated.

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed ……..(candidate)

Date ……..

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ……..(candidate)

Date ……..

Acknowledgements

I would like to sincerely thank Professor Jaafar Elmirghani for his invaluable expertise, guidance and encouragement.

I would also like to thank the students of the Optical Communications Research Group for their friendship and support over the past year. To all of them I would like to express my sincere greetings.

Finally, I would like to thank my parents, to whom I dedicate this thesis, for having the belief and confidence in me.

SUMMARY

This thesis considers a metropolitan WDM slotted ring network for which an original architecture has been proposed. In the past there has been a number of studies on multi-channel slotted rings for electronic networks. However, these studies are no longer applicable to modern telecommunication networks which exhibit a very large bandwidth-delay product and where multiple simultaneous transmissions are made possible by the large number of slots rotating around the ring.

An original node architecture has been designed by using one fixed transmitter and four fixed receivers (FT-FR^4) at each node. Previous studies on WDM multi-rings have considered one fixed receiver and one tuneable transmitter (TT-FR), or a fixed number (M) of both transmitters and receivers (FT^M-FR^M), with M being the number of wavelengths within a fibre. The proposed FT-FR^4 architecture has been shown to be very flexible and more easily scalable than previous proposals of WDM ring networks.

The performance of the network has been thoughtfully and extensively evaluated through theoretical analysis and simulations. A network simulator has been specifically developed to model the proposed network. Moreover, a self-similar traffic model has also been designed to simulate realistic bursty network traffic. The results have shown that the proposed architecture can accommodate a large number of access nodes with a limited number of wavelengths.

The four fixed receivers have latter been replaced by a tuneable receiver. This modification introduces receiver collisions as a node can receive more than one packet simultaneously on different wavelengths. A innovative receiver collision avoidance mechanism has therefore been proposed and it has been shown through extensive simulations that the performance of the network was unaffected by this scheme.

CONTENTS

List of Figures

List of Tables

List of Abbreviations

ACK	Acknowledgement
AF	Allocation Free
AP	Access Point
AN	Access Node
ASK	Amplitude Shift Keying
ATDC	Asynchronous Transfer on Data Channel
ATM	Asynchronous Transfer Mode
BSD	Berkeley Software Distribution
CA	Collision Avoidance
CSMA	Carrier Sense Multiple Access
DA	Destination Allocation
DAC	Data Access Control
DFB	Distributed FeedBack
DPSK	Differential Phase Shift Keying
DT-WDMA	Dynamic Time - Wavelength Division Multiple Access
DWDM	Dense Wavelength-Division Multiplexing
EDFA	Erbium Doped Fibre Amplifier
FIFO	First-In First-Out
FR	Fixed Receiver
FT	Fixed Transmitter
GbE	Gigabit Ethernet
GNU	Gnu is Not Unix
IAT	Inter-Arrival Time

ID	Identifier
IP	Internet Protocol
LAN	Local Area Network
MAC	Medium Access Control
MAN	Metropolitan Area Network
MTU	Maximum Transfer Unit
NAL	Node Activity List
OADM	Optical Add-and-Drop Multiplexer
PAC	Packet Against Collision
QoS	Quality of Service
RCA	Receiver Collision Avoidance
RCD	Receiver Collision Detection
RSQ	Reception Scheduling Queue
RSR	Receive Shift Register
RWA	Routing and Wavelength Assignment
SA	Source Allocation
SCM	Sub-Carrier Multiplexing
SDH	Synchronous Digital Hierarchy
SONET	Synchronous Optical NETwork
TCP	Transmission Control Protocol
TDM	Time-Division Multiplexing
TR	Tuneable Receiver
TT	Tuneable Transmitter
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing

Glossary of Symbols

Symbol	Definition
B_{DP}	Bandwidth-delay product per wavelength
D	Ring propagation delay
H	Hurst parameter
L	Gigabit Ethernet access link load
L_i	Load generated by an ON-OFF source I
L_N	Normalised network load
L_R	Ring length
m	Aggregation level
M	Number of ON-OFF sources
N	Number of nodes per wavelength
N_T	Total number of nodes within the network
N_W	Number of wavelengths per fibre
μ_i^{ON}	Mean duration of ON period of source i
μ_i^{OFF}	Mean duration of OFF period of source i
R_N	Total network transmission rate
R_{MIN}	Equivalent bit rate of S_{MIN}
R_W	Wavelength rate
S	Slot size
S_{MIN}	Average number of slots released
S_W	Number of slots per wavelengths
S_r	Slot-reuse factor

t	Time
T_{MAX}	Maximum throughput per node
$T_{MAX\text{-}S}$	T_{MAX} with source-stripping
$T_{MAX\text{-}D}$	T_{MAX} with destination-stripping
V	Light velocity in fibre
X_t	Incremental process of $Y(t)$
$X_s^{(m)}$	Aggregated process of X_t
$Y(t)$	Packet arrival cumulative process
α	Shape parameter of Pareto distribution
β	Scaling parameter of Pareto distribution
δt	Minimum time increase in simulator
n_i	Node number i (i=1, 2, …, N_T)
Λ	Poisson IAT
λ_j	Wavelength number j (j=1, 2, …, N_W)
η	Bandwidth efficiency
η_s	Bandwidth efficiency with source-stripping
η_d	Bandwidth efficiency with destination-stripping
τ	Packet duration of generated traffic (i.e. 12 μs)

Chapter 1

Introduction

In the past few years, there has been an enormous increase in the bandwidth requirements of modern telecommunications networks. This has been mainly driven by the massive and ever increasing popularity of the Internet and its related multimedia applications. This is already causing a fundamental paradigm shift in the scale of bandwidth required in the transport network and this tremendous expansion is expected to persevere for another five to ten years. Conversely, the network infrastructure built to date has largely been designed to carry voice traffic and it does not have the necessary capabilities to handle such a steady increase in the bandwidth requirements. Furthermore, network carriers have faced an unexpectedly rapid exhaustion of fibre availability and expensive major fibre installation programs cannot be considered as the cost of the current infrastructure has just been compensated. There has also been a sudden need to create a more open approach for creating capacity as the bandwidth demand increase is not expected to slow down.

To develop the robust and efficient networks that satisfy the requirements above, new network architectures and technologies are required. Photonic network technologies have emerged as the only suitable solution and dense wavelength-division multiplexing has been widely adopted as the technology of choice for increasing the transmission capacity of carrier networks. The WDM technology indeed allows multiple optical circuits to be multiplexed within a single fibre and each wavelength in the fibre may be considered a virtual fibre. Moreover, the capacity of the network increases with the number of available wavelengths, and new advances have shown that the maximum number of wavelengths is far from being reached. This is very attractive for carriers who can rely on WDM technology to provide the necessary transmission capacity in order to satisfy the ever increasing bandwidth demand.

There has consequently been a very large and rapid deployment of WDM systems by long distance carriers throughout North America, Japan and, more recently, Europe. In most cases, the WDM technology has mainly been used to carry signals from multiple SONET/SDH terminals in which the “real” networking functionality was actually implemented. More recently, the emergence of WDM into a full logical network layer has already made it possible to launch traffic from higher layers (i.e. ATM and IP) into a self-healing optical network, and specifically without the need for traditional SONET/SDH network elements. Moreover, add-and-drop systems, that enable any wavelength to be added or dropped at a single site without demultiplexing the entire wavelength bundle, have now been developed and have extremely improved the flexibility of WDM systems.

Furthermore, while WDM has undoubtedly become the solution adopted in order to increase the capacity of long-haul wide area networks, its deployment has also recently been considered in the metropolitan access networks. The increased demand for additional bandwidth has indeed also been witnessed in the local exchange network as new broadband services, such as broadcast video and high-speed interoffice links, have increasingly augmented the bandwidth requirements of metropolitan networks.

However, despite the fact that there is already a number of commercially available systems that are ready to be installed, there are still many challenges that have to be resolved before WDM can widely be deployed in the metropolitan networking environment. Networks in the metropolitan area must be very flexible and scalable in the sense that incremental capabilities are required in order to upgrade the network capacity when needed. Moreover, advanced network management techniques are required in order to control, inter-operate and co-ordinate WDM systems which are manufactured by different vendors. Therefore, a key factor for interoperability will undoubtedly be the emergence of both manufacturing and management standards. And finally, one must bear in mind that the metropolitan networking market is highly cost driven and that economic considerations will always be a strong issue when designing and implementing WDM access networks.

In this project, particular attention has therefore been given to the above requirements in order to propose a realistic and realisable architecture suited for metropolitan access network market.

1.1 Research objectives

The primary objectives of the work undertaken and presented in this thesis were to:

- Study and understand the different architectures and MAC protocols used in WDM networks.

- Design a logical architecture and a MAC protocol for metropolitan WDM ring networks.

- Develop a mathematical model to predict the network performance and estimate the scalability of the architecture.

- Design a network simulator using accurate network traffic sources (including self-similar traffic) to evaluate the performance of the proposed architecture.

- Evaluate the performance of an improved implementation of the MAC protocol using destination-stripping.

- Design and evaluate a mechanism to handle receiver collisions in the specific case where a tuneable receiver is to be used.

- Analyse and evaluate the performance when unbalanced traffic patterns are used in the network.

1.2 Organisation of the thesis

Following the introduction in Chapter one, the rest of the thesis is organised as follows:

Chapter two presents a review of WDM architectures and protocols. Consideration is given to wavelength routing networks and to broadcast-and-select networks. Network topologies are described and compared. Finally, existing ring architectures and associated MAC protocols are introduced, and their limitations are highlighted

Chapter three presents the proposed ring architecture, the new node design, and the MAC protocol structure. A theoretical model is introduced and evaluated. The model is used to predict network performance and the scalability of the proposed architecture.

Chapter four considers the design and implementation of a discrete-event network simulator which was developed specifically to assess the performance of the studied architecture. Two different traffic models were implemented to simulate realistic traffic patterns. Details of the simulator operation are presented.

Chapter five focuses on simulation results. The performance of the simulated network using the two different traffic sources is evaluated and compared with the theoretical predictions. The impact of using different traffic sources is also discussed.

Chapter six introduces an improved version of the architecture. The impact of receiver collisions is studied and a solution that can handle this problem is presented and evaluated. The effect of using unbalanced traffic sources is also analysed and discussed.

Chapter seven summarises the major contributions and conclusions of this thesis.

Chapter eight highlights areas for further investigation.

1.3 Original Achievements and Contributions

The author has

1. Proposed an original WDM ring network architecture and developed a novel analytical approach that can predict the bandwidth efficiency of the network. The scalability of the proposed architecture has been shown to be very attractive in the metropolitan networking environment.

2. Developed a complete network simulator and two traffic generators in order to simulate and evaluate the performance of the proposed architecture. The simulator is an original development which has been thoughtfully designed in a modular and flexible manner. Moreover, the validity of the traffic sources was demonstrated by measuring the variance of the generated traffic over a wide range of time scales and a crucial design parameter was optimally derived.

3. Evaluated the performance of the simulated network for two comparable realistic architectures. Both source-stripping and destination-stripping schemes were considered and the network performance was found to be very attractive even in the presence of highly bursty self-similar traffic.

4. Proposed a modified original node architecture and evaluated the effect (i.e. receiver collisions) induced by the use of a tuneable receiver. Furthermore, the effect of using unbalanced traffic sources was analysed. The proposed network was shown to perform in a satisfactory way under these conditions.

These original contributions are supported by the following publications:

1. C. S. Jelger, and J. M. H. Elmirghani, "A Simple MAC Protocol for WDM Metropolitan Access Ring Networks," *in Conference Proceedings, IEEE Globecom'01*, San Antonio, Nov. 2001, in press.

2. C. S. Jelger, and J. M. H. Elmirghani, "Performance Evaluation of a new MAC Protocol for WDM Metropolitan Access Ring Networks," *International Journal of Communication Systems*, in press.

The following articles are also being considered for publication:

1. C. S. Jelger, and J. M. H. Elmirghani, "Performance of a slotted MAC Protocol for WDM Metropolitan Access Ring Networks under Self-Similar traffic," submitted to the *IEEE ICC'02 Conference*.

2. C. S. Jelger, and J. M. H. Elmirghani, “Characterisation of a WDM Metropolitan Multi-Access Ring with FT and TR under Self-Similar Traffic,” submitted to *IEEE/ACM Transactions On Networking*.

3. C. S. Jelger, and J. M. H. Elmirghani, “WDM Multi-Ring Access Network Performance under Self-Similar Traffic with Balanced and Unbalanced Loads,” submitted to *IEE Proceedings in Communications*.

Chapter 2

Review

2.1 Introduction

WDM lightwave networks can be classified as either broadcast-and-select networks or wavelength routing networks [1-4]. In a broadcast-and-select network, a node's transmission is broadcast to all other nodes. The receiver at the destination extracts the desired signal from the entire group of signals transmitted. In wavelength routing networks, the wavelength of a signal is used to route it through the network. The routing can be either fixed or dynamic. Due to their scalability limitations [3] and their natural multicasting ability, broadcast-and-select networks are very suitable for local and metropolitan area networks [4]. On the other hand, wavelength routing networks are more appropriate for wide area networks [5] as they are more flexible in terms of scalability and modularity. It is also worth mentioning the simple WDM point-to-point link, a degenerate form, as this is not a network in the usual sense. These links were used in the early deployment of WDM technology as a solution to increase bandwidth availability [5].

Both broadcast-and-select and wavelength routing networks can be further classified into single-hop or multi-hop [3,6-8]. Single-hop networks allow direct communication between any two nodes. Data travels optically without conversion into electronic form until it reaches its destination. In a multi-hop network, a transmission may have to travel through intermediate nodes before reaching its destination. At each intermediate node, the data is switched electronically to the appropriate next node and then retransmitted in light form. This may be needed if there is no common wavelength between two nodes that wish to communicate.

Different topologies have been considered in both broadband-and-select and wavelength routing networks. Three basic physical architectures have been used, namely the bus, ring, and star topologies [8]. The more complex mesh topology is also widely studied due to its high flexibility and scalability features. One must also be careful to make a distinction between physical architectures and logical topologies. The former is the actual physical implementation of the network, whereas the latter describes the virtual topology seen by data being transferred between the nodes on the network.

Moreover, both broadcast-and-select and wavelength routing networks can be used to implement circuit switching and packet switching. Advantages, drawbacks and applications of all possible cases were presented by Green [5] and will be detailed in the following sections of this review.

Finally, in all the above cases, a protocol is needed to allow nodes to share access to the multi-wavelength network. As there are many possible WDM network

implementations, a very large number of protocols were proposed and studied over the last decade [9], particularly for the star topology [10]. They will be attentively presented in the next sections.

We will first present the early deployment and use of WDM technology which was mainly driven by the increasing demand for communication bandwidth. In the next section, we will explore the broadcast-and-select star networks, describe the existing architectures and protocols, and discuss some of the key considerations when designing such a network. Next, we will focus on wavelength routing networks, consider their main features and introduce the routing and wavelength assignment (RWA) problem. In the following section, we will concentrate on ring topologies, review the existing implementations and protocols for classical ring networks, and finally in the last section, we will present a number of proposed WDM ring architectures.

2.2 Point-to-Point WDM Systems

In late 1995, WDM technology became commercially available as a solution to the rapid exhaustion of the capacity of long distance fibre networks [11-13]. In practice, there were three basic alternatives that carriers could consider in adding capacity to their networks: adding more fibre, increasing the transmission baud rate, or implementing WDM systems in combination with TDM technology [11]. The cost reduction of installing WDM systems (compared to more costly solutions), the sudden need to create a more open ended approach to creating capacity, and the promising

performance and improvements of WDM transmission fuelled the exceptionally rapid adoption of this technology by long distance carriers throughout North America [11].

However, these early deployments were merely in the form of point-to-point links where independent TDM data streams, each on a unique wavelength, were multiplexed and sent on a fibre, and demultiplexed (separated) at the fibre's receiving end as seen in Figure 2.1.

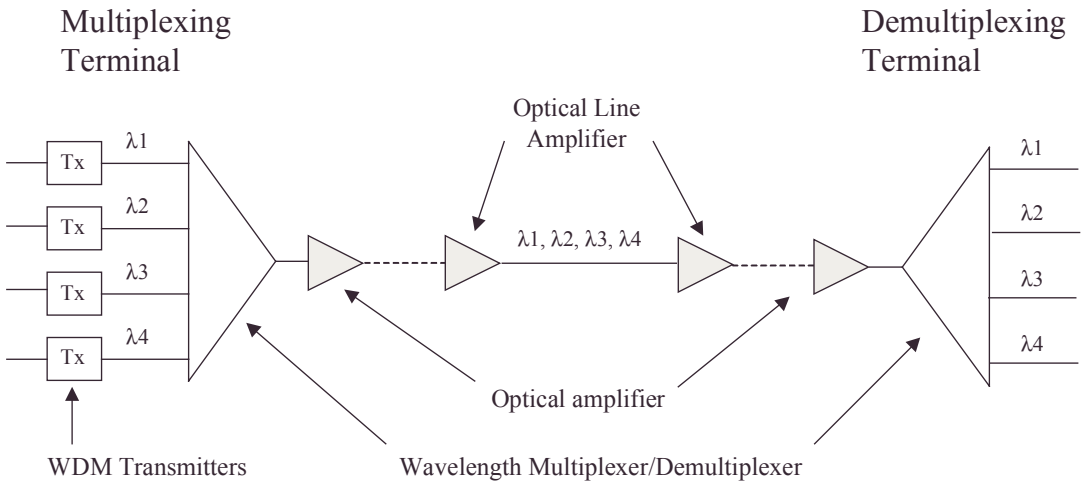


Figure 2.1. Point-to-Point WDM link [13]

In 1971, investigators at IBM built a five-wavelength system running at several Mb/s per wavelength as a laboratory demonstration of the potential transmission capacity of single WDM links [14]. By 1996, a number of commercially systems were available and were listed by Green [5], with the IBM 9729 product [15] already allowing (at the time) up to 20 wavelengths per single fibre. In October 2000, 64-wavelength systems were commercially available and the number of channels was still expected to

increase rapidly [13]. Indeed, the Alcatel 1640 OADM (Optical Add/Drop Multiplexer) nowadays offers up to 80 channels, each transmitting at up to 10 Gb/s.

2.3 Broadcast-and-select networks

Broadcast-and-select networks were introduced in the early 1980's and the first network to be built in the laboratory was the four-wavelength analogue video distribution system designed by Toshiba in 1985 [16].

The salient feature of a broadcast-and-select network is its ability to share information between users on a common medium. Indeed, the optical information streams transmitted by multiple sources on different wavelengths are broadcast to all other nodes in the network. The receiver at the destination, using an optical filter, extracts (selects) the desired signal from the entire group of signals transmitted. A shared medium is thus required and the most popular architecture to date is the star configuration, which has been extensively studied and implemented in the last decade. The star was preferred to the two other basic architectures, namely the bus and the ring, because it has a superior power budget [17,18]. The ring network, however, is very attractive due to its resilience against fibre breaks [19]. Moreover, the progress in optical amplifier technology and the advent of EDFAs (Erbium Doped Fibre Amplifiers) has enable the designers to overcome problems associated with the inferior power budget of ring networks which was mainly due to insertion losses at intermediate nodes. Recent research interests in WDM ring networks are presented in Section 2.6 of this review and this section will mainly focus on the star topology.

In the star network, all of the input signals transmitted by the network nodes are optically combined in a WDM passive coupler [20, 21], and the mixed optical information is broadcast to all of the nodes on their receive fibres. The star coupler is a piece of glass that combines the optical signal it receives on any of its input ports and equally forwards it to all of its output ports. Moreover, no power is needed to operate the coupler and there is no tapping or insertion loss as in a linear bus or ring architecture as these were thoughtfully compared by Henry [22] and Nassehi et al [23]. Selection at the destinations nodes is carried out by means of either fixed wavelengths or tuneable optical filters. Figure 2.2 shows a possible implementation of such a network.

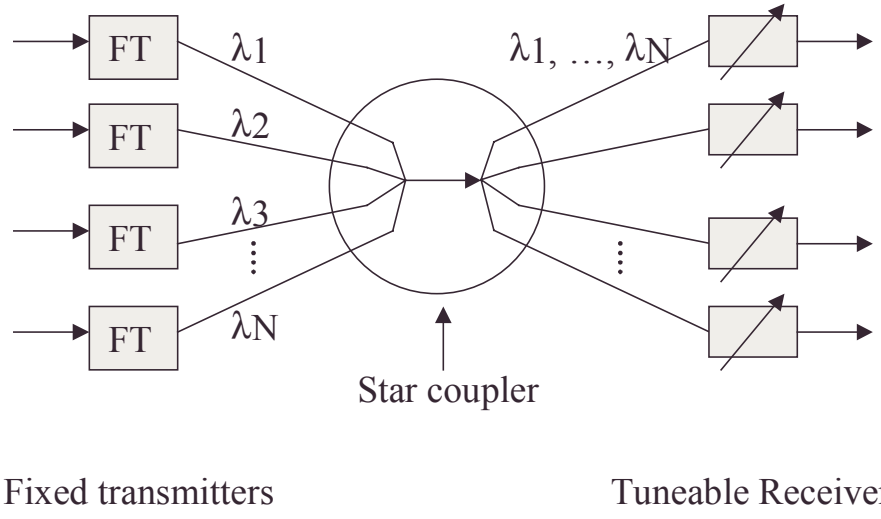


Figure 2.2. A broadcast-and-select star WDM network [6]

Indeed, each network node is typically equipped with a number of transmitters and receivers. These transceivers may be tuned to a specific wavelength or they may be dynamically tuneable to different wavelengths. In [4], Elmirghani et al give an extensive review and evaluate the enabling technologies and latest developments in

designing tuneable transmitters and receivers. Many combinations using fixed and/or tuneable transceivers can be realised and a large number of architectures were proposed and studied during the last decade. Four basic structures can be implemented as described by Mukherjee [6] and Senior et al [8]:

- Fixed transmitter(s) and fixed receiver(s) : FT-FR
- Tuneable Transmitter(s) and fixed receiver(s) : TT-FR
- Fixed transmitter(s) and tuneable receiver(s) : FT-TR
- Tuneable transmitter(s) and tuneable receiver(s) : TT-TR.

References [6] and [8] also introduce the use of superscripts to indicate the number of each component (no superscript indicates one component) at each node. For example, FT^{i}-TR^{j} indicates *i* fixed transmitters and *j* tuneable receivers at each node.

Finally, references [6], [7] and [8] introduce single-hop and multi-hop networks. In a multi-hop network, all nodes can act as intermediate routing nodes in order to achieve connectivity between any arbitrary pair of nodes. At each intermediate node, the data is switched electronically to the appropriate next node and then retransmitted in light form. In single-hop networks, data travels directly between two nodes without being rerouted by intermediate nodes.

2.3.1 Single-hop experimental networks and protocols

Several experimental single-hop network prototypes [24-28, 34-42] based on the broadcast-and-select star approach have been built and in the mean time, a large

number of protocols have also been proposed [29-33, 43-48]. Indeed, a significant amount of dynamic co-ordination between nodes is required. For a transmission to occur, the sending node and the destination node must agree to transmit and receive data on the same wavelength. One of the main design issue in broadcast-and-select networks is therefore the development of efficient protocols to allocate wavelength channels for different connections within the network. Such a mechanism must resolve wavelength contention, avoid or minimise collisions and provide some means of co-ordination among the nodes within the network. These protocols are further classified by Mukherjee [6] into two categories: those employing pretransmission co-ordination, and those not requiring any pretransmission co-ordination. In the first case, a shared control channel, possibly embedded on the data channels, is required to arbitrate the concurrent transmissions which take place through the data channels. On the other hand, no such control channel is needed in systems that do not employ any pretransmission co-ordination.

The following sections elaborate on the various experimental demonstrators and on the most popular protocols that were designed for single-hop star networks.

2.3.1.1 Systems based on no pretransmission co-ordination

In the first part of this section, two experimental demonstrators [24-28] are presented and their main characteristics are described. In each case, a short paragraph also outlines the main developments and progress that were achieved. Protocols [29-33] which have been designed for single-hop star networks will then be discussed in the second part of this section.

2.3.1.1.1 LAMBDANET

The LAMBDANET demonstrator [24, 25] is a FT-FR^M system which was developed at Bellcore in the late 1980'. The architecture (Fig. 2.3) is composed of M nodes connected by single-mode optical fibres to a hub location where the fibres are passively coupled by a star coupler.

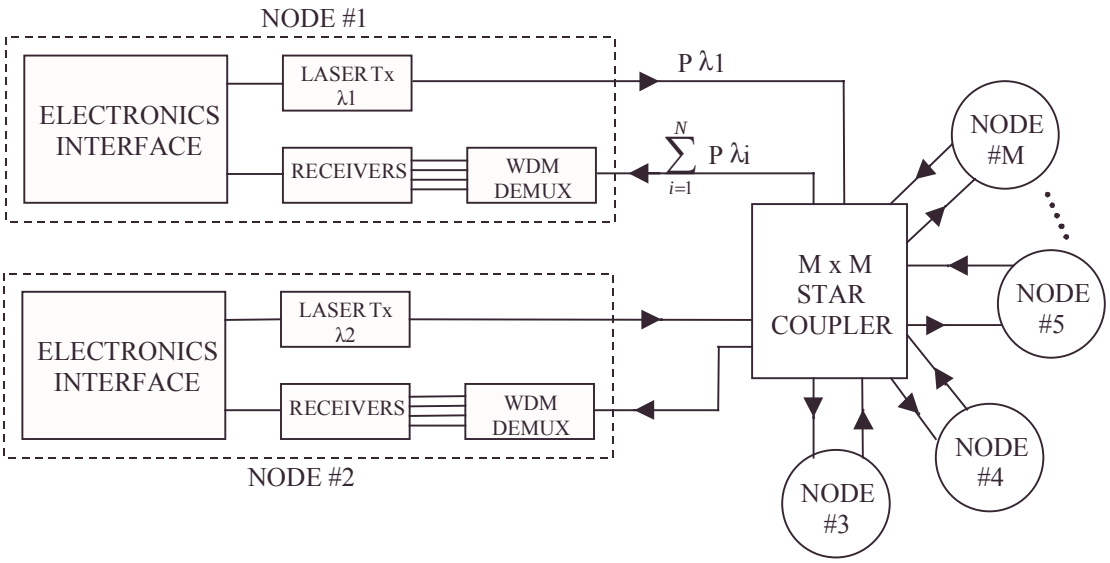


Figure 2.3. Block diagram of the LAMBDANET star network [24]

This experiment demonstrated the use of an array of M receivers at each node in the network, employing a grating demultiplexer to separate the different optical channels. Each transmitter time-division multiplexes its traffic destined for all other network nodes into a single high-speed data stream to modulate its own laser. Each receiving node simultaneously receives all of the traffic of the entire network, with a subsequent selection, by electronic circuits, of the traffic destined for that node. This creates an internally passive, nonblocking, and completely connected network.

The capabilities of LAMBDANET were demonstrated in three experiments as described in [24] and summarised in Table 2.1. The first experiment used 18 wavelengths, each running at 1.5 Gb/s with a transmission distance of 57.8 km. This represent the transmission of 27 Gb/s (18×1.5 Gb/s) over 57.8 km on each of 18 output fibres for a point-to-point figure of merit of 1.56 Tb×km/s (27 Gb/s×57.8 km) and an aggregate point-to-multipoint figure of merit of 21.5 Tb×km×node/s. In the second experiment, the bit rate was increased to 2 Gb/s but, due to laser chirping, fibre dispersion, and limited power, only 16 wavelengths could be used and the transmission distance was reduced to 40 km, resulting in lower figures of merit. For the third experiment, labelled "WDM" in Table 2.1, the star was replaced by a second grating multiplexer, increasing the available power at the receiver, with the results that 18 wavelengths were successfully transmitted at 2 Gb/s over 57.5 km as shown in the table.

	POINT-TO-POINT Tb×km/s	POINT-TO-MULTIPOINT Tb×km×node/s
LAMBDANET 1.5 Gb/s, 18 λ's, 57.8 km	1.56	21.5
LAMBDANET 2.0 Gb/s, 16 λ's, 40 km	1.28	18.0
WDM 2.0 Gb/s, 18 λ's, 57.5 km	2.07	….

Table 2.1. LAMBDANET experimental results [24]

There were several new results from the LAMBDANET demonstrator. The first was the architecture itself with its nonblocking, fully connected properties. The second was the demonstration of dense WDM with prototype-packaged components and

lasers. The third was the set of new transmission records noted above. Moreover, the LAMBDANET concepts served as a foundation for several other architectures.

2.3.1.1.2 Fast Optical Cross Connect (FOX)

The goal of the Fibre Optic Cross Connect (FOX) demonstrator [26, 27] was to examine the potential of using fast tuneable lasers and fixed receivers in a parallel processing environment. Each node was designed as a TT-FR system and the architecture employed two star couplers as shown on Figure 2.4.

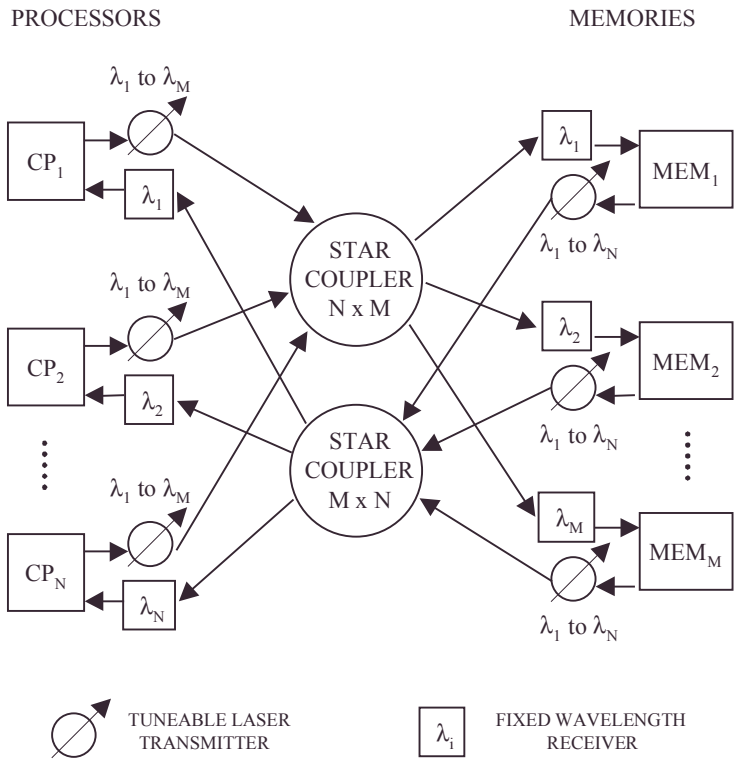


Figure 2.4. FOX: illustrating the use of tuneable lasers [26]

One network was used to transmit signals from the *N* processors to the *M* memories and the other was used to transmit in the reverse direction. To address a particular memory, a processor tunes its transmitter to the appropriate wavelength and selection is done at the memory's receiver by the fixed-tuned filter, and vice versa in the opposite direction.

However, in such a network, the possibility of contention exists, and may affect the overall throughput or response time. As the utilisation of the memory access was relatively slow as described in [26], a binary exponential-backoff algorithm was shown to be sufficient to obtain good performance.

The technology issue was to demonstrate the fast tuneability of lasers so that the network was not slowed down waiting for the transmitter to tune to the desired wavelength. Transmitter tuning times less than a few tens of nanoseconds were shown to be reasonably efficient for data packets ranging from 100 ns to 1 μs. The tuneable transmitters were described with details in a paper from the same research team [28].

2.3.1.1.3 Protocols based on no pretransmission co-ordination

A very large number of protocols that do not require any pretransmission co-ordination were proposed and evaluated. In this section we will only concentrate on the most important realisations [29-33].

Chlamtac [29], Ganz [29, 30], and Gao [30] introduced simple techniques that allow multiple single-hop communications. Those are based on fixed, or partial fixed,

assignment methods where time division multiplexing (TDM) is used over the WDM multi-channel environment. Time is divided into cycles, and it is predetermined at what point in a cycle and over what channel a pair of nodes is allowed to communicate. These protocols were designed to be used with TT-TR systems. In the following examples, we consider a simple case consisting of three nodes and two channels. In Tables 2.2, 2.3 and 2.4, an entry (i, j) for channel k in time slot l means that node i has exclusive permission to transmit a packet to node j over channel k during time slot l. In [29], the authors introduced techniques to derive the allocation matrix for any arbitrary pair of numbers M (number of nodes) and N (number of channels). Table 2.2 illustrates the case where fixed assignment is used.

Channel No	t	t+1	t+2
0	(1, 2)	(1, 3)	(2, 1)
1	(2, 3)	(3, 1)	(3, 2)

Table 2.2. An example of fixed assignment protocol [6]

This scheme has many limitations as it does not accommodate bursty traffic and performance is greatly affected as the number of nodes increases. An improved version was proposed by Ganz [30] where an optimised algorithm is used in order to minimise the need for transceivers tuning. Chlamtac [29] also introduces partial fixed assignment protocols. In these schemes, the channel allocation procedure is less restrictive than in the previous category. Indeed, with fixed assignment, both channel collisions and receiver collisions are avoided. With partial fixed assignment, the possibility of either channel collisions or either receiver collisions is introduced. In the destination allocation (DA see Table 2.3) protocol, more than one source are

allowed to transmit on a given channel to a unique destination node during a time slot. In this scheme, channel collisions are therefore possible. Indeed, if node 1 and node 3 want to transmit to node 2 during the same time slot, a channel collision occurs.

Channel No	t	t+1
0	(1, 2) (3, 2)	(1, 3)
1	(2, 3)	(3, 1) (2, 1)

Table 2.3. An example of destination allocation protocol [6]

A source allocation (SA see Table 2.4) protocol was also considered. In this case, only one node is allowed to transmit on a given channel but it is now allowed to transmit to more than one node. As a result, receiver collisions are introduced. For example, if node 1 and node 2 want to transmit to node 3 during the same time slot a receiver collision will occur at node 3.

Channel No	t	t+1	t+2
0	(1, 2) (1, 3)	(1, 3) (1, 2)	(2, 1) (2, 3)
1	(2, 3) (2, 1)	(3, 1) (3, 2)	(3, 2) (3,1)

Table 2.4. An example of source allocation protocol [6]

Finally, an allocation free (AF) protocol (which is a degenerated case of assignment protocol) can be defined in which source-destination pairs have full rights to transmit on any channel over any slot.

In the above protocols, the destination receiver is tuneable, although it is known *a priori* which channel it will be tuned to during the various time slots in a cycle. The time taken by the receiver to switch from channel to channel over consecutive slots may be large and, consequently, protocols have been developed to be used with fixed tuned receivers. These schemes therefore require the use of TT-FR systems. Dowd [31] proposed two slotted-ALOHA protocols. These were classified by Mukherjee [6] as random access protocols. Under the two protocols proposed, time is slotted (in data slots and acknowledgement (ACK) sub-slots) on all the channels. The disadvantage of these protocols is that collisions can occur at the star coupler if two nodes transmit to a common destination during the same time slot. Therefore, a collision-less acknowledgement mechanism was introduced to detect/notify the occurrence of data collisions. A node which correctly receives a packet during a data time slot will transmit an acknowledgement to the sender during the immediately following ACK sub-slot, and it will reach the destination without collision, because no other node could have received data from the same sender during the data time slot. A node assumes a collision has occurred if the ACK is not received. Different implementations of slotted-ALOHA were proposed by Ganz and Koren [32] but those were designed to be used with TT-FR^x systems (where x is a system parameter).

As discussed by Dowd [31] and Ganz et al [32], random access protocols handle bursty traffic better than fixed or partial fixed assignment protocols but they have limited throughput. Karol and Glance [33] later proposed the packet-against-collision (PAC) protocol to avoid packet collisions. These collisions are avoided by allowing a node access to a channel only if the channel is available. Also, packets simultaneously accessing the same channel are denied access. As shown in Figure 2.5, a PAC circuit

is used to carrier sense the channels before a node is allowed to transmit to one of them.

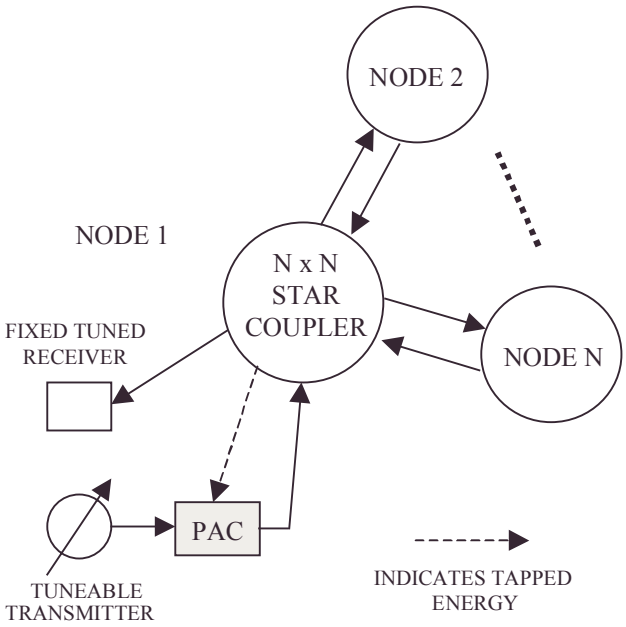


Figure 2.5. Architecture of the PAC optical packet network [33]

When two or more nodes try to access the same channel simultaneously, all of them detect the carrier (an *n*-bit burst that precedes the packet) and their access to the network is blocked. Blocked packets are reflected back to the sender. This mechanism actually required an additional control star coupler (not represented on Figure 2.5) and the architecture was not attractive to LAN and MAN network because of its hardware complexity.

2.3.1.2 Systems based on pretransmission co-ordination

In this section, three experimental demonstrators [34-42] are described. The first two [34-37, 38] use a separate control channel while the third architecture implements an

in-band polling mechanism. A few protocols [43-48] are then presented and their main features are identified.

2.3.1.2.1 Hybrid Packet Switching System (HYPASS)

In [34], Arthurs et al proposed an extension of the FOX project, as this system used both high-speed tuneable lasers and tuneable receivers. The goal of the new network was to propose a packet switch that is capable of several hundreds of Gb/s throughput. The HYPASS architecture is shown in Figure 2.6.

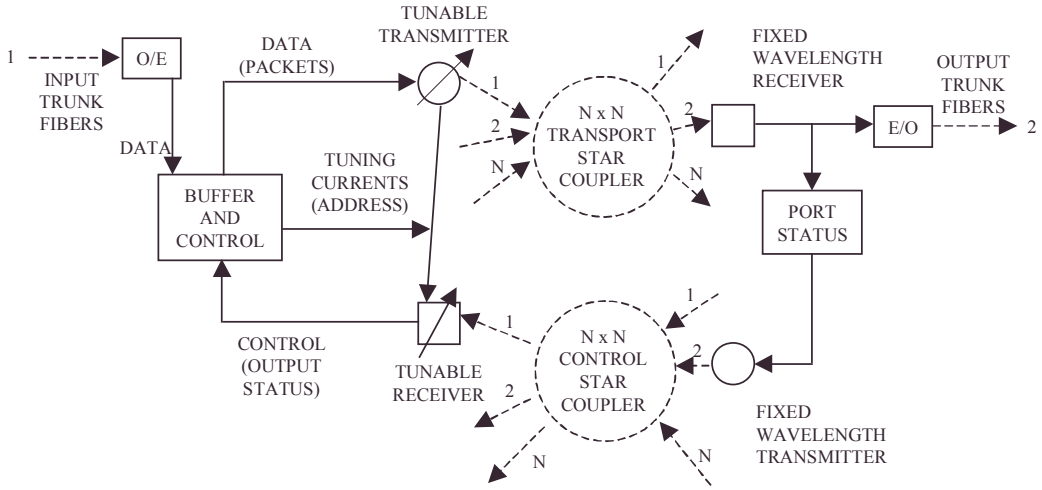


Figure 2.6. HYPASS architecture [34]

It is a hybrid electronic/optical switch because it uses electronics to do the signal processing and optics to do the signal transport. There are two $N \times N$ star couplers in the centre of the switch, one is dedicated to the transport of the information data from the inputs to the outputs and the other is for carrying the output-port status information back to the input as part of the control algorithm. In the transport path, the tuneable laser is tuned to address the desired output port. In the control path, the

input-port tuneable receiver is tuned to listen to a polling signal from the desired output port. The system can therefore be classified as TT-FR for information data and FT-TR for control. The control channel is mainly used to synchronise the transmitting and receiving nodes and to avoid collisions by using a tree-polling algorithm as detailed in [34].

The novelty of this architecture was the use of both high-speed tuneable transmitters and receivers. Implementations of suitable tuneable receivers (with tuning time of less than 10 ns) were proposed in [35], and [36]. Kanabar et al [35] used a permanently tuned array of receivers, one for each wavelength on the system, with the main drawback being that there must be as many receivers as there are wavelengths in the network. Kobrinski et al [36] proposed the use of a DFB laser-diode amplifier and demonstrated tuning times of 1-10 ns and up to eight wavelength channels. The principal limitation was the small tuning range (0.6 nm).

Finally, Goodman [37] proposed a modified architecture based on HYPASS where totally electronic contention resolution has been considered. This was known as the BHYPASS demonstrator.

2.3.1.2.2 STAR-TRACK

A network called STAR-TRACK was proposed by Lee et al [38] in order to take advantage of the inherent broadcasting capability of the optical star to achieve multicasting. The network uses a control electronic-ring-reservation scheme to handle

collisions that may occur at a destination node as those were designed as FT-TR systems. The architecture is shown in Figure 2.7.

A token, which consists of a sub-token for each of the output ports, is generated and passed through each of the input ports. A reservation scheme allows input ports to reserve one or more output ports (to realise selective multicast) for the next packet slot. This cycle takes place during the transmission cycle for the previous packet slot.

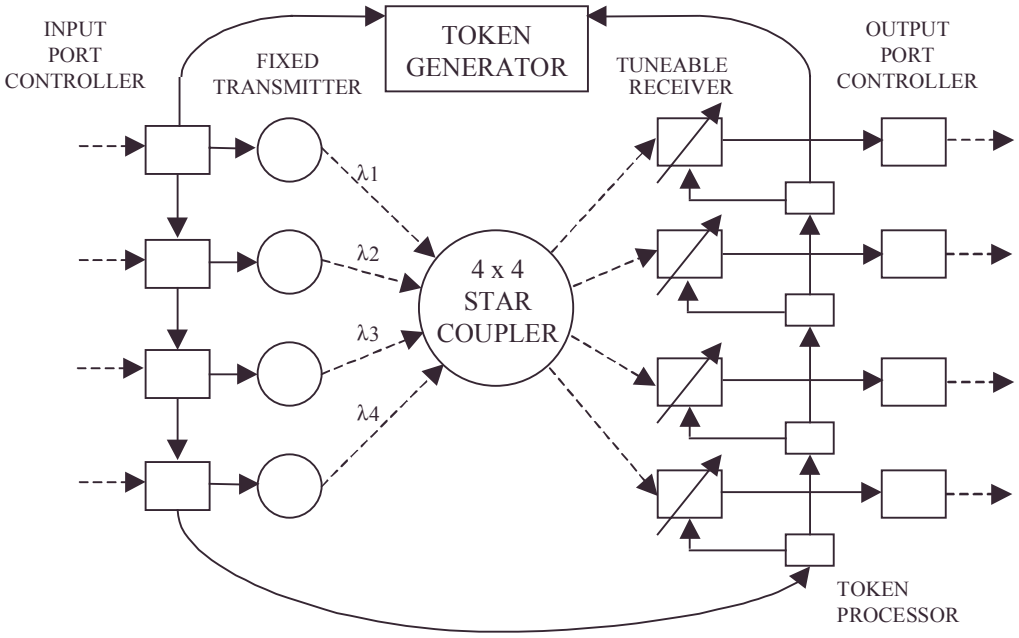


Figure 2.7. STAR-TRACK network architecture [38]

In addition to the selective multicast feature, STAR-TRACK allows a straightforward implementation of prioritised and reserved traffic. However, a significant factor which limits the performance of such a switch is the time it takes to process the token through the entire ring. This is quite different from the HYPASS network in which the input ports are processed in parallel, thus allowing better performance and scalability.

2.3.1.2.3 RAINBOW

The IBM Rainbow project [39-41] used a 32×32 star coupler with one fixed wavelength laser and one tuneable receiver at each node (FT-TR). It was designed to produce a circuit-switched metropolitan area network with a diameter up to 50 km. The transmission rate for each node was 200 Mb/s. A typical architecture, comparable to the LAMBDANET demonstrator, was considered and built but an interesting centralised alternative was also proposed. Indeed, in this version (see Figure 2.8), the lasers and tuneable optical filters are placed at the hub with the star coupler and a central controller is used to manage the control signalling.

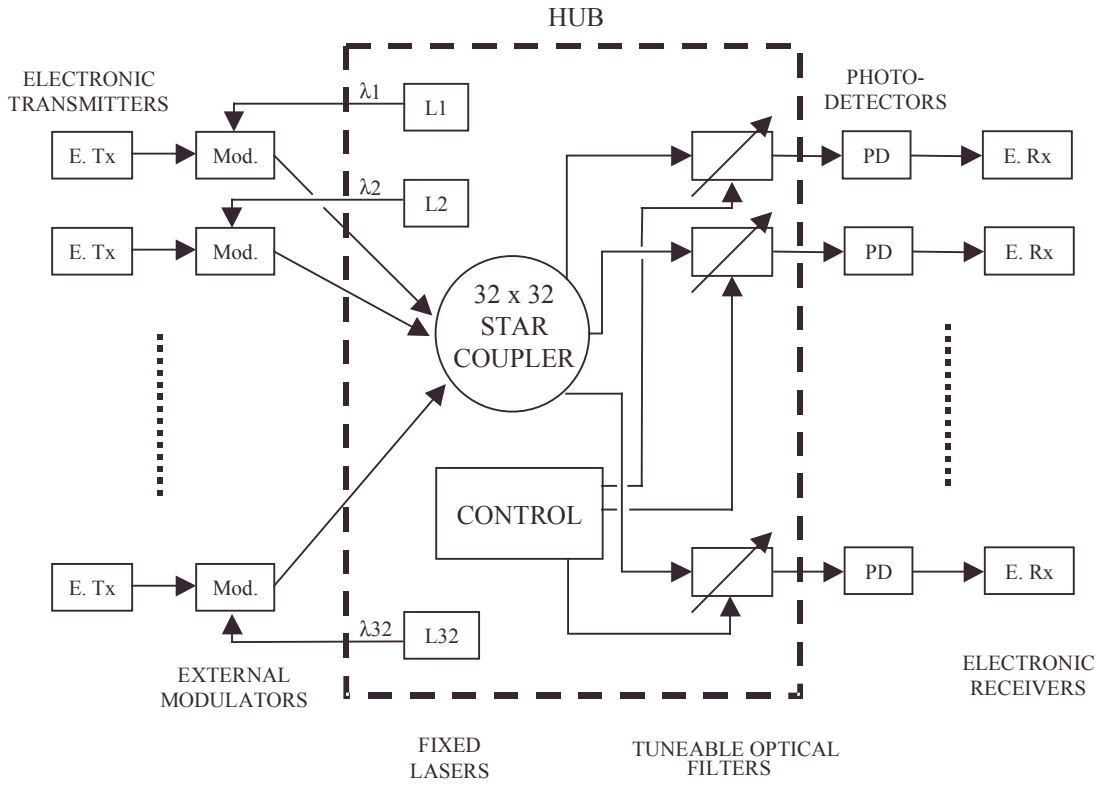


Figure 2.8. Centralised RAINBOW architecture [39]

In [39], the authors identified several advantages that were to be gained from the centralisation. All lasers were in a common temperature environment and each could be more easily controlled to avoid channel drifting. More importantly, lasers could be arrayed in a common semiconductor device in order to reduce costs and further simplify the stabilisation problem. Moreover, the design of each end station was greatly simplified as they were all identical and were not wavelength dependent. However, the main disadvantage of this implementation is that the hub becomes a single point of failure as it is no longer a passive unpowered star coupler.

The separate control protocol was used to co-ordinate the tuning of the receivers to select traffic from designated transmitters. In the first prototype, an in-band receiver polling method was implemented because of its simplicity and because it did not require a separate control channel. Idle receivers scanned all channels for setup requests directed to themselves. The setup request was a flag containing the destination receiver's ID. At the same time, the transmitting station tuned its receiver to the channel of the desired destination's transmitter to wait for a circuit setup acknowledgement.

A follow-up report [40] provided further information and the lessons learned from developing the first RAINBOW prototype were presented by Green [41]. A second prototype called RAINBOW-II [42] was built in the mid-90's. It involved a collaboration between IBM and the Los Alamos National Laboratory and provided higher connectivity between nodes (1 Gb/s) and improved control protocols capable of handling gigabit-per-second transmission rates.

2.3.1.2.4 Protocols based on pretransmission co-ordination

All the protocols based on pretransmission co-ordination employ one or more control channels, which can sometimes be embedded on the data channels. The control scheme is implemented to arbitrate the nodes access to the data channels. In this section, we will focus on the major protocols [43-48] that were proposed to achieve such functionality.

Habbab [43] proposed and evaluated different partial random access protocols which were defined as control-channel-ALOHA. These are similar to the ALOHA protocols presented in the previous section [31] but they are based on TT-TR systems. Moreover, protocols in [43] employ a separate control channel which is used by nodes to notify their transmissions to the intended destinations nodes. A node transmits a control packet over the control channel after which it immediately transmits the data packet on the data channel which was specified in its control packet. The control packet also contains the source and destination address of the nodes involved in the communication. A number of random access protocols were studied by using three different methods (ALOHA, slotted-ALOHA, and CSMA) to access both the control and data channels. However, Dowd [31] outlined the major limitations of these protocols (i.e. high probability of receiver collisions) and demonstrated superior performance by simpler protocols that did not require a separate control channel.

Nevertheless, Mehravari [44] extended the work in [43] to obtain improved protocols and performance predictions. Specifically, a node was required to delay its access to a data channel until after it learned that its transmission on the control channel had been

successful. As a result, better throughput performance was achieved and the number of collisions were reduced. He also proposed the use of higher level layers (e.g. transport layer) to detect lost packets (due to collisions) and initiate retransmissions.

Furthermore, Sudhakar et al [45] proposed two sets of slotted ALOHA protocols and one set of reservation ALOHA protocols. The first two sets of results were mainly variations of the schemes proposed in [43] and [44], but the major contribution of this work was the introduction of reservation mechanisms. Indeed, a node could reserve the same data channel for a subsequent number of cycles until its use of the channel was completed. This was a very interesting feature to accommodate circuit-switched traffic or traffic with long holding times.

In the previous protocols, the main limitation was the difficulty in detecting receiver collisions. However, even for such simple systems, Jia and Mukherjee [46] demonstrated how receiver collisions could be avoided by adding some intelligence to the receivers. Their receiver collision avoidance (RCA) protocol implemented a few basic mechanisms to overcome the receiver collision problem. By using a reception scheduling queue (RSQ), a node activity list (NAL), an asynchronous transfer on data channel (ATDC) mechanism and three receiver collision detection (RCD) schemes, each node could avoid receiver collisions, and the maximum throughput of the network was shown to be about 36 % of the total network transmission capacity. This was higher than the other schemes which used a similar structure (i.e. TT-TR with control channel). However, the main limitation of the work in [46] was that the packet delay could fluctuate over a relatively wide range under heavy traffic loading.

Chen et al [47] proposed a different implementation of the nodes structure. The dynamic time-wavelength division multiple access (DT-WDMA) requires that each node be equipped with two transmitters and receivers. One transmitter and one receiver at each node are always tuned to the control channel whose access is TDM based. With DT-WDMA, each node has exclusive transmission rights on a data channel to which its other transmitter is always tuned. The second receiver at each node is tuneable over the entire wavelength range. This architecture can be classified as FT^2-FRTR system. All channels are slotted and centrally synchronised. A slot on the control channel consists of N mini-slots, one for each of the N nodes. Each mini-slot contains a source address field, a destination address field, and an additional priority field. By monitoring the control channel over a slot, a node determines if it is to receive any data over the following data slot. If there are more than one node transmitting data to it over the next data slot, the receiver checks the priority field of the corresponding mini-slots and select the one with the highest priority. To receive the data packet, the node simply tunes its receiver to the sender's dedicated transmission wavelength. Even in the case of a receiver collision, exactly one transmission will always be successful. The main advantage of this architecture is its increased throughput (63 % of the total network capacity) capacity. However, the main limitation is its scalability as it requires as many wavelengths as nodes in the network.

Furthermore, Chlamtac and Fumagalli [48] extended the previous work by adding optical delay lines to the nodes receiver. These were used to buffer colliding packets which would have been lost with DT-WDMA. If a node was not going to receive packets from any source over the next data slot, then it could read a previously

buffered packet. Simulation results indicated that with a delay line capacity of 10 packets, the throughput could be raised to 95% of the total network transmission capability.

2.3.2 Multi-hop broadcast-and-select networks

Multi-hop systems based on a physical star topology have been widely studied and various architectures and protocols have been proposed. In contrast, very few demonstrators were built and most of the work remains theoretical. In this section, we present a number of theoretical architectures [49-52] which were proposed as suitable candidates for multi-hop networks. Finally, the STARNET [54-56] demonstrator is described and its main features identified.

In multi-hop networks, data is broadcast to all nodes, but electronic switching at intermediate nodes provides wavelength conversion on the path from the source to the destination because not all nodes can receive all wavelengths. A communication stream from a source to a destination has to hop through one or more intermediate nodes. In general, it is unlikely that there will be a direct path between every node pair. Moreover, the channels to which a node transceivers are to be tuned are relatively static, and this assignment is not expected to change except when a new global assignment is required, usually in order to optimise the topology.

The main advantage of using multi-hop networks when compared with single-hop architectures is that no protocol is required to arbitrate access to the network. The fairly static virtual topology is chosen (or computed) in order to avoid both access and

receiver collisions. In most of the presented architectures, each node is equipped with two fixed-tuned receivers and two fixed-tuned transmitters. Latter in this section, these networks will be noted as (2×2). They can be classified as FT^2-FR^2 systems, but one must keep in mind that in this specific case, the prefix F can either represent a fixed-tuned transceiver or a fixed one. In the first case, the network is re-configurable while in the second case the topology, once designed, is fixed.

An example of a fairly simple multi-hop network is shown in Figure 2.9. The physical topology is a star while the virtual topology is a (2×2) torus. In this example, node 1 can directly communicate with nodes 2 and 3, but in order to reach node 4 the information stream must multi-hop either through node 2 or node 3.

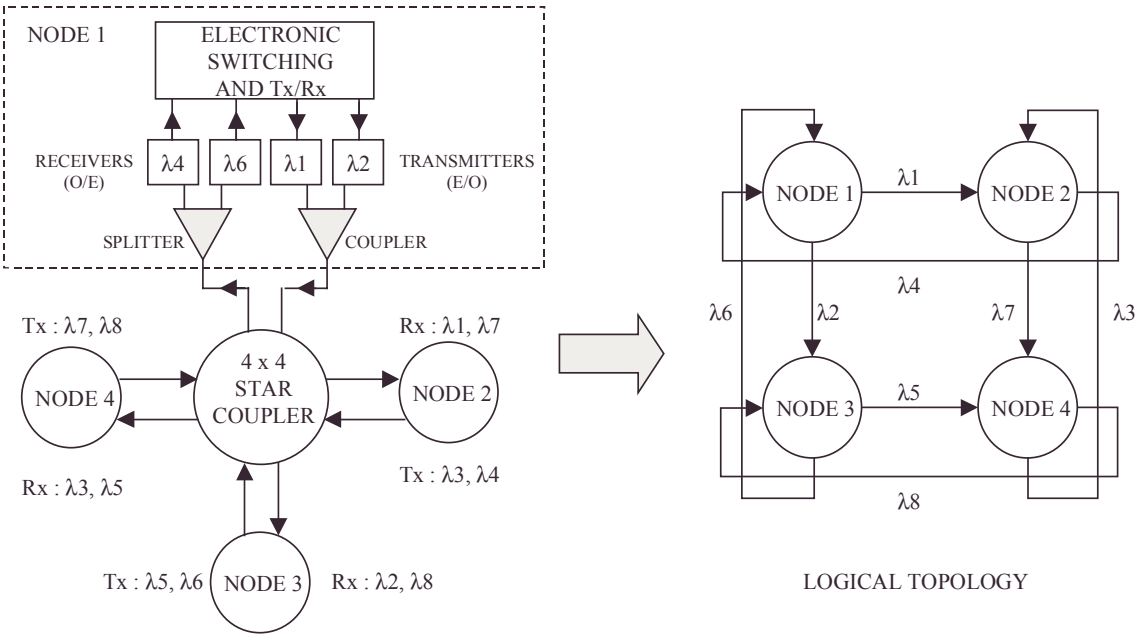


Figure 2.9. A (2×2) multi-hop network: physical and logical topologies.

Furthermore, in designing a multi-hop system, there are two main issues that must be considered. First, the virtual topology must be optimal in the sense that the average distance (hop) between nodes must be minimal, or the average packet delay must be small. Second, the processing complexity must be minimised and simple switching mechanisms must be employed because the very high-speed environment is extremely time sensitive. Finally, another important issue is whether to employ dedicated or shared channels. In the first case, each virtual link employs a dedicated wavelength channel when in the second, two or more virtual links can share the same channel in order to improve the channel utilisation and reduce the required number of wavelengths.

In the next paragraphs we will present different multi-hop network designs which proposed and evaluated diverse topological optimisation approaches.

The torus presented in Figure 2.9 was one the first architectures considered in designing multi-hop networks. Maxemchuk [49] proposed and evaluated such a structure in a metropolitan area network, the Manhattan Street Network (MSN). The main advantage of this architecture was its modularity but it was rapidly abandoned due to much lower performance compared to the following proposals.

The next approach in designing multi-hop network was the ShuffleNet [50]. A (p, k) ShuffleNet can be constructed out of $N = kp^k$ nodes which are arranged in k columns of pk nodes each, and the k^{th} column is wrapped around to the first in a cylinder fashion. The value p is defined as the degree of the system (i.e. the number of connections each node has with the adjacent nodes). A (2, 2) ShuffleNet is shown in

Figure 2.10. It has $N = 2 \times 2^2 = 8$ nodes. In this case, the i^{th} node transmits on wavelengths $\lambda_{(2i-1)}$ and $\lambda_{(2i)}$ and receives on wavelengths $\lambda_{[(N+i) \bmod 12]}$ and $\lambda_{[(N+i) \bmod 12]+4}$. Also, the average number of hops between any two randomly selected nodes is given by Hluchyi et al [50] by

$$\bar{h} = \frac{kp^k(p-1)(3k-1) - 2k(p^k - 1)}{2(p-1)(kp^k - 1)}.$$

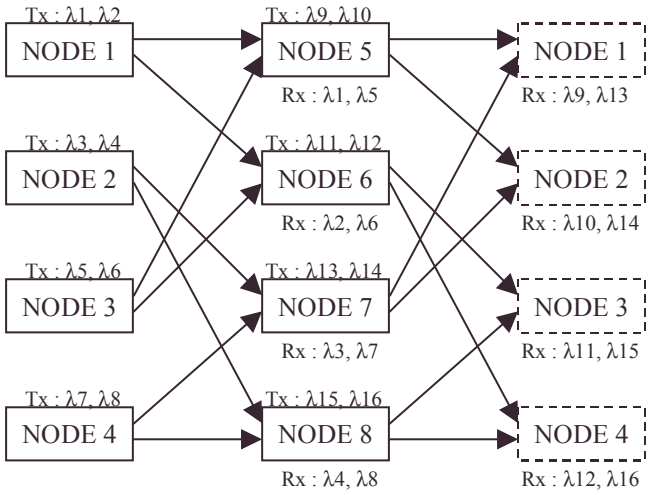


Figure 2.10. A (2, 2) ShuffleNet [50]

In the case of a (2, 2) ShuffleNet, the average number of hops is then $\bar{h} = 2$. The main limitation of the presented ShuffleNet architecture is that it requires $w = 2N$ wavelengths and for large values of N the average number of hops $\bar{h}$ tends to be quite considerable. Also, the diameter (the maximum hop distance) of the ShuffleNet (which is equal to $2k$-1) can be very large.

The shuffle exchange structure has been widely studied and one of the proposed architectures was the de Bruijn graph [51]. In a (Δ, D) graph ($\Delta \geq 2, D \geq 1$), the

degree of the network is given by Δ , the diameter is equal to D and the number of nodes is $N = \Delta^D$. The main advantage of the de Bruijn graph is that the average number of hops is smaller than when compared with a ShuffleNet architecture with a similar number of nodes. An upper bound on $\bar{h}$ is obtained by Sivarajan et al [51] and is given by

$$\bar{h} \leq D\frac{N}{N-1} - \frac{1}{\Delta - 1}.$$

On the other hand, the maximum throughput supportable by a de Bruijn graph is lower than that supportable by an equivalent ShuffleNet structure (same number of nodes and same nodal degree).

Iness et al [52] proposed the GEMNET (GEneralised shuffle-exchange Multi-hop NETwork) architecture as a generalisation of the shuffle exchange concept, including the ShuffleNet and the de Bruijn graphs. These are classified as (*K, M, P*) Gemnets, where K is the number of columns, M the number of rows, and P the degree of the system. The number of nodes is given by $N = K \times M$. It is demonstrated in [52] that a (*K, M, P*) Gemnet reduces to a ShuffleNet when $M = P^K$, and to a de Bruijn graph of diameter D when $M = P^D$ and $K = 1$ (with $D \geq 2$). An attractive scalability feature of GEMNET is the use of shared channels as evaluated in [53]. In a shared channel Gemnet (SC-Gemnet), the number of wavelength w is less than the number of nodes N, and each node has only one fixed-tuned transmitter and one fixed-tuned receiver. Channels are shared using time-division multiplexing (TDM). Figure 2.11 shows the logical topology of a (2, 4, 2) Gemnet and TDM frame used in that case. In this example, $M = P^K$, and therefore the resulting architecture is a shared-channel ShuffleNet. There are two main disadvantages of channel sharing. First, the

throughput per node is reduced when w is much smaller than N and second, the queuing delay introduced by the TDM scheme can reduce the networking performance.

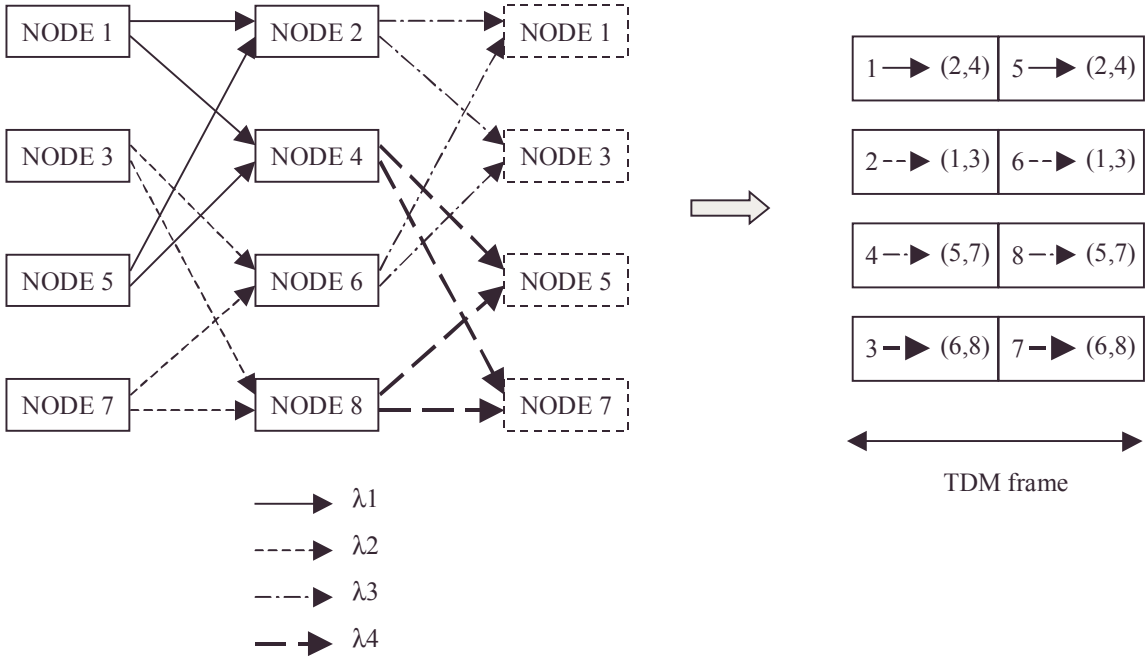


Figure 2.11. A (2, 4, 2) Gemnet with channel sharing [8]

2.3.2.1 STARNET

STARNET [54-55] is a WDM local area network developed by the Optical Communications Research Laboratory at Stanford University. STARNET provides two logical sub-networks which can simultaneously transport circuit- and packet-switched traffic on the same physical architecture. In this review, we will only briefly present the data switched sub-network which is designed as a multi-hop structure. Indeed, in STARNET, the physical architecture is a star but the logical topology is organised as a ring. Each node is equipped with a tuneable receiver permanently tuned

to the fixed-tuned transmitter of the previous node (of the resulting unidirectional logical ring). This implementation is shown in Figure 2.12.

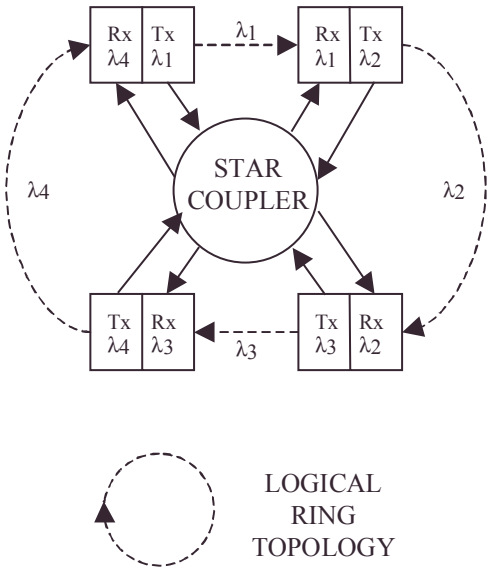


Figure 2.12. The data switched STARNET architecture [8]

When a node wants to send a packet, it simply transmits the packet to the downstream node which forwards it to the following node until the destination is reached. At each node, the optical data stream is converted into electronic form in order for the node to process the packet (i.e. mainly to check the destination address field). If the packet needs to be forwarded, it is optically re-sent to the following node of the virtual ring.

In an improved version called STARNET II [56], multiple subcarrier multiplexing (MSCM) was used to implement the superposition of the two sub-networks. In the first STARNET project, a combination of differential phase shift keying (DPSK) and amplitude shift keying (ASK) were used in order to multiplex the circuit- and packet-

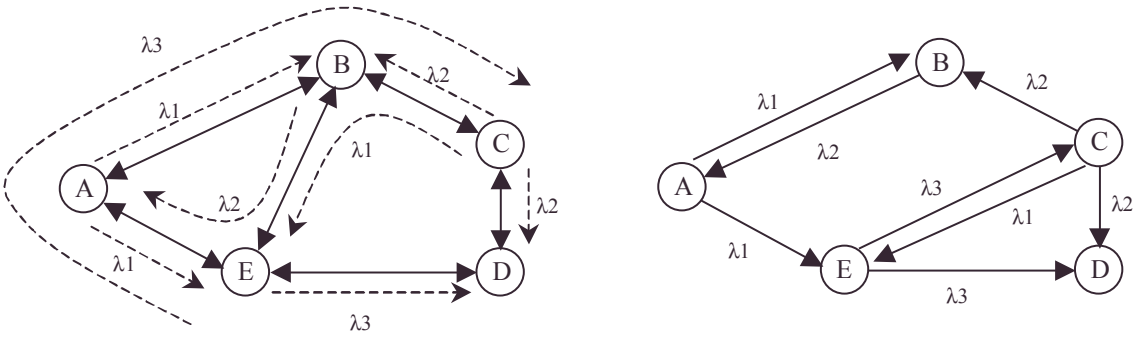
switched sub-networks. The performance of these two schemes were presented and analysed in [56].

2.4 Wavelength routing networks

While broadcast-and-select architectures are suitable for local and metropolitan area networks, they are not suitable for wide area networks [5, 57]. Their main limitation is inherent to their broadcasting mode of functioning. Indeed, because each transmission is broadcast to all other nodes in the network, most of the transmitted power is wasted on receivers that do not use it and, as the number of nodes increases, each station receives a smaller fraction of the transmitted power. Moreover, additional scalability must be provided by using each wavelength at many places in the network at the same time. A wavelength routing network [58] realises the latter function, and also avoids wastage of transmit power by channelling the energy transmitted by each node along a restricted route to the receiver, instead of letting it spread out over the entire network as with broadcast-and-select networks. Therefore, at each intermediate node between the end nodes, light coming in on one incoming port at a given wavelength is routed out of one and only one outgoing port by a wavelength router. This routing process may involve a change in the wavelength of the optical signal and more importantly, in all-optical networks [59], only optical switching components are to be used and electronic conversion of the signal must be strictly avoided.

Chlamtac et al [60] introduced the concept of *lightpaths*. A lightpath is an optical path, capable of carrying both packet-switched and circuit-switched types of data traffic, which is established between two nodes in the network and is created by the

allocation of the same wavelength throughout the path. In addition, the wavelengths and paths assigned must be such that no two paths that share an edge are assigned the same wavelength. The main advantage of using lightpaths is that they require no processing or buffering at intermediate nodes and, in the case of all-optical networks, no E/O conversions. Furthermore, a virtual topology must be implemented on top of the underlying physical topology. There are a number of issues that must be addressed when designing such a virtual topology. These will be described in a further section where the routing and wavelength assignment (RWA) problem will be discussed. A typical wavelength routing network is shown in Figure 2.13.



(a) physical topology and wavelenght assignment

(b) virtual topology

Figure 2.13. A wavelength routing network [4]

The network has five nodes and three wavelengths and bi-directional links are employed between them. Note that despite the fact that nodes C and E are not directly connected by a fibre, two lightpaths have been established through nodes A and B to interconnect them. In contrast, nodes B and E have a bi-directional fibre link, but no lightpath was established between them. However, information could possibly be routed through nodes A and C. For example, if node E wants to transmit to node B, it

uses the wavelength λ3 to reach node C, which will use the wavelength λ2 to forward the information stream to node B.

In the next section, we will present the first implementations and designs of wavelength routing networks and in the mean time, describe and compare the fixed and dynamic wavelength routing approaches. The main design issues that need to be considered when designing wavelength routing networks will be identified in the following section. Finally, a number of demonstrators will be briefly presented and their main features identified.

2.4.1 Fixed and dynamic routing architectures

Wavelength routing was first conceived as a fixed-routing architecture [58]. In this case, once designed, the network logical topology is fixed and static routing is performed. Indeed, the path taken by a signal is statically determined by the wavelength of the signal and the port through which it enters the network. If we consider a network with N inputs and N outputs, one might expect N^2 wavelengths would be required to form a complete interconnection. In fact, with any arbitrary topology, Barry and Humplet [61] demonstrated that a minimum of only $\sqrt{N}$ wavelengths is required to support one connection per node without any blocking. However, for a fully non-interfering interconnected network, only N wavelengths are required. Figure 2.14 shows one possible arrangement of fully connected 4×4 network. Its associated wavelength assignment table is shown in Table 2.5. For example, the wavelength to go from input port $S1$ to output port $R3$ is λ_2. Moreover, no output port can receive any given wavelength from more than one

input. This can be extended to any size network with N wavelengths but it does require N^2 interconnection fibres.

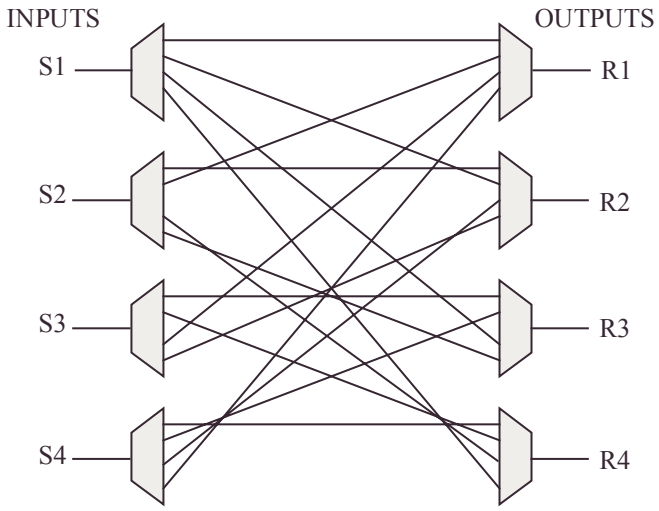


Figure 2.14. Example of 4×4 fully interconnected network [1]

INPUT	OUTPUT			
	R1	R2	R3	R4
S1	λ0	λ1	λ2	λ3
S2	λ1	λ0	λ3	λ2
S3	λ2	λ3	λ0	λ1
S4	λ3	λ2	λ1	λ0

Table 2.5. Associated wavelength assignment table [1]

However, it may be useful to modify the routing pattern dynamically in order to respond to changing network traffic or failures in the physical topology (e.g. fibre cut, node failure). In the first dynamic routing networks, this was done by introducing optical switches in the routing nodes as shown in Figure 2.15. This was implemented

in the first demonstration of wavelength routing networks by British Telecom Laboratories [62]. Figure 2.15 shows the architecture of a routing node which is equipped with two 3×3 optical switches. In that case, a data stream coming through any input port at any wavelength can be switched to any output port on the same wavelength. A wavelength assignment table must be defined for each optical switch but, in contrast with the previous example, it is not fixed and can be dynamically modified.

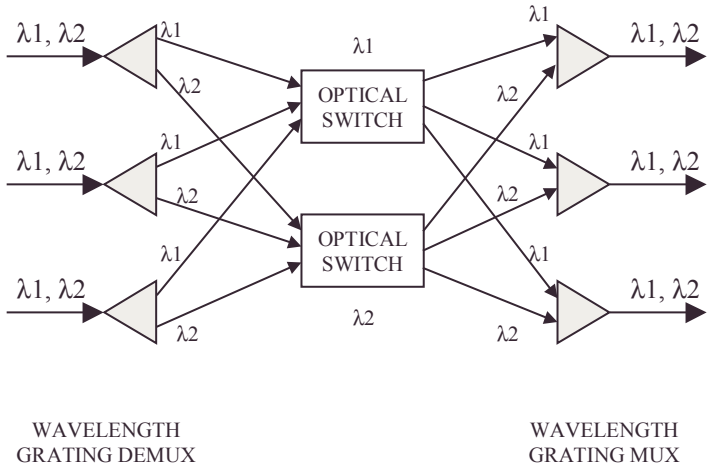


Figure 2.15. Architecture of an optical routing node [57]

As stated earlier, a dynamic wavelength routing network has the advantage of being re-configurable when compared with a fixed wavelength routing network. However, if the virtual topology is be changed dynamically, the algorithm used to re-assign wavelengths and lightpaths is of paramount importance. This is known as the routing and wavelength assignment problem.

2.4.2 Routing and wavelength assignment (RWA)

As identified by Chlamtac [60], two central issues must be addressed when designing the virtual topology of a WDM routing network. First, it is necessary to establish lightpaths efficiently in terms of the total number of wavelengths required. There is indeed a finite number of wavelengths available and they must be carefully used. Second, the requirement for establishing a lightpath using the same wavelength throughout its route, introduces a potential bandwidth loss when compared to a lightpath establishment in which the continuity constraint is not imposed. This may either induce an increase in the number of wavelengths required, or induce an increased lightpath blocking probability. Moreover, Ramaswami [59] noted that, if the use of dynamic wavelength conversion is allowed at intermediate nodes, the RWA problem becomes equivalent to the extensively studied routing in the circuit-switched telephone network [63].

In fact, when designing the logical topology, there are many other constraints that can also be considered. As a result, the routing and wavelength assignment (RWA) problem, or variants of it, has been widely studied and many RWA algorithms have been proposed and evaluated [59-61, 64-70]. Furthermore, these proposals can be classified depending on the main constraint on which they focus their optimisation study.

Ramaswami et al [59] and Chlamtac et al [60] concentrated on minimising the number of wavelengths needed and also analysed the blocking probability of their proposed routing protocols. In a latter work, Ramaswami [64] proposed and compared

different algorithms to minimise the network congestion and reduce the average packet delay. Birman [65] analysed the blocking probability of a specific type of fixed wavelength routing networks and gave theoretical and simulation results. Kovačević [66] and Barry [67] studied the impact of using wavelength converters at the network nodes and they compared the network performance with previous results. One of the main constraint when dynamically re-routing a virtual topology is to minimise the network disruption incurred by the reconfiguration. Lee [68] and Banerjee [69] concentrated on this issue and proposed algorithms to reduce the disruption period when optimising the topology to changing networking conditions. Finally, Wauters and Demeester [70] evaluated and compared the performance of different algorithms in a review of existing wavelength routing protocols.

2.4.3 Wavelength routing demonstrators

There has been a large number of testbeds and demonstrators that have been designed to evaluate the feasibility of wavelength routing networks. One of the most significant demonstrators is the Multiwavelength Optical Networking (MONET) project [71, 72], a five year program funded by the Defense Advanced Research Project Agency (DARPA) and undertaken by AT&T, Bellcore, Lucent Technologies and other partners. The MONET project envisioned a transparent, re-configurable and scalable optical layer which could provide a flexible infrastructure for building networks at higher layers, such as SONET, ATM, and IP. The major accomplishments of the MONET project were reported in [72]. These include a detailed review of the prototype network implementation, and experimental results of the network operation.

The MONET program ended in November 1999 and has accomplished much more than was originally proposed [71].

The European RACE program developed one of the first wavelength routing demonstrators, the Multiwavelength Transport Network (MTWN) [73]. The project was a more sophisticated outgrowth of early work by BT on the London fibre network, providing for interconnection between a wavelength-routed optical layer and a conventional electronic layer. One of the major contributions of the project was the demonstration of optical networks transparency and the ability to carry multiple data transmission formats.

Finally, the Wavelength Switched Packet Network (WASPNET) [74] and the Optical Packet Experimental Routing Architecture (OPERA) [75] projects are two experimental networks which demonstrated the use of subcarrier multiplexing (SCM) to encode packet headers in order to provide a reliable and efficient addressing technique. In both cases, an original switch architecture which used recent developments and progresses in optical components was proposed and described.

2.5 Ring technology and systems in electronic local area networks

Ring broadcast networks have been widely studied since the early 1970's and they have been widely used in local area networks. One of the main attractive feature of ring networks is that restoration and protection schemes (for link and node failures) can be implemented much easier than with other broadcast topologies. Another characteristic of a ring network is that it is not really a broadcast medium, but rather a

collection of individual point-to-point links that happen to form a cycle. Therefore, communications within a single ring are unidirectional and the data travels from one node to another either in the clockwise or in the anti-clockwise direction. In such a network, messages are transmitted in packets or frames, each of which contains the destination address of the intended receiver. Because of the ring inner topological structure a packet, once transmitted onto the ring, must be removed otherwise any further communications would immediately become impossible when the ring fills up.

There are two points in time when a packet can be removed from the ring. It can be removed from the ring either by the destination node upon its arrival (destination stripping) or by the source node upon its return after circulating the ring (source stripping). The main advantage of the latter case is that a positive message acknowledgement (ACK) can be performed "on the fly" by the destination node, e.g. by simply setting an acknowledgement bit at the end of the message to indicate the success of the reception. The main disadvantage of this scheme is that bandwidth is wasted in carrying the message back from the destination node to the source node. However, by using a destination stripping scheme, the bandwidth usage can be greatly increased. Positive acknowledgement "on the fly" cannot be made, but a separate, short message may be sent by the destination node to the sender to acknowledge the correct reception of the data. On the other hand, destination stripping schemes often require more complex hardware implementations.

Moreover, as in any medium shared network, a protocol must arbitrate the nodes access to the ring. Such a scheme is known as the medium access control (MAC)

protocol. Several MAC protocols for ring networks have been developed and they can be classified into three basic types: token passing, empty slot, and register insertion.

In the next three sections, we will discuss the main features of the different types of MAC protocols and will present some of the architectures that were proposed for ring networks. In the following section, the main design issues of implementing ring networks will be introduced and the main limitations of the presented protocols will be outlined.

2.5.1 Empty slot protocols and experimental networks

The empty slot system or slotted ring was originally suggested by Pierce [76] in 1972. The corresponding MAC protocol is also known as slotted-ALOHA [77]. In this type of rings, a small number of skeleton packets called slots circulate continuously round the ring. Their number is fixed and depends on the length of the slot (in bits), the total length of the ring (in meters), the message propagation speed in the ring medium (in meter/s) and the data rate of the network (in bit/s).

One node, known as the monitor, is responsible for initialising and maintaining the ring framing. A station with data to transmit partitions it into fixed-length frames and then simply waits until an empty slot passes on the ring. The data packet is then transferred to the slot data field, and a *full/empty* flag in the slot is set to *full* to indicate to other nodes that the slot is in use. In the Pierce ring [76], destination stripping was used and the destination node was therefore responsible for marking a slot *empty* after it had been correctly received. In the Cambridge experimental ring

[78], source stripping was used and thus the source node marked a slot *empty* after it had completed an entire loop. In both networks, slots which were not removed after a number of revolutions (e.g. in the case of a node failure) were emptied by the ring monitor. Figure 2.16 shows a schematic diagram of a slotted ring with eight slots and four nodes.

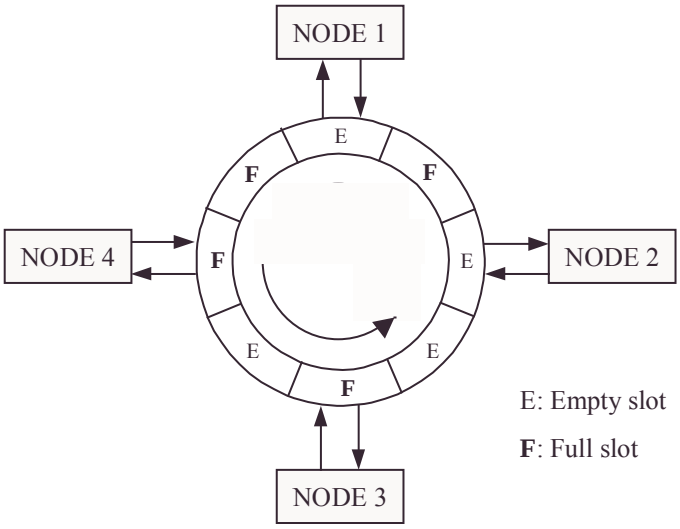


Figure 2.16. An slotted ring architecture

The main advantage of the slotted ring is that more than one message can be carried by the network at any given time, and hence simultaneous transmissions are possible. The main drawback is that the slots size is fixed and must be carefully chosen for the network to be efficient.

2.5.2 Token passing protocols and experimental networks

The second basic methods for controlling access to a ring is token passing. In this scheme, a special bit pattern, called the *token,* is passed from node to node around the ring. When a station wants to transmit a packet, it is required to seize the token and

(logically) remove it from the ring before transmitting. Once the transmission is completed, the token is (logically) released onto the ring and other nodes may be able to transmit. Because there is only one token, only one station can transmit at a given instant.

There are a number of variations of the basic token passing protocol described above in addition to the variation in source or destination stripping. The first one is concerned with the number of packets a node can transmit when it holds the token. The second variation is determined by the pattern and location of the control token, which may be indicated in the message being sent (within the frame) or it may be a separate message that has no information field. The third variation depends on the point in time when the control token is released by the sending node to the next downstream node (i.e. immediately after message transmission or after message removal).

The first implementation of a token passing protocol is the Newhall [79] ring. In this scheme, the token is outside the message frame and is released by the sender immediately after the message transmission. The message is later removed by the sender upon its return (source stripping). A similar mechanism was used in the Prime's Ringnet [80] which was the first commercially available local computer network using a ring.

In the beginning of the 1980's, IBM developed a ring [81] which would latter become the Token Ring IEEE 802.5 standard [82]. In this protocol, the control token is within the message header and it is released by the sender when the message header returns.

The rest of the message is then removed by the sender. When a node has a message to send, it waits for the token to come by, changes it to a busy token on the fly (by altering one the TK bit in the header), and transmits the message. This operation is shown in Figure 2.17. To detect a persistent busy situation, the monitor sets a monitor bit to 1 on any passing busy token. If it detects a busy token with the bit already set, it knows that the transmitting station has failed to delete its frame and the monitor can take the necessary action by changing the busy token to a free token. To detect a lost-token condition, a time-out is initiated greater than the time required for the longest frame to circulate the ring. If no token is detected in that time, the monitor deletes any residual data in the ring and issues a new free token.

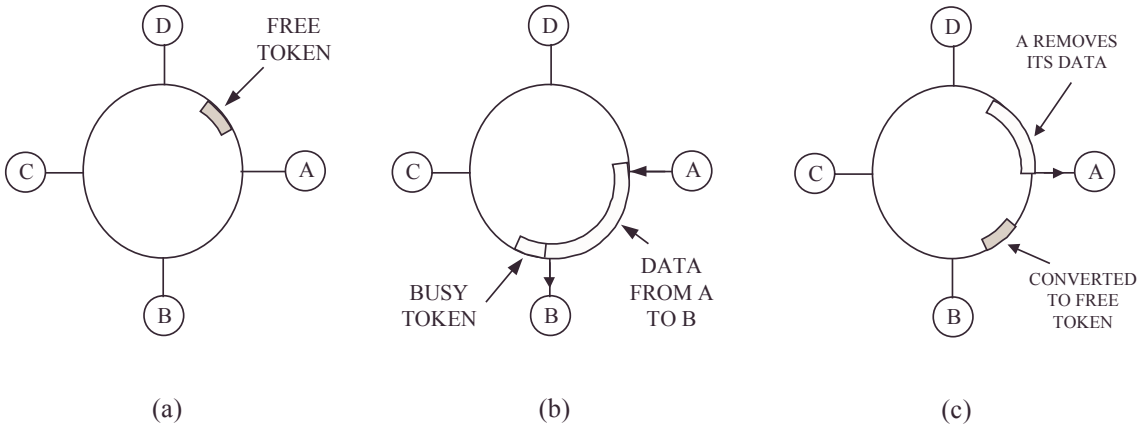


Figure 2.17. The IBM token ring mechanism

The principal advantage of a token ring is that traffic can be regulated by allowing stations to transmit different amounts of data when they receive the token, or by setting priorities so that higher-priority stations have first claim of the circulating

token. The main disadvantage is that the token ring architecture requires carefully designed token management techniques.

2.5.3 Register insertion protocols and experimental networks

The third method for controlling access to the ring is called register insertion or buffer insertion. It operates in a similar manner to the slotted ring. When a node has information to transmit, it loads this into a shift register. To transmit the information, the register is then switched in series with the ring connections at a node so that it forms part of the data transmission path through the network. This is illustrated in Figure 2.18. The switching occurs whenever there is a convenient gap between other packets travelling round the network.

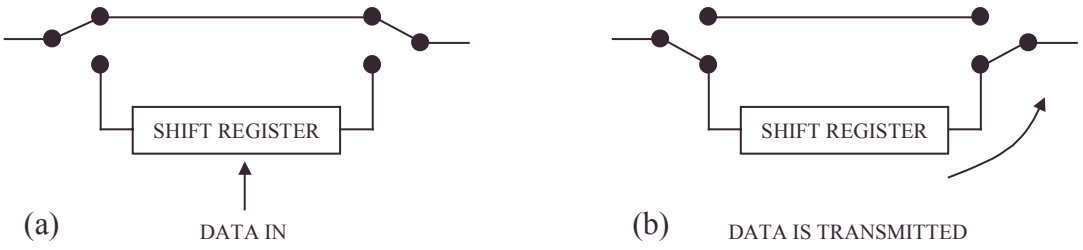


Figure 2.18. Register insertion: (a) loading – (b) transmission

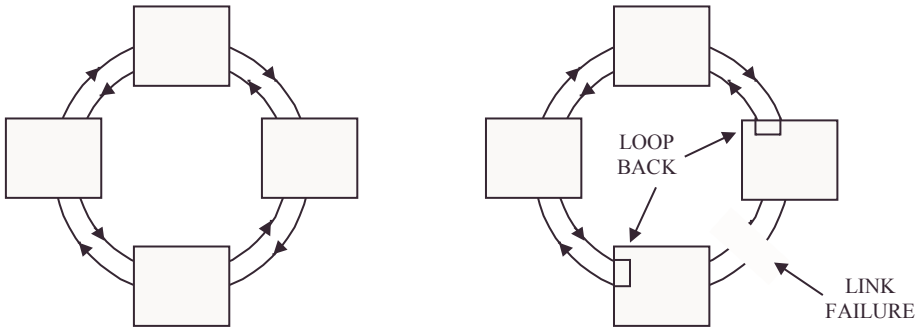
The register stays in series with the ring so that all data may be diverted through the register. When the transmitted message returns and is completely stored in the register, it is switched out of circuit and the message is thus removed from the ring.

This is, in principle, fairly simple, but it becomes much more complex in a practical system. One of the first proposals to use the register insertion technique was made by Hafner et al [83]. In fact, the Hafner ring uses two registers at each node as during the transmission phase, data sent from upstream nodes may reach a transmitting node. In that case, a receive shift register (RSR) is used to temporarily store the incoming messages. Immediately after its transmission completes, the node switches the RSR to the ring to relay its content.

An interesting property of this ring architecture is that it allows variable packet lengths to be transmitted. Furthermore, if the ring is idle, any node can have the entire bandwidth if it requires so. On the other hand, if the ring is busy, there might be insufficient empty space in the ring to allow a node to transmit its register content.

2.5.4 Ring design issues

As stated earlier, ring networks are very popular due to their resilience to link failures. This reliability feature has also been widely studied and it can be implemented with any of the above three types of ring networks. However, in the case of a link failure, different mechanisms must be implemented. The most popular technique involves the use of two counter-rotating rings, with all nodes being attached to both. When a link failure occurs, the bad section of the network is isolated from the network by an automatic loop-back procedure that occurs at the two nodes that are adjacent to the section. This mechanism is shown in Figure 2.19.



(a) NORMAL OPERATION (b) FAULT OPERATION

Figure 2.19. Recovery from link failure

Such a scheme was first implemented in a token passing network by Paulish in the Burroughs ring [84] at the Defense Communication Engineering Center in Virginia. It was also a feature of the PLANET (Private Local Area Network) system [85], an empty slot ring which was commercially available in the mid-1980's. A similar technique was used by Liu et al [86] in the Distributed Double-Loop Computer Network (DDLCN) at the Ohio State University. This was a variable-length register insertion ring where nodes could simultaneously use both rings for active transmission of messages.

In a comparative study of ring networks, Bux [87] simulated and evaluated the performance of both the token passing and slotted rings. In general, the token passing scheme performed considerably better than the slotted protocol. The poor performance of the slotted protocol was due to one main reason which limited the efficiency of this type of ring networks. Indeed, the main weakness of such a network was its extremely low bandwidth-delay product, which is defined as the carrying capacity (in bits) of the ring. Indeed, for a ring length of 1000 metres and a data rate

of 10 Mb/s, and if we consider a signal propagation speed of 2×10^8 m/s, the ring can only contain 50 bits at any instant. In Bux's study, the ring usually contained a single slot and moreover, the ratio of the header length to the slot length was extremely unfavourable (i.e. 0.4). It was also shown that, as the bit rate increased, the slotted ring performance became better and it also improved with higher number of slots, provided this was achieved without diminishing the length of the slots. The main conclusion is that a high bandwidth-delay product is required to achieve good performance in slotted rings. Also, the ratio of the header length to the slot length must be reduced to a minimum.

It is also worth mentioning that, despite the fact that token passing schemes perform better than slotted protocols, complex token management mechanisms must be implemented in token passing rings and it therefore requires a more costly hardware implementation.

2.6 WDM ring networks

Although most of the research interest in WDM architectures has been focused on broadcast-and-select star and wavelength routing mesh networks, there has been an increased attention focused on WDM ring networks in the past few years. There are, indeed, a number of advantages associated with ring networks. Lee et al [88] outlined the main advantages in a recent paper: simple routing policy, simple control and management of network resources, simple hardware system, and simple protection from network failures. Moreover, ring networks are predominant in the current metropolitan area network market, and WDM rings are expected to appear as the next

generation access networks in the MAN area [13]. As for any kind of WDM systems, rings can be designed as wavelength routing or broadcast-and-select networks. These two types of WDM rings will be respectively presented in the following two sections.

2.6.1 Wavelength routing ring networks

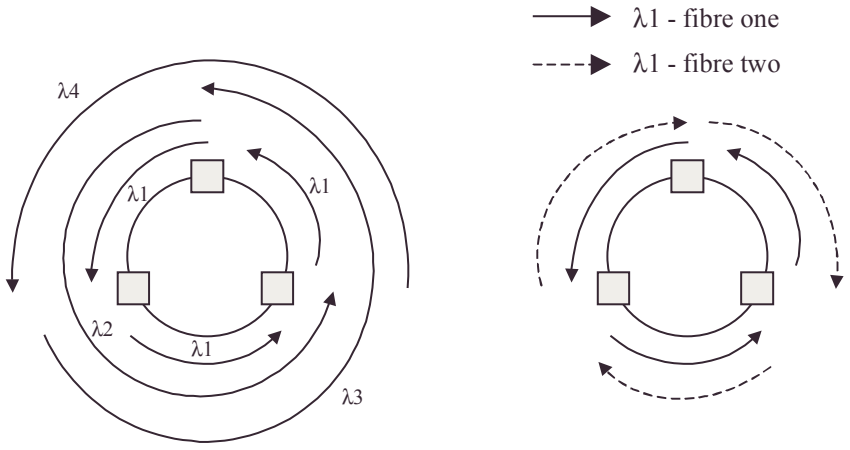
Recently, several studies on wavelength routing in ring networks have been performed. There are however two versions of the RWA problem. In the off-line version of this latter problem [88, 89], the main objective is to find the minimum number of wavelengths needed to accommodate a given set of lightpaths (source-destination pair) so that no two lightpaths that share a link are assigned the same wavelength. On the other hand, on-line versions [90, 91] consider the case with dynamic connection arrivals and removals. It that case, the main objective is typically to minimise the call blocking probability [88]. Moreover, wavelength conversion, which is the ability to convert the data on one wavelength to another wavelength, can be considered in both versions of the RWA problem. In the case where wavelength conversion capabilities are available [92, 93], the number of required wavelengths can be reduced but it usually increases the implementation cost of the network.

However, most of the above studies concentrate on a limited set of connections, i.e. the case where nodes are not fully connected to each others. While this is useful if the topology has to support a higher level network (e.g. SONET/SDH), it is restrictive in the sense that not every node can communicate with all other nodes in the network. In the following part of this section, we will only focus on the case of a fully-

interconnected ring where any node can transmit to all other nodes within the network.

We first consider the simple static wavelength routing network shown in Fig. 2.20 where three nodes are fully interconnected and no wavelength conversion capabilities are provided. In the first case (a), the ring consists of a single fibre and four wavelengths are required to fully interconnect the nodes. In the second case (b), we consider a counter-rotating dual-fibre ring and in that case only one wavelength per fibre is required, and it can be the same for each fibre assuming that each node has adequate receiver capabilities.

In fact, it is easy to derive that, to fully interconnect N nodes, $N(N-1)$ connections are required. We also want to introduce an important parameter as defined by Ramaswami [92], which is the network *load* (C), i.e. the maximum number of connections carried by any link within the ring.



(a) Single-fibre ring (b) Dual-fibre ring

Figure 2.20. A wavelength routing WDM ring

In Fig 2.20 (a), it can be seen that $C = 3$. In fact, in the case of a fully interconnected ring, it is also easy to derive that $C = \frac{N(N-1)}{2}$ [89].

Moreover, Tucker [94] already demonstrated that in the latter case, a maximum of $W = 2C - 1$ wavelengths are required to solve the RWA problem and better solutions can be found. Indeed, with $N = 3$ only 4 wavelengths are required ($2C - 1 = 5$).

If we now consider Fig. 2.20 (b), the number of wavelengths required is greatly reduced. In fact, Marcenac [93] reported that a shared protection ring (denoted OMS-SPRing) needs only $W = (N^2 - 1)/4$ wavelengths to provide full connectivity. This represents the optimum value and different algorithms have been considered to allocate the wavelengths and the optimum value is the same with or without wavelength conversion [93].

The above results are summarised in Table 2.6.

N	*L* (single-fibre)	*W* (single-fibre)	W (OMS-SPRing)
3	3	5	2
4	6	11	4
5	10	19	6
6	15	29	9
7	21	41	12
8	28	55	16
9	36	71	20
10	45	89	25
11	55	109	30
12	66	131	36

Table 2.6. Number of wavelengths required for full connectivity [93]

It can be seen that the use of a second fibre greatly reduces the number of wavelengths required and moreover, it also permits the implementation of protection mechanisms. This feature is very attractive in the metropolitan networking environment where reliability is of very high importance.

One of the main advantages of static wavelength routing rings is that, in the case of a fully interconnected network, a node has access to the full bandwidth of a wavelength for any given lightpath. On the other hand, such networks require a large number of wavelengths. For example, in the optimal case of the OMS-SPRing, 100 wavelengths would be required to accommodate the full connectivity of only 20 nodes. In real networks, however, dynamic allocation schemes are required to cope with uncertain time-varying traffic. These schemes are expected to greatly reduce the number of wavelengths required and to improve bandwidth efficiency. However, recent studies [90, 91] consider restrictive sets of lightpaths and are not practically ready to be implemented in a real network.

2.6.2 Broadcast-and-select ring networks

In the last few years, there has been a number of proposed broadcast-and-select ring architectures. Indeed, the progress achieved in optical amplifiers allows for the compensation of insertion losses at intermediate nodes, which was the main disadvantage of the ring when compared to the star topology. WDM ring architectures have been proposed [96-100] both for slotted and token passing schemes and these will be presented in the next two sections.

2.6.2.1 WDM slotted ring architectures

Early implementations of slotted rings were severely limited by the very low bandwidth-delay product. Nowadays, this is clearly not a limitation. Indeed, in high-speed networks, the duration of a slot (containing a fixed number of bits) is becoming shorter and shorter as compared to the ring propagation time and hence, a slot, is much smaller than the size of the network it moves in. For example, for a ring length of 1000 metres and a data rate of 10 Gb/s, and if we consider a signal propagation speed of 2×10^8 m/s, the ring can contain 50,000 bits at any instant. Moreover, in MAN ring with a circumference of, say 100 km, the ring capacity can reach 5,000,000 bits.

This buffering capability leads to the concept of slot reuse often pointed out as the major difference with respect to traditional low speed networks: several slots can be carried simultaneously through the optical medium. If slots are freed after reaching their destination, they can be used several times as they propagate around the ring. Therefore, much better throughputs can be achieved in ring networks than in star networks where space reuse is not possible.

Moreover, WDM technology allows for multiple ring architectures where a number of channels run in parallel. Bhuyan et al [95] demonstrated in the late 1980's that multiple rings exhibit much better performance than single rings. However, in this study, the network considered had a very low bandwidth-delay product (namely 300 bits) and a source stripping scheme was used, which both limited the performance.

2.6.2.1.1 HORNET

The feasibility of designing a WDM slotted ring network was partially demonstrated by the Hybrid Opto-electronic Ring Network (HORNET) project [96, 97] developed by the Stanford Optical Communications Research Laboratory and the Sprint Advanced Technology Laboratories. HORNET is expected to be used as a WDM metropolitan ring network that could be capable of transporting IP packets or ATM cells directly over the WDM layer.

It is a TT-FR architecture which is designed to scale to 100 access points (AP) and a circumference of around 100 km. It is built as a multi-hop architecture as multiple APs can share the same drop wavelength, and packets carry the destination AP address on a subcarrier-multiplexed header. When an AP drops its designated wavelength, it checks the packet's destination. If the packet is destined for a downstream AP on the same drop wavelength, it is retransmitted on the same wavelength until it reaches its destination. The main advantage of multi-hopping is that it introduces signal re-generation at intermediate nodes [96]. On the other hand, a carrier sense multiple access with collision avoidance (CSMA/CA) MAC protocol based on the sub-carrier multiplexing scheme was implemented to arbitrate the transmission of packets. This protocol allows APs to monitor traffic on all wavelengths to avoid packet collisions when transmitting. When an AP wants to transmit a packet to a given destination, it multiplexes the sub-carrier frequency that corresponds to the destination AP and, assuming that there is no other packet on the destination AP's assigned wavelength, it can transmit the packet and the multiplexed sub-carrier tone. The main advantage of using sub-carrier multiplexed header is that

the header to data ratio is no longer a limitation on bandwidth efficiency as the header is transmitted in parallel with the data.

The first demonstration of the HORNET network used fixed-sized slots whose length was equal to the length of an ATM cell (53 bytes). In a second demonstration, the ring contained three different slot sizes which were more suitable to carry variable sized IP packets. In both cases, synchronisation was achieved on packet-by-packet (or cell-by-cell) basis.

Despite the fact that the HORNET testbed is not fully operational, it has already demonstrated the feasibility of designing a slotted WDM ring network with subcarrier multiplexing tones being used for addressing purposes. However, the performance of the network under a variety of traffic conditions still needs to be evaluated.

2.6.2.1.2 The SR3 MAC Protocol

The Synchronous Round Robin with Reservations (SR3) MAC protocol was proposed by Marsan et al [98, 99] for the particular case of multi-channel slotted rings where nodes are equipped with a tuneable transmitter and a fixed receiver (TT-FR systems). SR3 combines a packet scheduling strategy (Synchronous Round Robin or SRR), a fairness control algorithm (Multi-MetaRing or MMR), and a reservation mechanism.

In [99] the case where the number of wavelengths (W) is equal to the number of nodes (M) is considered and the case where $W < M$ is studied in [98]. For simplicity, we will describe the case where $W = M$. All nodes that need to transmit to the same

destination share the same logical channel. Fixed-length data packets are transmitted, and the slot size is such that one packet exactly fits into one slot.

It can be observed that, due to ring symmetries, each node has better than average access to the channels leading to some destinations, and worse than average access to other channels leading to other destinations. Indeed, since each channel brings information to one destination and a destination stripping scheme is considered, slots from a given logical ring are emptied by the channel destination. The multi-channel ring can thus be viewed as a set of staggered logical unidirectional buses and for a specific destination, the node that is at the other end of the bus (with respect to the destination) has better access than the one that is the direct upstream neighbour of this destination. The dependence of access opportunity on the position along the multi-ring raises fairness problems. One node can use all the slots leading to a given destination, thus possibly starving downstream nodes competing for access to that channel.

The main objective of the SR3 protocol is therefore to alleviate fairness problems and also guarantee good utilisation of the bandwidth. It is subdivided into three schemes defined as follows. The Synchronous Round Robin (SRR) protocol is responsible for maintaining transmission fairness within a node itself. Indeed, each node has M-1 FIFO queues, each associated with a particular destination. In order to ensure that a node will equally transmit to all other nodes within the network, the SRR protocol ensures, in a cyclic manner, that packets from all queues are equally transmitted. A second scheme called Multi-MetaRing (MMR) is used to guarantee transmission fairness among the nodes in the network. Fairness is granted by the circulation of a

control message, named SAT, which normally rotates in the opposite direction (i.e. on a second ring) with respect to the data traffic it is regulating. Nodes are assigned a maximum number of packets to be transmitted between two SAT visits. Each node normally forwards the SAT message on the ring with no delay, unless it is not SATisfied, in the sense that it has not transmitted the permitted number of packets since the last time it forwarded the SAT. The SAT is delayed at unsatisfied nodes until satisfaction is obtained, in the sense that either the node packet buffer is empty or the number of permitted packet transmission is achieved. Finally, a reservation scheme is introduced in [99] in order to allow nodes to reserve slots. SAT messages are modified to contain a reservation field (the SAT becomes a SAT-RF message) which can be used by a node to notify the other nodes in the network that a slot reservation request has been made. A limited number of slots can be reserved and the position of the reserved slots is automatically established by the access strategy in every node by using a global synchronisation algorithm.

Simulation results in [98] and [99] show that the intrinsic unfairness of the considered topology can largely be overcome. However, there is a number of limitations with the SR3 protocol. First, since the proposed network architecture only provides co-directional rings, the SAT message must propagate in the same direction as the regulated data. If the SAT is forwarded to the upstream node, the SAT propagation delay becomes very large as the message must traverse almost the entire network to reach its destination. Second, there are a number of configuration parameters that need to be precisely tuned (transmission quota, reservation limit) to achieve efficient networking performance and no information is given on how optimal values can be

derived. Finally, the proposed physical architecture is unfair in nature, and relatively complex mechanisms must be designed to overcome this situation.

2.6.2.2 WDM token passing ring architectures

Another approach when designing WDM multiple ring architectures is the use of multi-token protocols. In a multi-token ring, multiple tokens circulate within a ring and therefore simultaneous transmissions are possible. Bhuyan et al [95] demonstrated that multiple token rings achieve better performance than single token standard protocols, and the performance improvement was shown to be increased as a function of the number of available rings. As WDM technology now offers a very large number of wavelengths, WDM rings have become naturally a potential candidate to implement multi-token passing protocols.

As a result, Cai et al [100] proposed The Multitoken Interarrival Time (MTIT) access protocol as a suitable MAC protocol for WDM ring networks. The considered architecture makes use of W data channels and one control channel, with a total of $W+1$ wavelengths. A dedicated wavelength is used as the control channel for the purpose of access control and ring management. Each node has one fixed transmitter and one fixed receiver for each of the data channels, thus realising a FT^{W+1}-FR^{W+1} system. This architecture allows the nodes to transmit and receive packets independently and simultaneously on any data channel.

Access to the data channels is regulated by the MTIT protocol. Each data channel is associated with one specific token that is circulating among the nodes (using the

control channel) to regulate the access to the corresponding data channel. Thus, a total of W tokens circulate around the ring. A node is allowed to transmit multiple packets on a data channel as long as it holds the corresponding token. The token holding time is controlled by means of a target token inter-arrival time (*TTIT*) and a token inter-arrival time (*TIAT*). The *TTIT* value is agreed upon by all the nodes within the ring. The *TIAT* is defined as the time elapsed between two consecutive token arrivals at a node. As tokens are not distinguishable, two consecutive arrivals are likely to correspond to distinct tokens. Upon a token arrival, a node is allowed to hold that token for a period of time equal to $TTIT - TIAT$. When the token holding time is up, the node must release as soon as the current ongoing packet transmission is complete (early-token release). The token can also be released earlier if the node does not have more packets to transmit. If upon a token arrival the computed time $TTIT - TIAT$ is negative, the token is considered to be late and must be released immediately. The packets, after having completed a round trip, are removed by the source node.

The main advantage of this protocol is that concurrent transmissions on distinct channels are possible at a given node when two or more tokens are simultaneously held by the node. Moreover, if two tokens become closer one to the other, the *TIAT* value between the two tokens will decrease. Consequently, the second token will be held for longer times by a number of nodes, assuming that these nodes have enough packets to transmit. This will happen until the correct relative position of the two tokens is re-established. Since the same mechanism apply to any two pairs of adjacent tokens, the distribution of the tokens around the ring tends to be uniform. More importantly, the MTIT allows the transmission of variable size packets over the WDM network, an attractive feature not easily achievable in slotted rings.

On the other hand, this architecture requires as many wavelengths as there are nodes in the network, and this severely limits the scalability of such a network. Moreover, and as in any token passing schemes, complex token management mechanisms must be implemented. With current technologies, these mechanisms cannot be implemented in the optical domain and hence the control channel must be converted to electronic form at each node, the data must be processed, and then re-transmitted onto the ring. This may introduce undesirable delays and node complexity. Finally, the overall network performance is strongly dependent on the *TTIT*, and finding its optimal value is not trivial. Additionally, adding or removing nodes to the network implies computing a new optimal value of the *TTIT* and no mechanism has been proposed to implement such a task automatically (i.e. without requiring the need for human intervention).

2.7 Summary

This chapter has presented the two principal types of WDM network architectures, namely the broadcast-and-select and the wavelength routing networks. Different architectures and demonstrators were identified and described. This chapter has also presented the three original approaches that were considered when designing electronic ring networks. Finally, a number of WDM ring architectures were reviewed and their main features were described.

Chapter 3

Network architecture and protocol

3.1 Introduction

Stoll et al [101] describe the metropolitan networking market as being highly cost driven, and they emphasise the fact that, in order to be adopted in the MAN environment, the implementation and management costs of future WDM metropolitan networks must be reduced to a minimum. Moreover, a metropolitan network must be relatively flexible and scalable, in the sense that it is expected to satisfy a large number of possible configurations as the physical architecture is strongly dependent on the geographical relative positions of the access nodes. Finally, reliability is also an important factor that must be considered when designing a metropolitan network.

Rings impose low requirements on the optical hardware and on the network management system. Protection mechanism can be implemented at a relatively low cost because a ring is merely a collection of point-to-point links that can be isolated in the case of a link failure. Finally, network scalability is mainly achieved by the WDM

technology which enables access to an extremely large bandwidth, thus making it possible to support an ever increasing network traffic.

The main objective of this research was therefore to propose a WDM ring architecture which could satisfy most of the constraints of the metropolitan networking market. While protection mechanisms were not considered, the proposed work mainly focuses on scalability, flexibility and simplicity of the network architecture and operation.

3.2 Network architecture

The network architecture considered is presented in Figure 3.1.

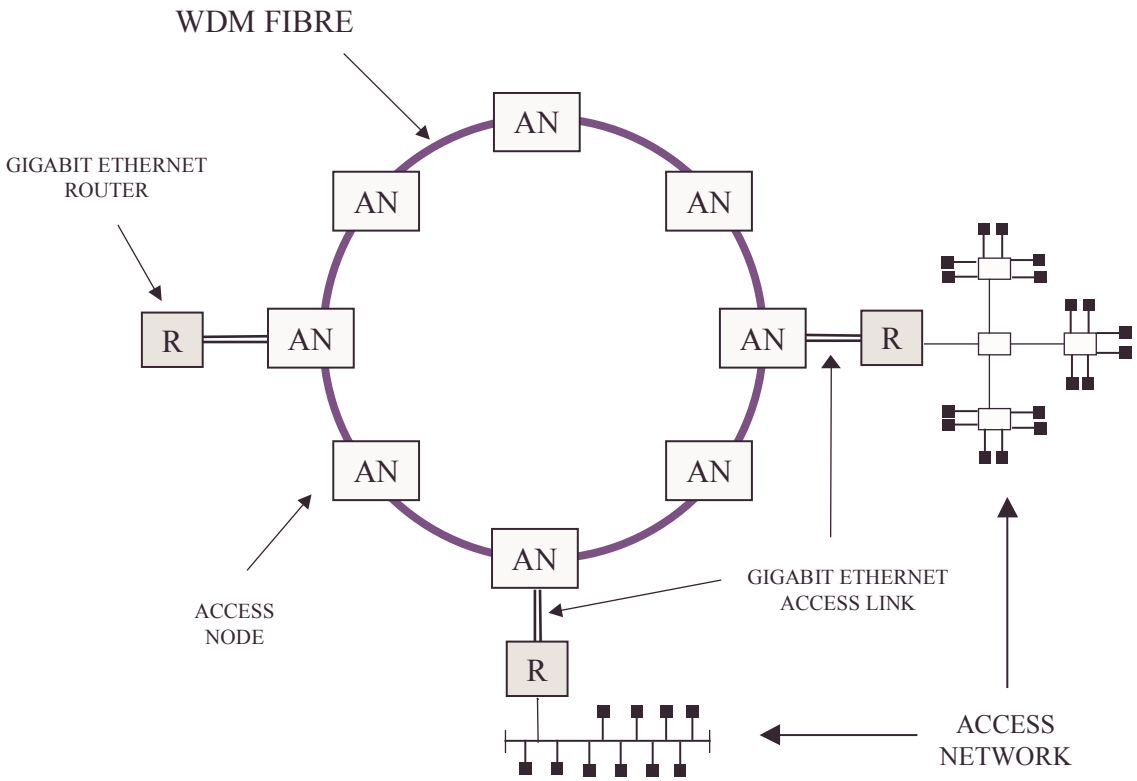


Figure 3.1. Network architecture

The network is a single-fibre multi-channel slotted ring which is designed to interconnect access nodes (ANs) on a regional scale (i.e. ring circumference of about 100 km). Each AN has add-and-drop capabilities to access the ring slots and is used to connect a local area network (or access network) to the ring. Since the vast majority of LANs throughout the world are Ethernet-based, the access networks are connected to the ANs by Gigabit Ethernet (GbE) links operating at 1 Gb/s. The slots on the ring have a fixed size which is equal to the Ethernet maximum transfer unit (MTU) frame size, i.e. about 12,000 bits (or 1,500 bytes).

To evaluate the scalability performance of the network, this architecture is considered in two comparable networking environments. In both cases, the length of the ring is identical, i.e. 138 km, which leads to a ring diameter of about 44 km. Also in both networks, the number of wavelengths is intentionally small and equal to four.

The first network is used to inter-connect 16 nodes, and each wavelength is considered to have a transmission rate of 2.5 Gb/s (i.e. OC-48). This first architecture is denoted **S-16/4/2.5**, i.e. **S**imulation with **16** nodes and **4** wavelengths, with each wavelength operating at **2.5** Gb/s.

This architecture is subsequently upgraded by a factor of 4, and is used to inter-connect 64 nodes, with each wavelength operating at a rate of 10 Gb/s (i.e. OC-192). This second architecture is denoted **S-64/4/10**, i.e. **S**imulation with **64** nodes and **4** wavelengths, with each wavelength operating at **10** Gb/s.

Table 3.1 summarises the main parameters of both architectures, alongside with the characteristics of the physical ring. B_{DP} is the bandwidth-delay per wavelength, and it must be multiplied by the number of wavelengths per fibre to obtain the total network bit carrying capacity.

Ring length : L_R	138 240 meters	
Light velocity in fibre : V	2×10^8 m/s	
Propagation delay : $D = L_F / V$	691.2 μs	
Number of wavelengths per fibre : N_W	4	
Slot Size : S	12,000 bits	
Architecture	**S-16/4/2.5**	**S-64/4/10**
Wavelength rate : R_W	2.5 Gb/s	10 Gb/s
Total network rate : $R_N = R_W \times N_W$	10 Gb/s	40 Gb/s
Number of access nodes : N_T	16	64
Bandwidth-delay product : $B_{DP} = R_W \times D$	1,728,000 bits	6,912,000 bits
Slots per wavelength : $S_W = B_{DP} / S$	144	576

Table 3.1. Network parameters

3.3 Access node architecture

Each access node has a fixed transmitter and four fixed receivers (i.e. FT-FR^4 system) which allows it to transmit on a unique wavelength and receive data on any wavelength. This architecture is almost equivalent to the FT-TR model, the only difference being that with a FT-TR system receiver collisions may occur. Replacing the tuneable receiver by four fixed receivers avoids receiver collisions.

Because there are less wavelengths than nodes in the network, each wavelength is shared both for transmission and reception. In the case **S-16/4/2.5**, 4 nodes will share a wavelength for transmission, and in **S-64/4/10**, 16 nodes will share a wavelength for transmission. In both cases, all nodes will receive information on all four wavelengths. This is not a limitation as the wavelength rate is greater than the node throughput and therefore a wavelength can be shared (for transmission) by a large number of nodes. The main advantage of this architecture is that the number of nodes can be much larger than the number of wavelengths, therefore satisfying the scalability requirements of the MAN market.

The FT-FR^4 architecture is preferred to the FT^N-FR^N model proposed by Cai et al [100] where the number of wavelengths must be equal to the number of nodes. This is clearly a scalability limitation and such an architecture is hardly suitable to the MAN environment. On the other hand, our proposed implementation is relatively similar to the TT-FR system presented by Marsan et al [98, 99], in the sense that the wavelengths are shared by a fixed number of nodes. However, in Marsan's proposal, each node can use any wavelength for transmission but it receives data on a fixed wavelength. Therefore, if there are less wavelengths than nodes, a fixed number of nodes will share a wavelength for reception, and all the wavelengths will be shared for transmission. Bononi [102] analytically evaluated the performance of both FT-TR (equivalent to our FT-FR^4) and TT-FR slotted architectures and concluded that they have a similar theoretical networking performance. However, one must notice that in our case, i.e. FT-FR^4, the scalability study of the network is greatly simplified. Indeed, if we assume that the maximum throughput achievable per wavelength is known, it is possible to know how many nodes can be assigned a common wavelength

for transmission (assuming that the nodes transmission rates are also known), and hence scalability predictions are relatively straightforward. In contrast, scalability predictions in a TT-FR system are difficult to achieve.

Because in our FT-FR^4 system, a node can receive packets on any wavelengths, each node is assigned a different sub-carrier multiplexed tone. When a node wishes to transmit to a specific destination, it transmits the packet onto a slot of its assigned transmission wavelength and multiplexes the destination's sub-carrier tone, which represents the packet's destination address. Meanwhile, each node constantly monitors all wavelengths in parallel in order to detect its own sub-carrier tone, which will notify it that it is the intended destination of the corresponding packet. The feasibility of such a mechanism for a slotted ring has already been demonstrated in the HORNET [96, 97] network project.

3.4 MAC protocol

The operation of slotted rings was described in Chapter 2 but it is slightly modified in our implementation. When a node wishes to transmit data on its assigned wavelength it simply inspects the activity of the channel. If an empty slot is observed, a packet can be transmitted in the slot data unit (we assume that the packet size is always equal to the slot size). If the slot is used, the transmission is simply delayed. In the case of source-stripping operation, the sender is responsible for marking the slot empty after it has completed an entire ring loop. With destination-stripping, the destination marks the slot empty once correctly received and thus makes the slot reusable earlier than in the previous scheme. A restriction is also introduced, which does not allow a node to

immediately reuse a slot it just marked empty. This scheme introduces a very simple fairness mechanism where nodes are forced to release empty slots to their downstream neighbours.

In order to derive theoretical predictions for the network performance, a number of assumptions are made. First, nodes are assumed to be equally spaced (physically) around the ring. Second, the nodes that share a wavelength for transmission are also equally spaced around the ring, i.e. if we assume node numbers to be increasing in the transmission direction, the channel on which node n_{i+1} ($i = 0, 1, \ldots, N_T - 1$) transmits is λ_{j+1}, with $j = i \bmod N_W$ (where N_W is the number of wavelengths). Figure 3.2 shows the logical topology (only transmission is considered) for the case **S-16/4/2.5**.

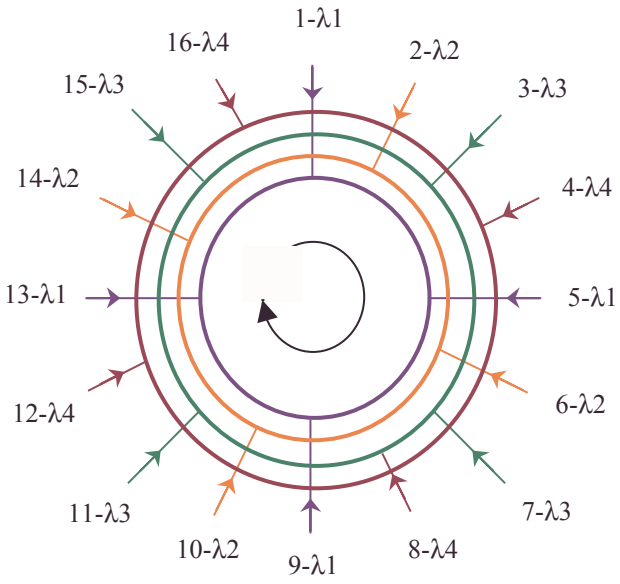


Figure 3.2. **S-16/4/2.5** : logical topology (transmission)

3.4.1 Theoretical predictions

Because the topology considered is a ring, the performance of the network is mainly dependent on the slot-reuse factor of the network. The slot-reuse factor is an analytical tool which is used to derive the bandwidth efficiency of the proposed ring architecture. These two parameters are defined in the following paragraph. We also define N as being the number of nodes that share a wavelength for transmission and, due to the previous assumptions, $N = N_T / N_W$, where N_T is the number of nodes in the network.

If a complete ring rotation is defined as being a normalised distance $d = 1$, then the normalised distance it takes for a slot, once filled, to be made reusable (not emptied), is defined as the slot-reuse factor S_r. For example, if the source-stripping scheme is used and if the restriction introduced in the design of the MAC protocol is ignored, a slot, once filled with data, must complete an entire ring rotation so that, upon its return, it can be emptied and reused by the sender. In this case $S_r = 1$, i.e. a complete rotation was needed for the slot to be made re-usable.

Furthermore, the bandwidth efficiency is simply defined as the maximum number of packets that can be transported by any slot during a full rotation round the ring. It should also be remembered that the packet size is supposed to be equal to the slot size. In that case, the bandwidth efficiency is simply equal to $\eta = 1/S_r$ and in the previous case $\eta = 1$.

However, if the MAC protocol restriction is considered with source-stripping, a slot, upon its return to the sender, cannot be reused immediately and it has to be released empty to the next downstream node that uses the same wavelength for transmission. In this case, and because it is assumed that nodes are equally spaced around the ring (i.e. the normalised distance between any two nodes sharing a common wavelength for transmission is equal to $1/N$), $S_r = 1 + 1/N$. In this case, the bandwidth efficiency is given by

$$\eta_s = \frac{1}{1+1/N} = \frac{N}{N+1}. \tag{3.1}$$

Now the destination-stripping scheme is considered. It is assumed that each node transmits to all other nodes with equal probability (i.e. $1/(N_T - 1)$). Therefore, if N_T is even and for $N > 1$, the slot-reuse factor is on average equal to $S_r = 1/2 + 1/N$ and the bandwidth efficiency with destination-stripping is given by

$$\eta_d = \frac{2N}{2+N}. \tag{3.2}$$

In the particular case where $N = 1$, the bandwidth efficiency is simply $\eta_d = 1$, i.e. slots are released earlier but cannot be reused by another node but the sender.

Figure 3.3 gives graphical representations of the three previous cases.

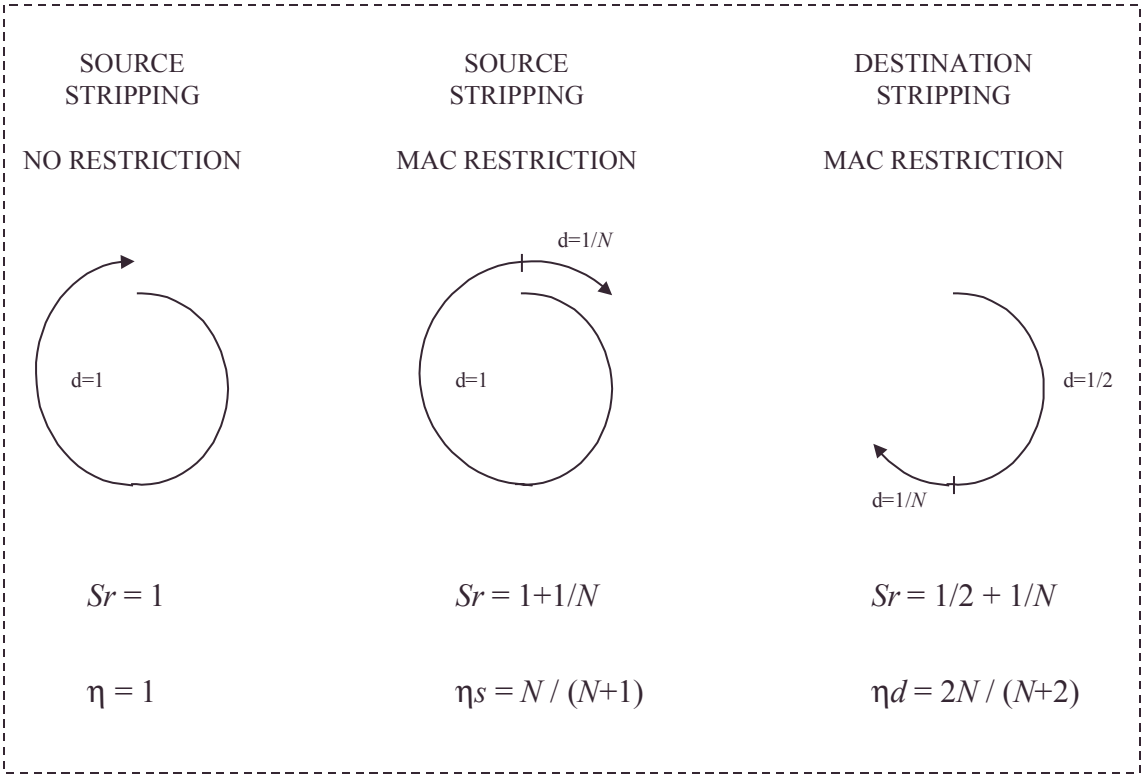


Figure 3.3. Slot-reuse factor and bandwidth efficiency

It is clear from equations (3.1) and (3.2) that the destination-stripping scheme greatly improves the bandwidth efficiency as slots can be reused much earlier when compared to source-stripping. This can be seen in Figure 3.4.

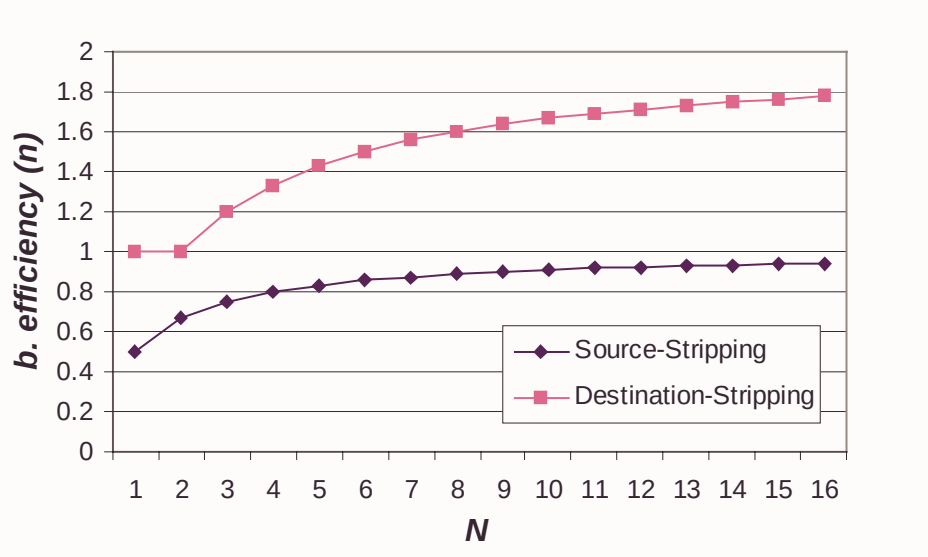


Figure 3.4. Bandwidth efficiency (η)

Moreover, it can be seen from equations (3.1) and (3.2) that in both cases, the bandwidth efficiency increases with N. Indeed, as N increases, the normalised distance between any two nodes that share a wavelength for transmission (i.e. $1/N$) decreases. Therefore, slots that have been released but not reused (because of the MAC protocol restriction) travel a smaller distance before they can be reused and hence, the effect of the MAC protocol restriction is reduced.

It is now assumed that **all nodes transmit at the same rate**. In this particular case, the maximum throughput per node T_{MAX} is thus simply derived from the bandwidth efficiency as being

$$T_{MAX} = \frac{\eta \times R_W}{N}. \tag{3.3}$$

The maximum throughput per node can therefore be derived for source-stripping ($T_{MAX\text{-}S}$) and destination-stripping ($T_{MAX\text{-}D}$) as equal to

$$T_{MAX-S} = \frac{\eta_s \times R_W}{N} = \frac{R_W}{N+1} \tag{3.4}$$

and $$T_{MAX-D} = \frac{\eta_d \times R_W}{N} = \frac{2 \times R_W}{N+2}. \tag{3.5}$$

In both cases, it can be seen that T_{MAX} decreases when N increases, because a larger number of nodes share the same wavelength for transmission. From equation (3.4) and in the particular case where $N = 1$, $T_{MAX\text{-}S}$ is equal to $R_W / 2$, i.e. half the available wavelength bandwidth. However, in practice this is not a limitation as nodes do not transmit data at a rate faster than $R_W / 2$ (e.g. 1 Gb/s against 10 Gb/s). Also, in the majority of cases, more than one node typically share a wavelength for transmission.

Moreover, and in both cases, if the nodes transmission rate is much smaller than the wavelength rate, a large number of nodes is required for T_{MAX} to decrease below the actual nodes rate. For example, in the case of source-stripping with $R_W = 10$ Gb/s and four wavelengths ($N_W = 4$), nine nodes are required to reduce $T_{MAX\text{-}S}$ to the nodes rate of 1 Gb/s and such a network could accommodate 36 nodes without experiencing congestion. With destination-stripping and under the same wavelength rate, 18 nodes are required to reduce $T_{MAX\text{-}D}$ to the nodes rate of 1 Gb/s and the same network could be used with 72 nodes without congestion.

Furthermore, the previous results can be extended to the cases where the early assumptions are ignored.

The first assumption was that the nodes were equally space around the ring. It is now still assumed that the logical topology is the one presented in Figure 3.2 but nodes do not need to be equally spaced. In this particular case, the normalised distance between any two nodes sharing a wavelength for transmission is, on **average** this time, still equal to $1/N$. This means that the bandwidth efficiency and the maximum throughput per node remain identical for both the source- and destination-stripping schemes.

In addition, if it is now assumed that each node does not transmit to all other nodes with equal probabilities, the previous results are unchanged with source-stripping because slots are emptied by the sender and the destination nodes are not involved in the slot-release operation.

Finally, and with source-stripping, if we suppose that the nodes transmission rates are different, an unequal amount of empty slots will be released by each node to the first downstream neighbour that uses the same wavelength for transmission. However, this downstream neighbour may use some or all of the slots it receives from its upstream neighbour, but eventually all these slots will be released to the next node down the stream, i.e. the unused slots will just go through and the used ones will be released after one ring rotation. Since the same mechanism applies to any node, the same average amount of empty slots will reach any node every N frames (the time duration

of a ring round). On the other hand, the instantaneous nodes throughput may indeed fluctuate in a random manner.

With destination-stripping, if the nodes transmission rates are different but if it is still assumed that each node transmits to other nodes with the same probability, a similar self-adjusting mechanism as the one described in the previous paragraph will take place. On average, an equal amount of slots will be emptied and released by all nodes as each node will receive an equal proportion of the traffic sent by the other nodes. On the other hand, if each node does not transmit to the other nodes with equal probabilities, the network behaviour becomes unpredictable and it can typically be analysed only through simulations. However, in the particular case where one node is the preferred destination of most of the other nodes' traffic (e.g. the node linking the ring to the WAN backbone), this destination will evenly release slots on all of the wavelengths. As the amount of traffic received by this particular node may be much larger than the other nodes throughput, a small amount of the released slots will be used by the first nodes down the stream and the rest will possibly go through to all of the other downstream nodes. In spite of this, the nodes further down the stream might be starved if all of the slots are used before reaching them. This problem can however be solved by using a second counter-rotating ring but this is out of the scope of this study.

It was also assumed that packets transmitted by the nodes have a fixed length equal to the slot size. This assumption will still be maintained in the following chapters.

3.5 Summary

In this chapter, the ring network architecture was described and the main parameters of the two networks considered were presented. The access nodes architecture was also introduced and the design of both the network and the access nodes was compared with previous work from other researchers. Finally, the operation of the medium-access control protocol implemented was described and theoretical predictions on nodes throughput and performance were derived and discussed.

Chapter 4

Design and implementation of a network simulator

4.1 Introduction

In the study of communication networks there are three basic suitable approaches for investigating the performance of a system. These three techniques are identified by Jain [103] as analytical modelling, simulation, and measurement.

Analytical modelling uses mathematical analysis to evaluate a given system and it has the advantage that it is usually the quickest method. It is also quite inexpensive as it does not require the use of costly equipment and instruments. However, analytical modelling generally requires many assumptions in order to simplify the problem to a level that makes analysis practical. The level of accuracy is therefore often open to question.

Simulations are used to model a system and its behaviour, usually by means of a computer program, in a form suitable for deriving information about the system's characteristics and properties that are of interest to the study. Simulation techniques are usually more accurate as they can incorporate more details and require less assumptions than analytical modelling and, thus, are more often closer to reality. Moreover, the ever-increasing power of computer systems and simulation software has made the approach an attractive and viable option.

The third technique, i.e. measurements, involves testing actual working systems and networks, and it is usually the most expensive method as it often requires the use of expensive measurement instruments alongside with the cost of actually building a system prototype. It is generally very accurate as every detail can be included in the study.

While the analytical modelling of the proposed architecture and protocol was presented in the previous chapter, a network simulator has been implemented to evaluate the performance of the network. It is a *discrete event* simulation [104], i.e. it models the system as a sequence of events, where it can be assumed that nothing of interest takes place between those events. A slotted ring is the perfect example of a system that can be modelled by a discrete event simulation because the gradual rotation of the slots can be simplified and modelled in a discrete manner.

There are a number of network simulators available such as OPNET, NS (Network Simulator) and OMNet++. OPNET is a professional simulation package that can support the modelling of a very large number of communication networks and

systems but because it supports such a large variety of networks, it is very expensive and its price makes it unaffordable in the present case. On the other hand, NS and OMNet++ are both freeware packages and both have very powerful modelling features. Nevertheless, these two products are not easy to master and moreover, none of them provided the necessary building components to model our proposed WDM ring topology and it would have taken a very long time to develop a specific module that can be incorporated into one of these packages. It was therefore decided that a network simulator should be developed from scratch.

4.2 Simulator design

A specific discrete event simulator has been developed to assess the performance of our proposed network. The programming language that was used is C because it is extremely flexible, quick, and reliable and moreover, it has been designed so that separated modules can be developed independently and can afterwards be combined altogether [105]. The main advantage of this technique is that a complicated development can be subdivided into smaller and less complex modules in order to reduce the complexity level faced by the developer.

A multi-channel slotted ring can be modelled extremely well by a discrete event simulation because the rotation of the slots can easily be described in a discrete way. Indeed, all the details of the physical architecture are known and because the slots are of fixed size, the gradual propagation of slots around the ring can be simplified without affecting the accuracy of the study. Indeed, there are two main events of interest in the slots propagation scheme: first the arrival of a slot at the edge of a node,

and second, its departure. The state (i.e. full or empty) of a slot is known when it reaches the edge of a node and the only aspect of interest is then to know the state of the slot when it leaves the node. Because the amount of time it takes for a slot to traverse a node is a fixed known network parameter, time can therefore be expressed in a non-continuous way, i.e. the smallest time duration we consider is the time it takes for a slot to propagate through a node. If this slot propagation time is δt, then the minimum time increase in the simulator is equal to δt.

Writing a network simulator is a very challenging task as a large variety of simulation results are expected in order to accurately study the behaviour of the network studied. Moreover, the network traffic applied to the simulator must be realistic because all the results depend on the validity of the traffic that was used within the simulations.

One must also keep in mind that the main objective of the simulator is to evaluate the performance of the network and therefore it is important to define what parameters are to be used to evaluate the network operation. Classically, in network performance study, there are a basic number of parameters that has to be considered: node throughput, packet queuing delay in transmission buffers, buffer load, packet dropping probability, and packet end-to-end transmission delay. The main constraint when developing the simulator was therefore to implement accurate mechanisms to log the above parameters values during the course of a simulation.

The network simulator that has been designed is capable of measuring the parameters described above for each access node in the network. It provides average results for the entire course of a simulation and moreover, it also logs these values on a frame-

by-frame basis , i.e. for each frame (a frame is the time it takes, with respect to the slots, to complete an entire rotation around the ring) the average parameters values are logged in order to evaluate the dynamic behaviour of the network. In addition, a slot-by-slot logging capability has also been implemented but it generates an extremely large amount of results and these were only used to validate the operation of the simulator.

4.2.1 Implementation of the simulator

In order for the simulator to be as modular as possible, a number of different objects has been defined. Each object is defined in a different C module but some of them have access to the properties of other objects to which they are linked. The central object defined in the simulator is the node. A node has a number of properties and it is the point of origin of any request that is made in order to have access to the properties of the objects it is linked with. This is depicted in Figure 4.1.

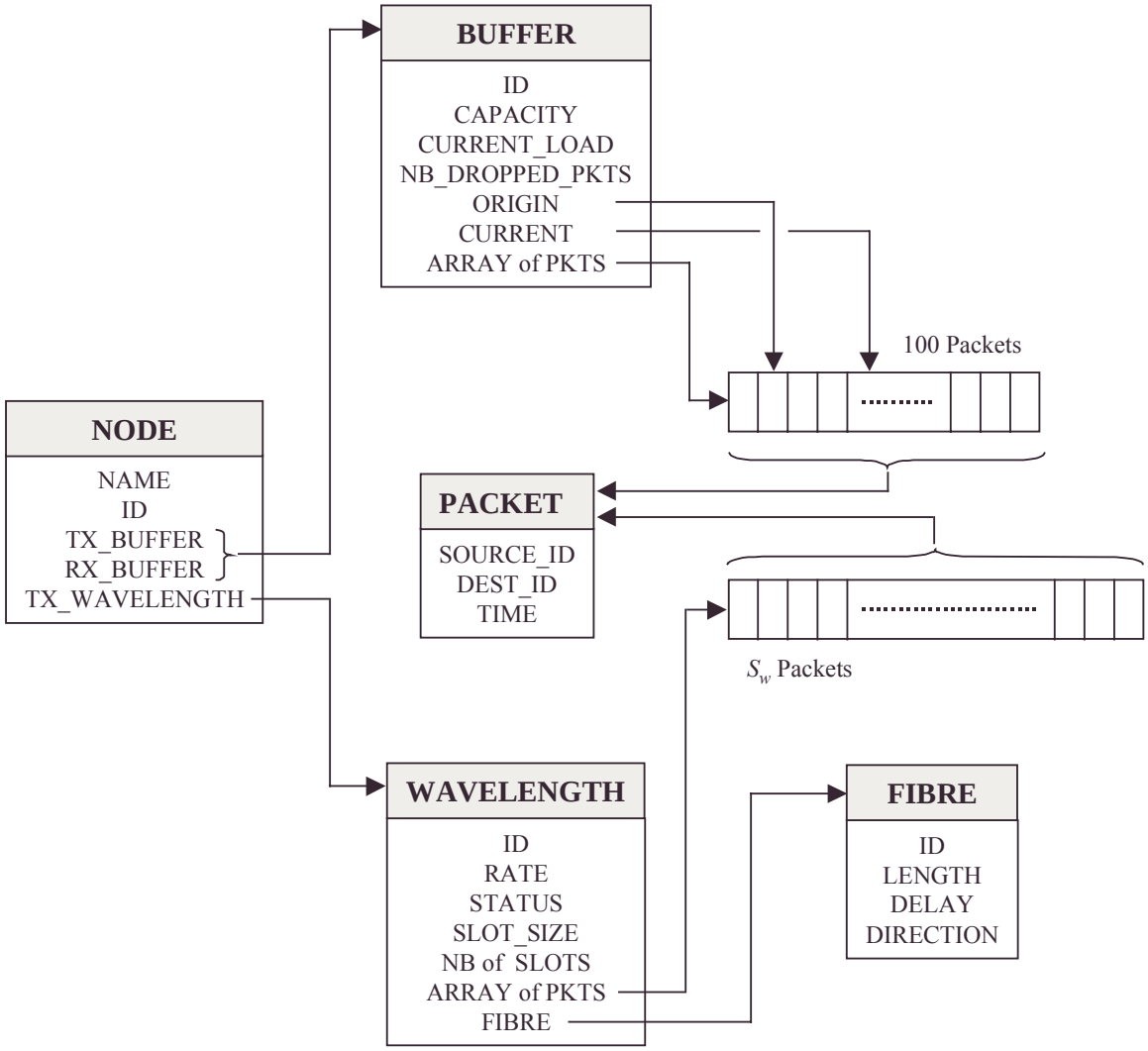


Figure 4.1. Schematic diagram of the objects interconnections

As seen on Figure 4.1, each object has some properties associated with it. For example, a fibre is characterised by its ID (identifier, i.e. a unique number), its length (in meters), its propagation delay (in microseconds) and its direction (clockwise or anti-clockwise). The arrows indicate that the object at the origin of the arrow is linked to the object pointed by the arrow. For example, a node, which is characterised by its name, its ID, its buffers (for the sake of simplicity only one buffer is represented), and

the wavelength it was assigned for transmission. The field TX_WAVELENGTH is simply a pointer to the memory address of the wavelength object. Furthermore, a node has access to its wavelength's properties, and moreover, it also has access to the fibre properties associated with this wavelength.

One of the most basic elements of the simulator is the packet. A packet is defined by its source and destination addresses, and by the time it was created. As already stated in the previous chapter, packets have a fixed length that is equal to the slot size. Therefore, slots and packets are identical objects in the simulator. Nevertheless, we will still refer to slots when talking about the actual slots of a wavelength. An empty slot is simply a packet with both addresses equal to zero (as there is no node 0 within the network). The time field in the packet (given in μs) is set when the packet is sent to a node by its access network, and it is used to compute both the queuing and transmission delays. The start of a simulation is defined as the time origin ($t = 0$).

If we now consider the wavelength object, it can be seen that one of its field is a pointer to an array of slots (i.e. packets). The array is represented as a line but it does correspond to the slots around the ring. It can be seen as if the ring was cut at the edge of the node and both ends were stretched apart. Actually, it simply symbolises the memory implementation of the array. A simple modulo operation is used to loop back the array from the last slot to the first.

The buffer object also points to an array of packets which simply represents the FIFO (First-In, First-Out) stack of packets waiting to be transmitted (onto the ring if the transmission buffer is considered). The buffer maximum capacity is 100 packets. In

fact, the FIFO buffer operation is very simple. The buffer can be symbolised as a pile of packets. Packets are added up on the top and removed from the bottom of the pile when transmitted. However, implementing this mechanism is not so trivial. Indeed, shifting packets down the pile when a packet is removed is far too costly in terms of computer processing time. To overcome this problem, two pointers are used to represent the bottom (origin) and top (current) of the pile. When a packet is added to the buffer, the current pointer is first increased by one (i.e. in pointer arithmetic) and the packet is then copied to the new current location. When the bottom packet of the pile is removed from the buffer, the origin pointer is simply increased by one. When any of these two pointers reaches the top of the pile and is latter increased by one, it is simply taken back to the start of the array by a simple modulo operation.

The memory implementation of the buffer operation is shown in Figure 4.2. From (a) to (b) a packet is simply added on the top of the buffer. From (b) to (c), a packet is removed from the bottom. Finally, from (c) to (d), a packet is added on the top of the buffer and the CURRENT pointer is taken back to the start of the array by the modulo operation.

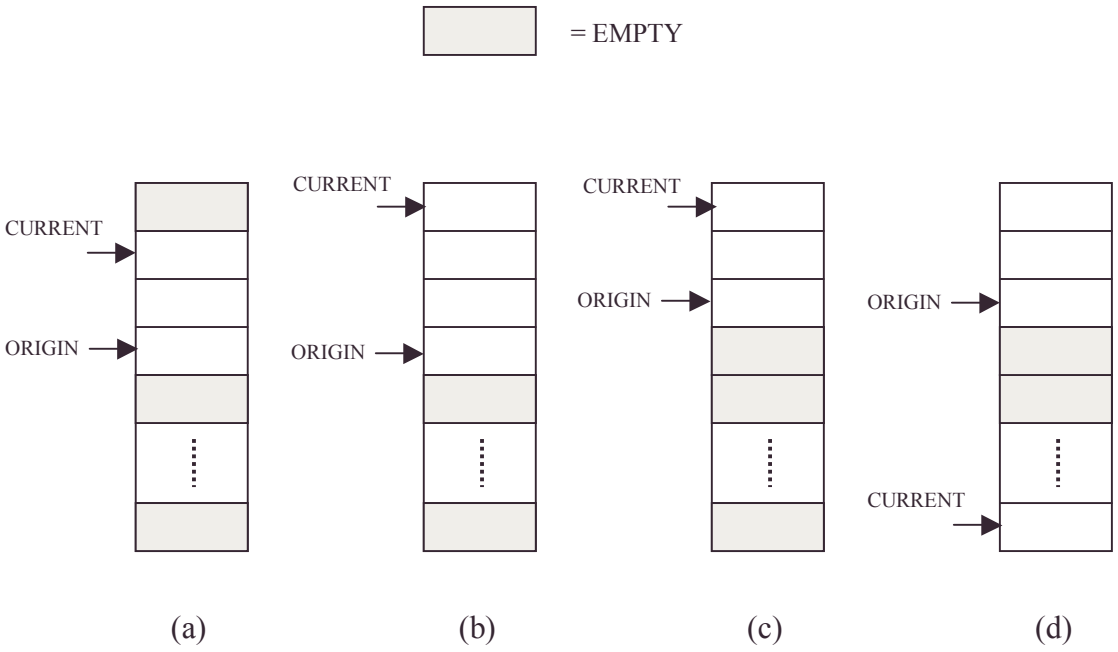


Figure 4.2. Memory implementation of the buffer operation

Actually, each object is associated with a number of operations that can be applied to that object. For example, there are 3 basic operations for the buffer object: a buffer can be initialised, a packet can be added to a buffer, and the bottom packet of a buffer can be removed. There are in fact more functions but they are not basic operations and will not be detailed (e.g. copy the bottom packet of the buffer to an empty slot).

In fact, the objects and their associated functions constitute the core elements of the simulator. So far, most of the properties of the network can be defined and initialised. Specific modules (the INIT functions) have also been created to facilitate the creation and initialisation of the objects because there can be a very large number of objects as in the **S-64/4/10** architecture, i.e. 64 nodes and their associated objects. More functions have also been designed to open the files that will store the simulations

performance results. These functions (the LOG and RESULTS modules) are also used to trace the network performance.

However, the simulator is still a static system, and thus a transmission scheme has been developed to control the operation of the simulator. This module is called the Data Access Control (DAC) and it is actually the C implementation of the MAC protocol. It is responsible for transmitting and removing (either source- or destination-stripping) packets to and from the slots, controlling the rotation of the slots around the ring, and moreover, managing the arrival of packets from the access networks to the nodes. The traffic sources used to simulate the activity of the access network are detailed in the next sections, but it is worth mentioning here that, due to technical restrictions explained later, this traffic is generated prior to the launch of the simulations and it is stored in large files which will further be described.

The DAC operation is shown in Figure 4.3. Each step of the flowchart involves a number of functions which will not be detailed. For example, the MAC protocol restriction on slot-reuse in controlled by a simple mechanism, i.e. a control flag, which is set during the packet removal procedure. If a node empties a slot from the wavelength it is assigned for transmission, the flag is set to TX_NOT_ALLOWED and the node is not allowed to reuse the slot. The mechanisms implemented to manage the network traffic are described in the following section.

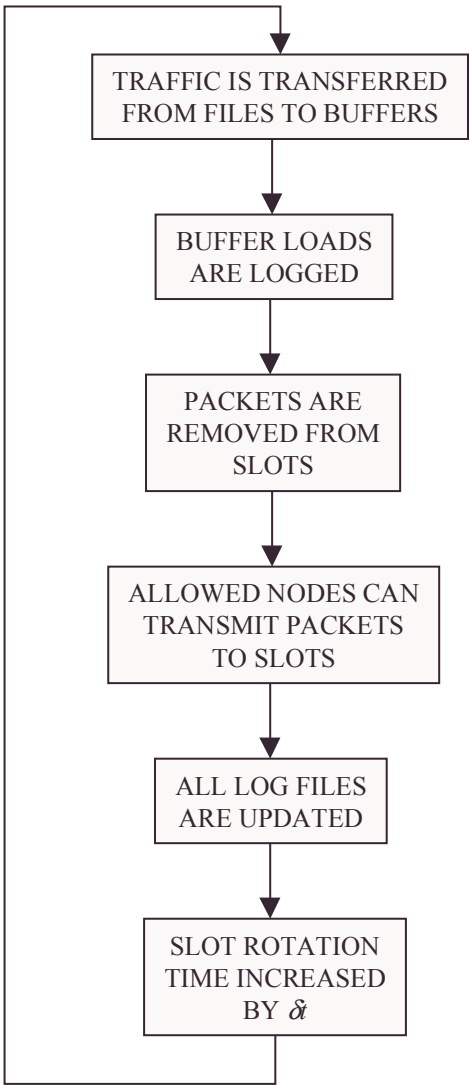


Figure 4.3. DAC flowchart

As seen earlier, the DAC module has brought the simulator to life and it is now capable of simulating our proposed ring architecture and the MAC protocol associated with it. Unfortunately, due to the lack of space in this thesis, the implementation of the simulator could not be described in details as it is a very large development whose complexity cannot be summarised in an easy way. The complete and commented C listings of the modules can be found in Appendix 2. The makefile (see [105], Chapter 8 for details) used to combine the different modules in order to build the simulator can also be found in Appendix 2.

Finally, the simulator was compiled and run on a FreeBSD 4.2 operating system. The development environment provided by this BSD-like UNIX system is very powerful and provides a very large variety of programming tools that are very helpful to the programmer. The GNU C compiler (i.e. gcc), a freely available product, was used to compile the simulator and was found to be very efficient in optimising the execution time of the simulations, i.e. less than 30 seconds in the worst situations. In general, a simulation generated 8 files per node in order to trace the network performance, plus an additional number of files used to control and evaluate the behaviour of the simulator.

4.2.2 Traffic sources

Designing and implementing the network simulator was a challenging task and moreover, probably an equal amount of time was spent in checking and validating its operation. The simulator can therefore be considered as a reliable tool that can accurately model the proposed ring architecture and MAC protocol. In fact, this was made possible because the simulator was designed to model a system whose operation was strictly defined. Further reassurance regarding the simulator operation, and the accuracy of the analytic predictions made, was obtained when agreement was later observed between simulation and analysis. The comparisons are given in Chapters 5 and 6.

However, the whole simulator performance depends on one and only one input parameter, i.e. the network traffic applied to the access nodes. Indeed, for a given architecture (either **S-16/4/2.5** or **S-64/4/10**), the only varying parameter is the model

used to simulate the access networks traffic. Various traffic loads can of course be simulated but one can only produce accurate results if the traffic model used is realistic, i.e. whose characteristics are close to the traffic measured in real networks.

The long-held paradigm in the communication and performance communities has been that voice traffic and, by extension, data traffic are adequately described by certain Markov models (e.g. Poisson). Poisson processes have attractive theoretical properties which lead to well-developed analysis techniques that could predict tight bounds on performance results. This so-called "teletraffic theory" was extremely suitable for describing the behaviour of the public switched telephone network (PSTN) and Willinger [106] does not hesitate to describe it as "one of the most successful applications of mathematical techniques in industry". The tremendous success of teletraffic theory was mainly due to its simplicity, physical interpretation and practical relevance. As a result, the Poisson traffic model, whose full dynamics can be described with one parameter, had been widely adopted in network simulations.

However, Leland et al [107] demonstrated in a seminal paper that the above teletraffic model was severely unsuitable to modern communication networks whose traffic is highly variable and bursty. That is, it does not come at steady rate, but instead in bursts which occur on many different time scales. In contrast, the Poisson traffic "smoothes out", becoming quite steady when the time scale increases. The relevant mathematics that describes modern data networks is thus one of *high variability* and *self-similarity* [107].

Figure 4.4 [106] is a visual demonstration of the failure of Poisson modelling to capture the burstiness present in actual data network traffic over a large range of time scales. The first column represents self-similar traffic and the second column corresponds to the Poisson model. The time scale of the top row is 100 ms, i.e. each point in the plots reflects the number of packets observed during a 100 ms interval.

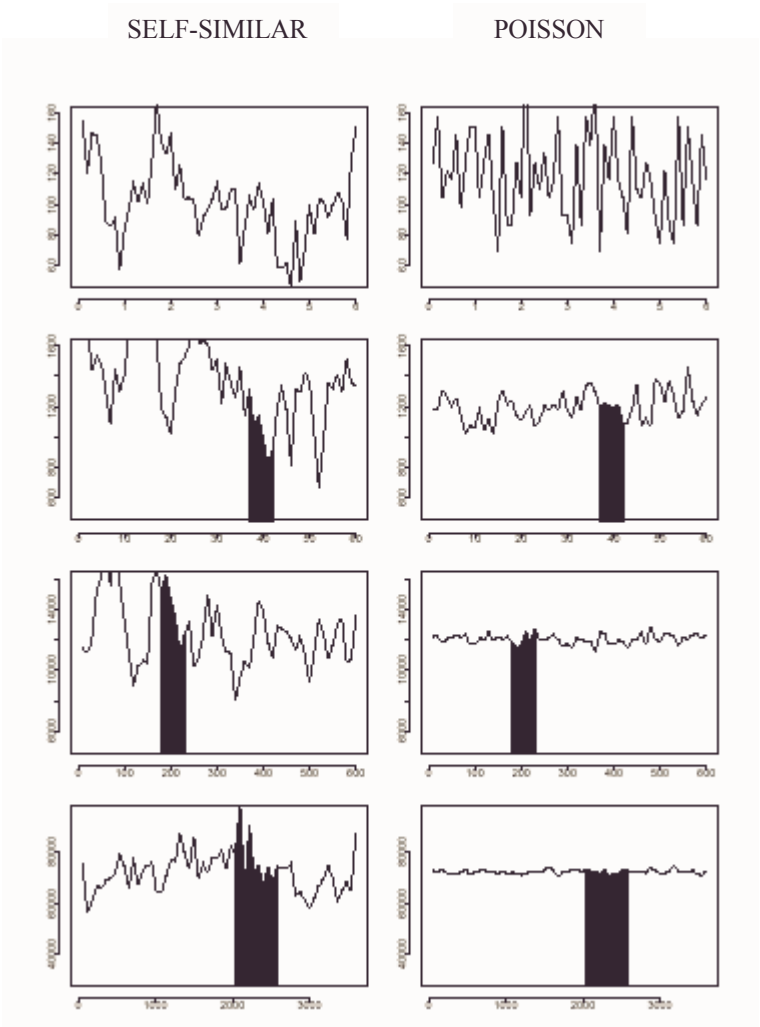


Figure 4.4. Synthesised traffic from self-similar and Poisson models [106]

The second row shows a time scale a factor of ten larger and, moreover, the Y-axis has also been increased by a factor of 10. With the third and fourth row, the scale was again increased by a factor of 10 with respect to the scale of the previous row.

The self-similar model used in [106] was found to exhibit the same characteristics as the traffic measured in modern data networks. Traffic from the self-similar model is bursty over a large range of time scales and, in contrast, traffic from the Poisson model looses its burstiness as the time scale increases.

Moreover, while the earlier work in [107] demonstrated the self-similar nature of data traffic in local area networks, Paxson and Floyd [108] extended this study and verified that network traffic also exhibited self-similarity in metropolitan and wide area networks. Typically, the degree of burstiness of self-similar traffic is expressed by the *Hurst* ($0.5 \geq H > 1$) parameter, i.e. the burstiness increases with H.

It was therefore essential to use self-similar traffic sources in our simulator in order to obtain performance results based on a realistic network traffic model. In fact, both Poisson and self-similar models were designed in order to evaluate the impact of the traffic model on the architecture's performance. Both models were used to simulate the traffic of the Gigabit Ethernet links that interconnect the nodes and the access networks.

4.2.2.1 Poisson traffic

Generating Poisson traffic is straightforward. Packet inter-arrivals (i.e. for a given access link, the packet inter-arrival time is the time duration between two packet transmissions) are generated using an inverse transformation [103]. A uniform distribution U(0,1) is generated using a pseudo-random number generator and the exponential inter-arrivals (IAT) are computed as $-\Lambda \times \ln(u)$ where Λ is the mean

inter-arrival time between two packets. A different seed is used for each access link to initialise the pseudo-random number generator and different network loads are simulated using different values of Λ. The duration of a packet (in seconds) is $\tau = 12$ μs (i.e. 12,000 bits at 1 Gb/s).

The generated load (L) of a Gigabit Ethernet link is simply given as

$$L = \frac{\tau}{\tau + \Lambda} \tag{4.1}$$

and therefore

$$\Lambda = \frac{\tau(1-L)}{L}. \tag{4.2}$$

For example, for a load $L = 0.5$ and $\tau = 12$ μs, then $\Lambda = 12$ μs. Also for $L = 0.25$, the inter-arrival time is $\Lambda = 36$.

The C module generating Poisson traffic is presented in Appendix 3.

4.2.2.2 Self-similar traffic

On the other hand, it is more difficult to generate a network traffic which exhibits self-similarity. Although actual network traffic is intrinsically complex, Willinger et al [109] demonstrated that it can be modelled using parsimonious (relatively simple) models without the need for highly parameterised mathematical models. Indeed, self-similar traffic can be generated by multiplexing (aggregating) several sources of Pareto-distributed ON and OFF periods. ON periods correspond to packet-trains (i.e. packets transmitted back to back) and OFF periods are the periods of silence between

packets trains. Figure 4.5 is a graphical representation of the aggregation process of three ON-OFF sources.

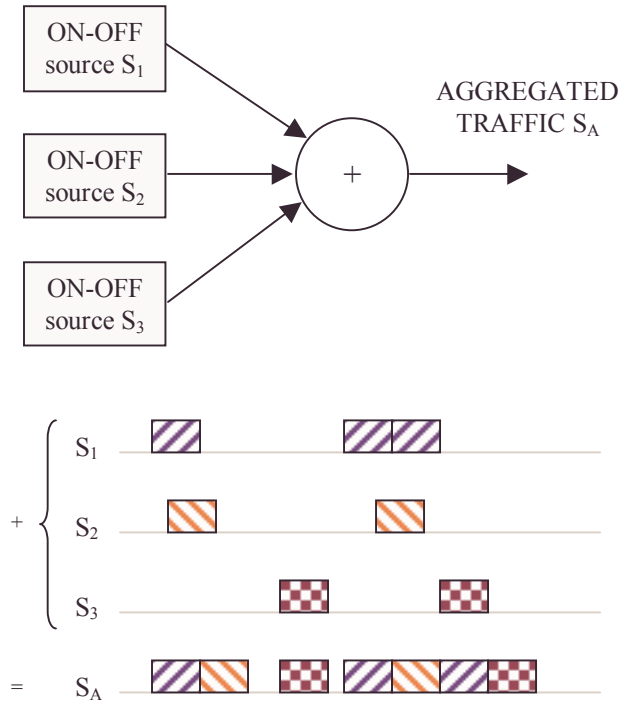


Figure 4.5. Aggregation process of three ON-OFF sources

Willinger [106, 109] demonstrated that Pareto distributions can be used to generate the distribution of the ON and OFF periods of each individual source. Moreover, it is desirable to generate traffic of a predefined load L as it was done previously for the Poisson model. Obviously, the resulting load L of the aggregated link is just the sum of the load L_i generated by each individual source i. Given M sources,

$$L = \sum_{i=1}^{M} L_i = M \times L_i . \tag{4.3}$$

It is considered that L_i is the same for each individual source, and it is simply equal to the mean duration of the ON period divided by the sum of the mean duration of the ON and OFF periods and

$$L_i = \frac{\mu_i^{ON}}{\mu_i^{ON} + \mu_i^{OFF}} . \tag{4.4}$$

The Pareto distribution has the following probability density function:

$$P(x) = \frac{\alpha\beta^{\alpha}}{x^{\alpha+1}}, \; x \geq \beta . \tag{4.5}$$

α is called the shape parameter or tail index, and β is the scaling parameter (the minimum value of x). Also, the mean value of the Pareto distribution is given by

$$\mu = \frac{\alpha\beta}{\alpha - 1}, \; \alpha \geq 1 \tag{4.6}$$

and therefore

$$\alpha = \frac{\mu}{\mu - \beta} . \tag{4.7}$$

The parameters for the ON and OFF period are respectively denoted α_{ON}, β_{ON}, α_{OFF}, and β_{OFF}. First, equation (4.4) is re-arranged in the following form

$$\mu_i^{OFF} = \frac{\mu_i^{ON}(1 - L_i)}{L_i} \tag{4.8}$$

and considering (4.3)

$$\mu_i^{OFF} = \frac{\mu_i^{ON}(M-L)}{L} \tag{4.9}$$

Finally, from equations (4.9) and (4.6), the scaling parameter of the OFF period can be expressed as

$$\beta_{OFF} = \frac{\alpha_{ON}\beta_{ON}(M-L)(\alpha_{OFF}-1)}{L\alpha_{OFF}(\alpha_{ON}-1)}. \tag{4.10}$$

Moreover, Willinger [109] defined the Hurst parameter of the aggregated traffic of identical ON-OFF sources as

$$H = \frac{3-\alpha_{ON}}{2},\ 1 < \alpha_{ON} \le 2 \tag{4.11}$$

when $\alpha_{ON} > \alpha_{OFF}$. Furthermore, α_{OFF} can be set to a fixed value [109].

Therefore, given a specific Hurst parameter (the degree of burstiness of the aggregated traffic) and given load *L*, all the parameters can be derived as follows.

$$\alpha_{ON} = 3 - 2H \tag{4.12}$$

$$\beta_{ON} = 12, \text{ (the duration of a packet, i.e. 12 μs)} \tag{4.13}$$

$$\alpha_{OFF} = 1.2, \text{ see reference [109]} \tag{4.14}$$

$$\beta_{OFF} = \frac{2 \times \alpha_{ON}(M - L)}{L(\alpha_{ON} - 1)} \tag{4.15}$$

The only parameter that cannot be derived is *M*, i.e. the number of individual ON-OFF sources. Moreover, the traffic generator must be evaluated in some way in order to validate its operation. The main parameter of interest is the Hurst parameter and, despite the fact that it is a known value when generating the traffic, one cannot assume that the generated traffic will exhibit the desired burstiness, mainly because the validity of the traffic is strongly dependent on the method used to actually implement the generator. The self-similarity must be validated and the validation method proposed by Willinger [109] is to check the variance of the generated traffic.

First, a number of definitions must be made.

$Y(t)$ is defined as the number of packet arrivals up to time t, and it is known as the packet arrival cumulative process.

X_t is defined as the incremental process of $Y(t)$, i.e. $X_t = Y(t+1) - Y(t)$.

$X_s^{(m)}$ is the aggregated process of X_t, i.e. $X_s^{(m)} = \frac{1}{m}[X_{sm-m+1} + X_{sm-m+2} + ... + X_{sm}]$.

From [109], the generated traffic is defined exactly self-similar if

$$X^{(m)} = m^{H-1} X \tag{4.16}$$

and therefore

$$Var(X^{(m)}) = m^{2(H-1)} Var(X) \tag{4.17}$$

and

$$\log\left[\frac{Var(X^{(m)})}{Var(X)}\right] = (2H-2)\times\log(m)\,. \tag{4.18}$$

Therefore the log-variance plot must have a slope of $2H-2$ with respect to $\log(m)$. In contrast, the log-variance plot of a Poisson incremental process has a slope of -1.

The log-variance-log(m) of our self-similar traffic generator is shown in Figure 4.6 for two different values of M (the number of aggregated ON-OFF sources). The Hurst parameter was $H = 0.8$ and thus the expected slope of both curves was –0.4.

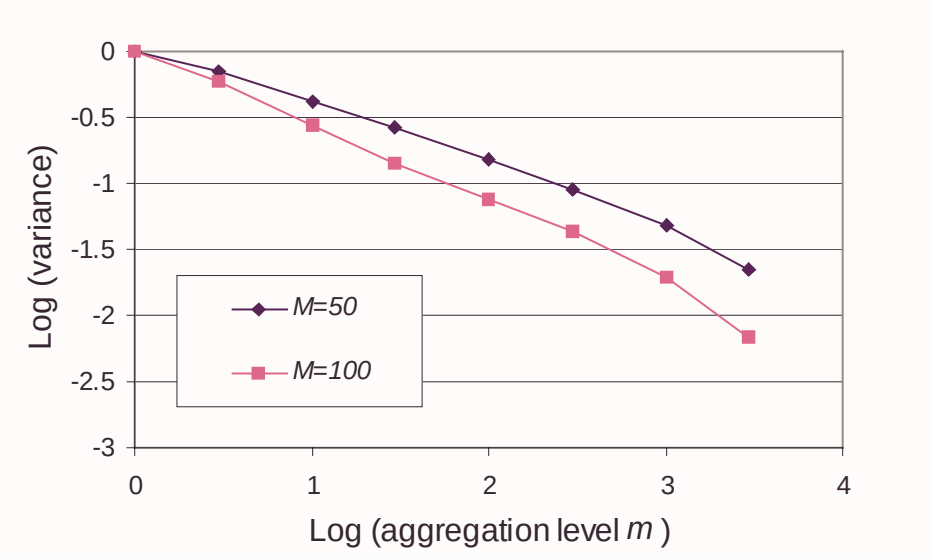


Figure 4.6. Log-variance-log(m)

It can be seen that for $M = 50$ the slope is equal to about -0.4 and thus the exhibited Hurst parameter is 0.8 as expected. However, with $M = 100$, the slope is equal to

about -0.55 and hence the exhibited Hurst is 0.725, i.e. less than expected. Therefore, the optimal number of ON-OFF sources was chosen as $M = 50$.

Moreover, Figure 4.7 shows the log-variance-log(m) plot for $M = 50$ and two different values of H. The Poisson traffic log-variance-log(m) plot is also shown in the same figure, and its slope is found to be close to –1 as expected.

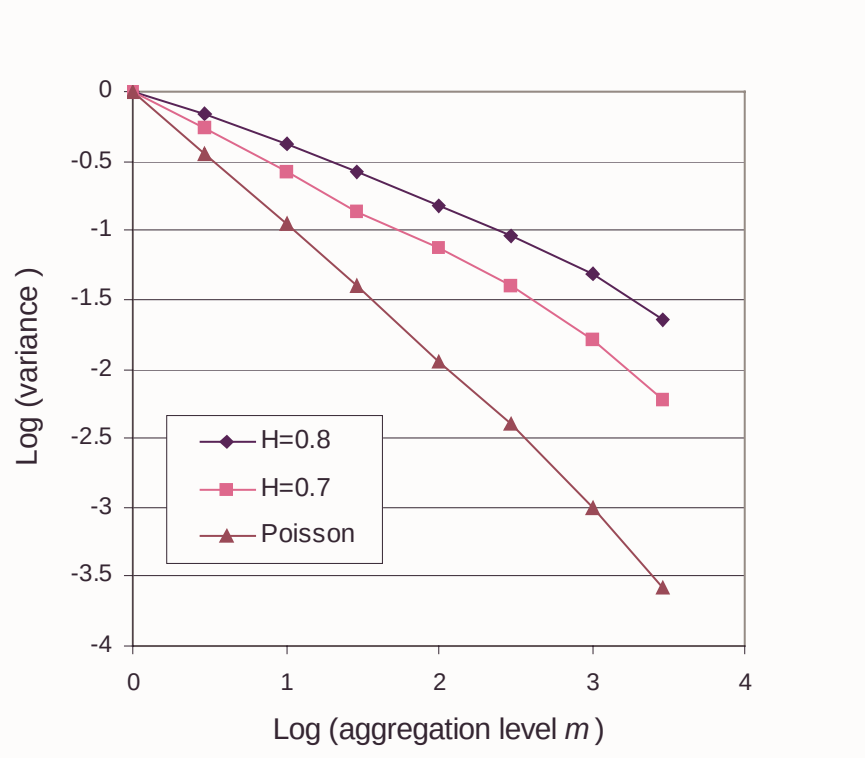


Figure 4.7. Log-variance-log(m) for self-similar and Poisson traffic

In the end, the Pareto distributed variates are generated using a similar technique to the one used for the Poisson traffic, i.e. an inverse transformation [103]. A uniform distribution U(0,1) is generated using a pseudo-random number generator and the pareto variates are computed as equal to $\frac{\beta}{u^{1/\alpha}}$. The C module generating the self-similar traffic can be found in Appendix 4.

Finally, a similar technique to the one used to plot Figure 4.4 was used to draw Figure 4.8. The left column represents the self-similar traffic and the right column corresponds to the Poisson traffic. The simulated link load in both cases was $L = 0.5$ and the Hurst parameter of the self-similar traffic was $H = 0.8$. As in Figure 4.4, it can be seen that the Poisson traffic burstiness decreases as the time scale increases, in contrast to the self-similar traffic which remains bursty over the three ranges.

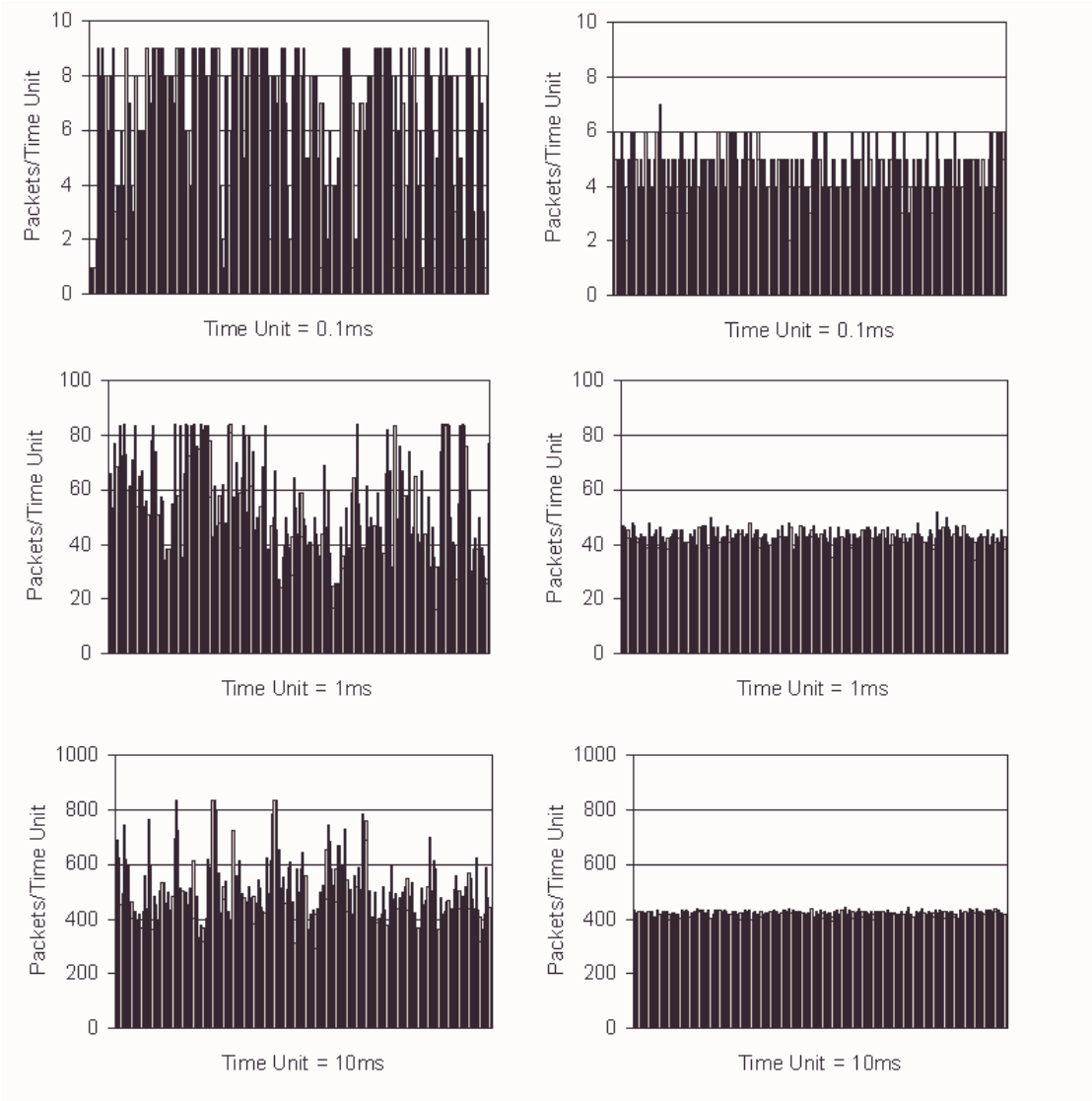


Figure 4.8. Generated self-similar and Poisson traffic

4.2.2.3 Implementation issues

The main restriction when generating both Poisson and self-similar traffic was that it could hardly be generated in real-time, i.e. during the course of a simulation. This was mainly due to a restriction when producing random numbers. In order for the network sources to be different, for each node's traffic a different seed was used to initialise the uniform distribution used to generate the exponential (Poisson) and Pareto (self-similar) variates. This could hardly be implemented in real-time in C.

To overcome this problem, each node's traffic was simply generated prior to the start of the simulation and stored in large files. Furthermore, when a simulation was started, a specific function from the DAC module was used to fetch the traffic into the nodes' transmission buffers. In fact, a temporary second buffer was used by the DAC to maintain time synchronisation, i.e. to make sure that packets were transferred from the file to the transmission buffers at the correct simulation time.

4.3 Summary

This chapter has presented the network simulator which has been specifically designed to simulate the WDM ring. The modular structure of the C program was introduced and some implementation issues were presented. The self-similar nature of data traffic in modern networks was also discussed and the two traffic models used in the simulator were described and their main features identified.

Chapter 5

Performance evaluation

5.1 Introduction

The simulator was used to evaluate the two architectures presented in Chapter 3 in order to validate the accuracy of the theoretical analysis. However, while the theoretical analysis gives an estimation of the bandwidth efficiency and the maximum throughput per node, the simulator provides much more results which give a more complete performance evaluation of the architecture.

It is also important to note that all simulations were ran for a time long enough to reach steady-state results. In general, between two and eight millions packets were transmitted per node for each simulation. Also, all traffic sources within a given simulation are statistically identical processes, i.e. the they differ at any instantaneous time but have identical long run average characteristics. Moreover, each node transmits to all other nodes in the network with an equal probability, i.e. $1/(N_T - 1)$ and the source-stripping scheme was first considered (results from destination-

stripping are presented in Chapter 6). Each simulation is characterised by its total normalised network load and by the traffic model that was used. The normalised network load L_N is defined as the ratio of the sum of all the nodes transmission rates to the total network transmission capacity, i.e. a normalised load of 1 means that the added nodes transmission rates reach the total transmission capacity of the network. The Hurst parameter used to generate self-similar traffic was equal to 0.8 which is a high degree of burstiness that is close to reality [109].

5.2 Performance evaluation

First, the theoretical predictions for the two networks considered are presented in Table 5.1 and are derived from equations (3.1) and (3.4). In both cases, the number of nodes was chosen such that the maximum throughput per node T_{MAX} was less than the nodes maximum rate of 1 Gb/s. In practice this is done by network operators as traffic peaks are statistically not expected to happen at the same time.

	S-16/4/2.5-Src	**S-64/4/10-Src**
Bandwidth efficiency η_s	0.8	0.94
$T_{MAX\text{-}S}$	500 Mb/s	588 Mb/s

Table 5.1 Theoretical predictions (source-stripping)

A specific notation is introduced to specify the removal scheme and the traffic model that are used. The suffix **Src** is used to specify that the source-stripping scheme is considered. A second suffix is used to define the traffic model, i.e. **P** for Poisson and

S for self-similar. For example, the notation **S-16/4/2.5-Src-P** means that the **S-16/4/2.5** architecture is considered with source-stripping and Poisson traffic.

5.2.1 Performance fairness

In all the simulations, an excellent fairness is achieved between the nodes that share a wavelength for transmission. The position of the nodes on the ring does not alter the results (i.e. wherever the nodes are situated, results are identical). The average throughput, queuing delay, buffer load, and packet dropping probability are very close (less than 1% difference with Poisson traffic and 5% with self-similar traffic) among all the nodes sharing a specific wavelength. This is achieved without any complex mechanisms involved as in previous proposed protocols [98-100]. Therefore, the restriction introduced in the design of the MAC protocol (slot-reuse) is a very simple scheme which is shown to perform in a highly satisfactory way even in the case of highly bursty traffic.

In the following sections, the average results among all the nodes within a simulation will be presented. Indeed, because of the very little performance difference among nodes, the average results significantly describe the individual nodes behaviour.

5.2.2 Throughput per node

The first set of results presented in Figure 5.1 is the average throughput per node.

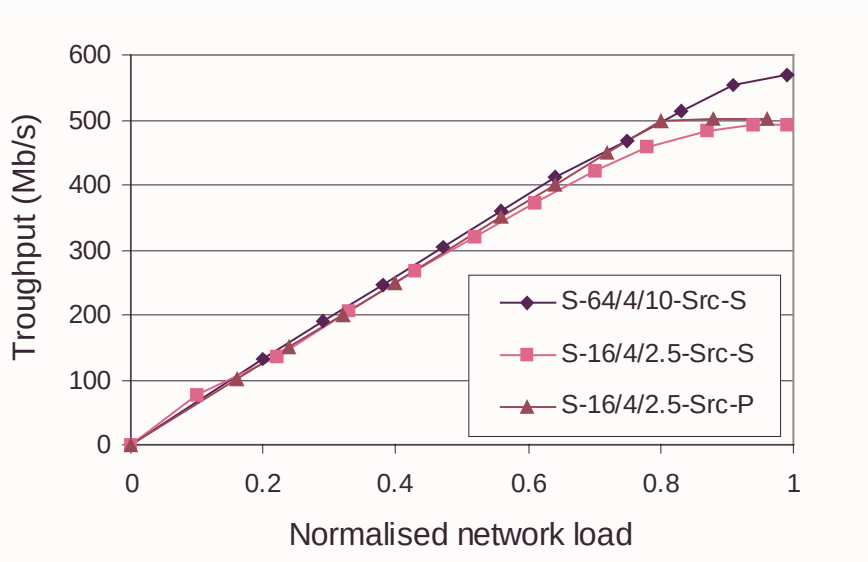


Figure 5.1. Average throughput per node

In both cases **S-16/4/2.5-Src-P** and **S-16/4/2.5-Src-S**, the maximum throughput achieved is 500 Mb/s and in the case **S-64/4/10-Src-S** the maximum value is 575 Mb/s. For both types of traffic, these values are very close to the theoretical predictions given in Table 5.1.

To evaluate the impact of the traffic models, both Poisson and self-similar traffic were used with the **S-16/4/2.5-Src** architecture. It can be seen that the use of self-similar traffic sources slightly reduces the achievable throughput when compared with the case where Poisson traffic sources are used. This is directly a consequence of the high degree of burstiness exhibited by self-similar traffic. Traffic peaks tend indeed to be very high and as the normalised network load increases, the probability of traffic peaks occurring at many nodes at the same time gets higher and thus the total traffic

may exceed the network capacity. In that case, the network gets congested and therefore the throughput is reduced.

In the **S-64/4/10-Src-S** simulation, the throughput is seen to be better than in the previous case (i.e. **S-16/4/2.5-Src-S**). This behaviour was expected and was referred to in the theoretical analysis (see Section 3.4.1) and it has been validated by the simulations.

5.2.3 Queuing-delay and transmission buffer load

These two set of results are presented in the same section because they have very similar properties and are closely linked. The average queuing-delay is presented in Figure 5.2 and the average transmission buffer load is shown in Figure 5.3.

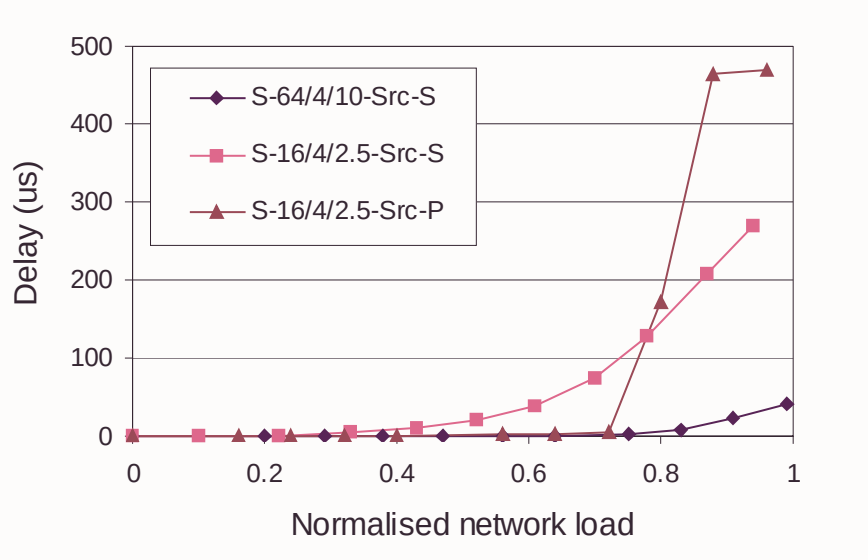


Figure 5.2. Average queuing-delay

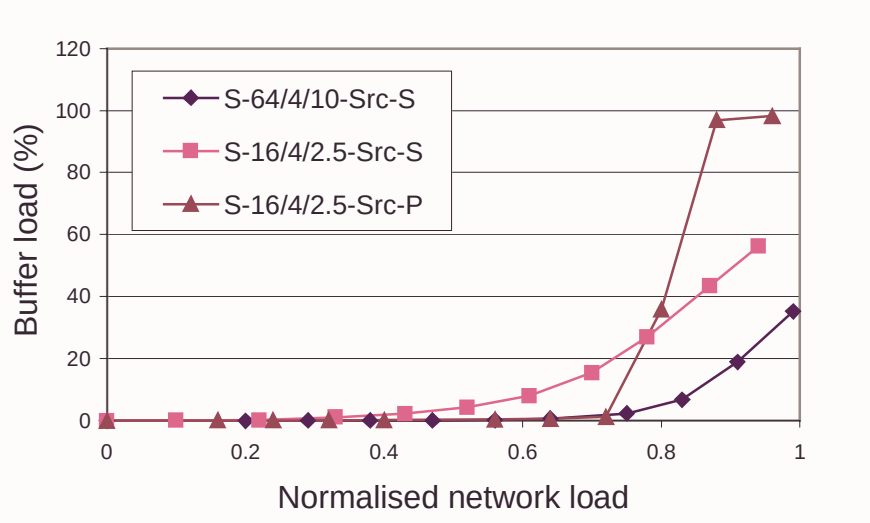


Figure 5.3. Average transmission buffer load

As in the previous section, the influence of the traffic models was evaluated by using both Poisson and self-similar traffic sources with the **S-16/4/2.5-Src** network. For both the queuing delay and the buffer load, the use of self-similar traffic gives worse performance than with Poisson traffic when L_N is below 0.8. Above this value, the performance under self-similar traffic is much better than in the case where Poisson traffic is applied to the network. This is consistent with the nature of Poisson traffic whose transmission rate, at high loads, becomes more constant as the packet inter-arrival duration decreases (see Fig. 4.8). When the maximum transmission capacity of the network is reached (i.e. $L_N = 0.8$), the congestion state is immediately deteriorated by the almost non-variable, constant arrival of packets and the nodes transmission buffer are persistently filled with packets. As a direct result, the queuing-delay also increases abruptly when the congestion state is reached. On the other hand, with self-similar traffic, packets bursts tend to frequently increase the buffer loads and the average value is seen to be higher than with Poisson traffic when $L_N < 0.8$. However, as the maximum transmission capacity of the network is reached, the

buffers keep being filled in an intermittent way and the congested state does not induce an abrupt decline in networking performance.

Also note in both cases that, increasing the number of nodes and wavelength rate by a factor of 4 (i.e. upgrading the network from **S-16/4/2.5-Src-S** to **S-64/4/10-Src-S**) results in both lower queuing-delay and buffer load. This is understood by observing that a larger number of nodes leads to earlier slot re-use (not release), as the distance between nodes decreases, and hence the performance is improved.

More generally, the average queuing-delay is very low as it is always below 500 μs, a more than suitable value even in the case of real-time multimedia traffic. Moreover, with the more realistic case of self-similar traffic, the queuing-delay is always below 300 μs in the case **S-16/4/2.5-Src-S**, and below 50 μs in the **S-64/4/10-Src-S** architecture.

5.2.4 Packet dropping probability

Packet dropping is a direct consequence of the limited storage capacity of the transmission buffers (i.e. 100 packets). As a result, when a buffer is full, packets that arrive from the access network to be transmitted onto the ring are simply dropped as they simply cannot be added to the buffer. This dropping mechanism has been implemented in various networks such as ATM (Asynchronous Transfer Mode) and Frame-Relay because, in the event of packets losses, recovering capabilities are expected to happen at higher layers (e.g. TCP/IP) and moreover, the implementation costs of such networks are also reduced.

The average packet dropping probability was measured in the three architectures already considered in the previous sections and it is shown in Figure 5.4.

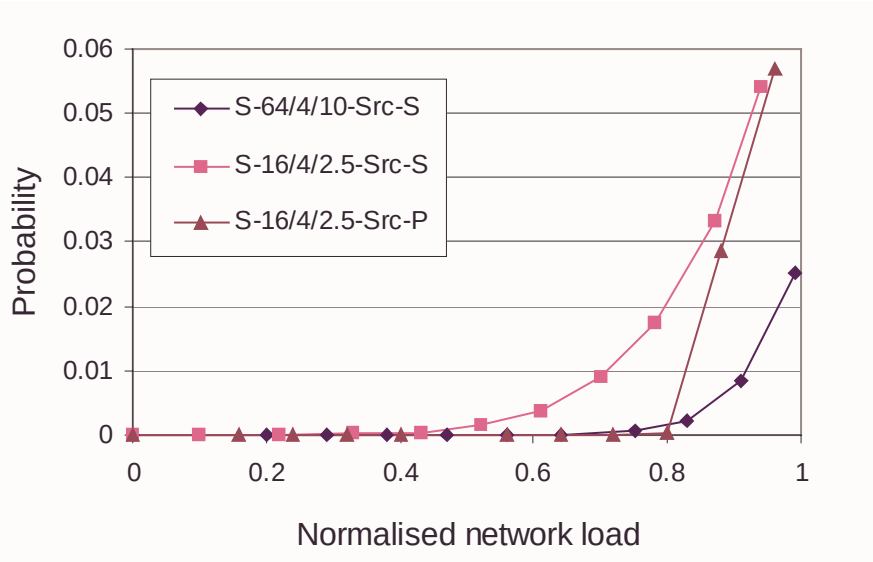


Figure 5.4. Average packet dropping probability

In fact, the packet dropping probability is directly proportional to the load on the transmission buffers. When buffers are almost empty, the packet dropping probability is evidently low, but as the buffer load increases the packet dropping probability subsequently gets worse. This is striking in the case **S-16/4/2.5-Src-P**, when the very low average packet dropping probability rapidly increases when the normalised network load exceeds 0.8. This directly corresponds to the also rapid increase of the average transmission buffer load as seen in Figure 5.3. Furthermore and with self-similar traffic, the packet dropping probability is still determined by the corresponding buffer load.

It can be seen that the average packet dropping probability reaches 0.01 (1% of packets are dropped) for $L_N = 0.7$ in the case **S-16/4/2.5-Src-S** and it reaches 0.01 for $L_N = 0.9$ in the **case S-64/4/10-Src-S**. These values are unfortunately not acceptable in a metropolitan environment because the high dropping probability will trigger a large number of packet re-transmissions which will add-up to the ordinary traffic and consequently reduce the networking performance. However, it is shown in the next chapter how the use of destination-stripping greatly enhances the performance.

5.3 Summary

This chapter has presented the results obtained with the network simulator with source-stripping. The network throughput performance was compared with the analytical predictions derived in Chapter 3. Furthermore, a complete set of simulation results was presented and analysed. The impact of using different traffic sources was discussed and the performance of two different architectures were also compared.

Chapter 6

Implementation issues

6.1 Introduction

In the previous chapter, the throughput performance was seen to be quite close to the theoretical predictions derived in Chapter 3. Moreover, the average packet queuing–delay was very satisfactory as it never exceeded 500 μs with Poisson traffic and was always below 300 μs with self-similar traffic sources. On the other hand, the average packet dropping probability was found to be unsuitable for a metropolitan networking environment.

However, these results were considered with source-stripping and the theoretical analysis demonstrated that performance can greatly be improved by implementing the destination-stripping scheme, as this will be shown in the next section.

Furthermore, the physical node architecture was designed as a FT-FR^4 system. While the main advantage of this architecture is that it avoids receiver collisions, its

flexibility is limited because it considers a fixed number of wavelengths and adding new wavelengths would induce a hardware modification of all the network nodes. The use of a tuneable receiver was therefore considered as it increases the flexibility of the network but, on the other hand, it introduces receiver collisions and a specific mechanism had to be used to overcome this problem. This is also presented latter in this chapter.

Finally, the simulation results were restrictive in the sense that all traffic sources were statistically identical. While this can be assumed in simulations, it is usually not the case in a real networking environment where nodes may transmit at completely different rates. The use of unbalanced traffic loads was therefore considered and this will also be presented in this chapter.

6.2 Destination-stripping

In order to be able to compare the performance improvement induced by using the destination-stripping scheme, the **S-16/4/2.5** architecture was simulated with exactly the same self-similar traffic sources that were used with source-stripping. With destination-stripping, this architecture is denoted **S-16/4/2.5-Dest-S**.

6.2.1 Node throughput

The theoretical predictions given in Chapter 3 are presented in Table 6.1. The predictions for the **S-64/4/10-Dest** architecture are also given but this network was not simulated.

	S-16/4/2.5-Dest	S-64/4/10-Dest
Bandwidth efficiency η_d	1.33	1.77
$T_{MAX\text{-}D}$	833 Mb/s	1.11 Gb/s

Table 6.1 Theoretical predictions (destination-stripping)

In both cases it can be seen that, because destination-stripping induces slot-reuse, the bandwidth efficiency exceeds 1 and the maximum throughput per node is greatly increased when compared with source-stripping.

The average throughput per node is shown in Figure 6.1 for the two considered architectures, i.e. **S-16/4/2.5-Src-S** and **S-16/4/2.5-Dest-S**.

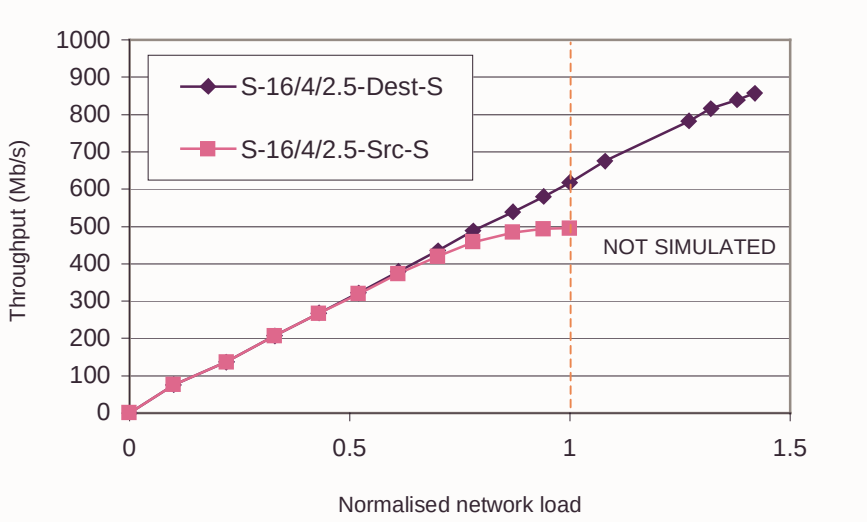


Figure 6.1. Average throughput per node (2)

The architecture **S-16/4/2.5-Src-S** was not simulated with $L_N > 1$ as the throughput threshold is reached when $L_N = 0.8$. With destination-stripping, the throughput is seen to be better than source-stripping when the normalised network load is above

0.6. Moreover, it constantly increases and reaches 620 Mb/s for $L_N = 1$ but, due to slot-reuse, the throughput can be increased to reach up a maximum of 860 Mb/s and in that case $L_N = 1.42$. Actually, the performance is seen to slightly outperform the theoretical predictions given in Table 6.1 which assumed that all nodes were sending packets to other nodes with equal probabilities and in practice there will always be a little difference due to the random number generator accuracy.

6.2.2 Queuing-delay and transmission buffer load

The average queuing-delay is presented in Figure 6.2 and the average transmission buffer load is shown in Figure 6.3 and in both cases the maximum normalised network load considered was 1.

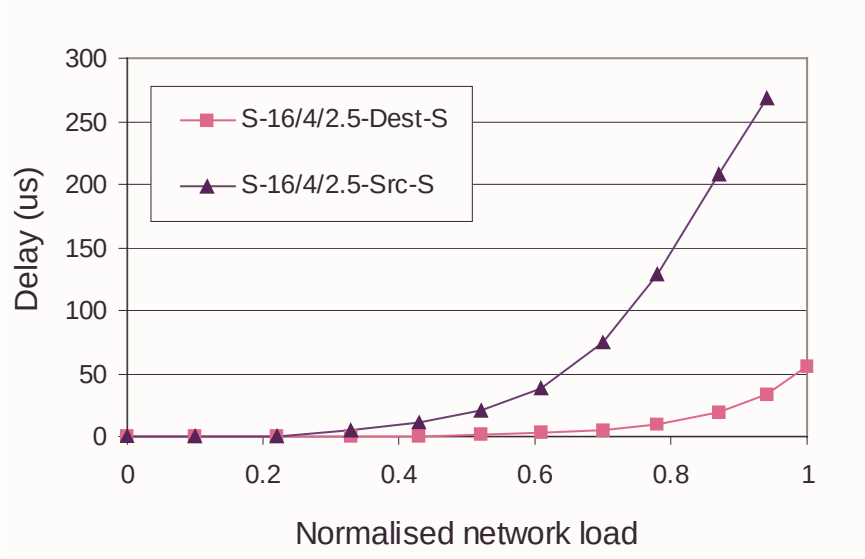


Figure 6.2. Average queuing-delay (2)

When the destination-stripping scheme is used, the queuing-delay is seen to be between five to eight times better than with the source-stripping scheme and it only

reaches 55 µs when $L_N = 1$. This is a direct consequence of using destination-stripping as slots are freed and re-used earlier than with source-stripping and as a consequence the queuing-delay is greatly reduced. Moreover, an interesting result which is not represented on Figure 6.2 is that the queuing-delay only reached 150 µs when the normalised network load was equal to 1.42 and the maximum node throughput was achieved. This is indeed inferior to the maximum queuing-delay experienced with source-stripping when $L_N = 1$.

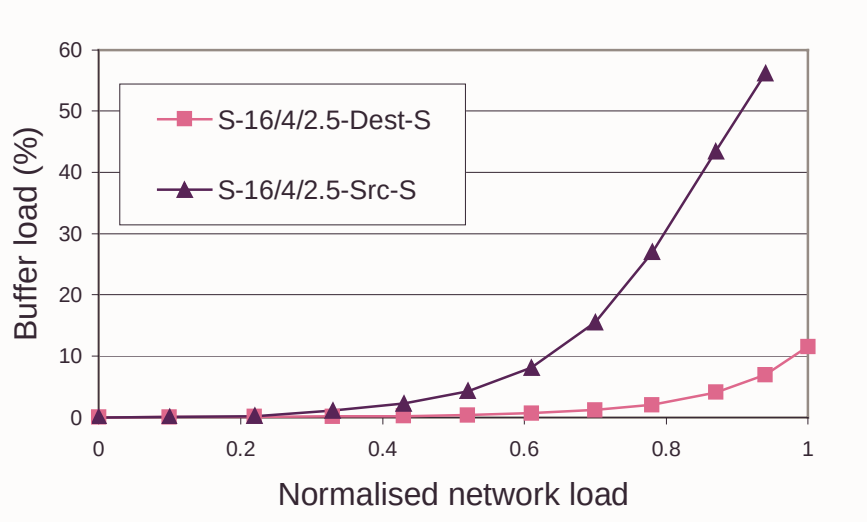


Figure 6.3. Average transmission buffer load (2)

The benefit of using destination-stripping is also directly visible on Figure 6.3. Indeed, the reduced queuing-delay induced by slot-reuse also implies a reduction in the average buffer load which is also seen to be between five and eight times lower than with source-stripping. In fact, the main advantage of having a reduced buffer load is that the packet dropping probability will be significantly improved.

6.2.3 Packet dropping probability

The average packet dropping probability is shown in Figure 6.4. As a direct result of the reduced average buffer load, the packet dropping probability is seen to be much smaller with destination-stripping.

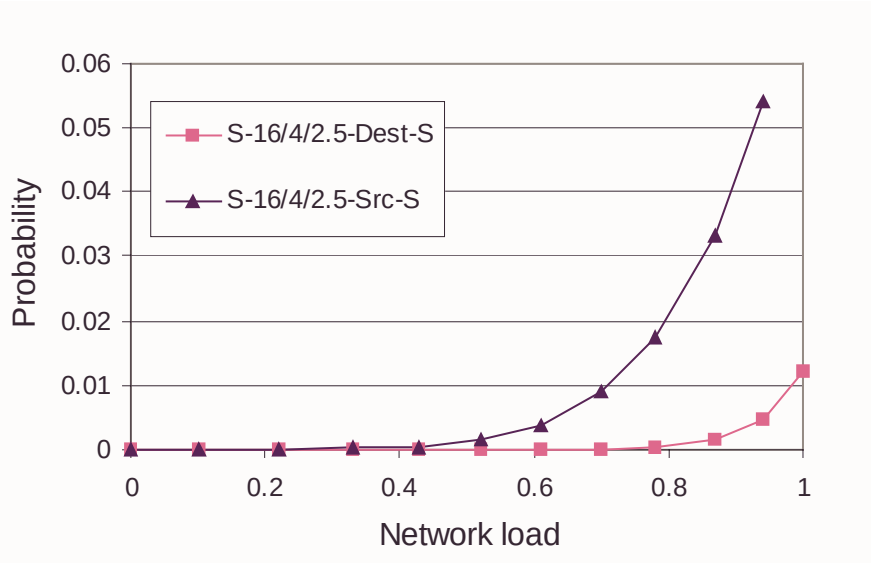


Figure 6.4. Average packet dropping probability (2)

Actually, with destination-stripping, the dropping probability reaches 0.01 (1% of packets are dropped) for a normalised network load just inferior to 1 and it is equal to 4×10^{-4} for $L_N = 0.8$. In the latter case, it means that, on average, 1 packet out of 2500 is dropped. This a much better figure than with source-stripping. Moreover, with $L_N = 0.7$ the packet dropping probability was null during the entire course of the destination-stripping simulation.

6.2.4 Performance comparison

Compared with the results in [100] where a different and more complex MAC protocol is used, the proposed implementation achieves a better performance with a much simpler protocol. The proposed implementation can achieve a $T_{MAX\text{-}S}$ of 8.88 Gb/s with a bandwidth efficiency (η) of 0.89 when $R_W = 80$ Gb/s, $N = 8$ and only one wavelength is used. In [100] (Poisson traffic) and under comparable constraints, $T_{MAX\text{-}S}$ is 1.7 Gb/s and $\eta = 0.18$. Also in [100] with $R_W = 40$ Gb/s, two wavelengths and $N = 8$, $T_{MAX\text{-}S}$ is 4.8 Gb/s and $\eta = 0.55$. Our implementation achieves a maximum throughput of 8 Gb/s and $\eta = 0.8$, with $R_W = 40$ Gb/s and two wavelengths.

6.3 Performance of the FT-TR architecture

As already stated in the introduction of this chapter, the physical node architecture was designed as a FT-FR^4 system. While this architecture avoids receiver collisions, its flexibility is limited because the number of wavelengths is fixed and that reduces the scalability of the network as adding more wavelengths implies a hardware upgrade of all nodes.

To overcome this problem, the fixed receivers can be replaced by an adequate tuneable receiver (TR) with a large tuning range (e.g. 16 wavelengths). Hence, a network upgrade would only imply upgrading the software configuration of the access nodes. While implementing a tuneable receiver will increase the cost of the network

infrastructure, it will increase the scalability of the network and reduce the cost of future inevitable upgrades.

However, the use of a tuneable receiver introduces receiver collisions. A node may indeed receive more than one packet at the same time on different wavelengths. Therefore, a mechanism must be introduced to handle receiver collisions. Solving the problem by implementing a more complex and costly hardware architecture is not desirable in our proposed architecture because increasing the hardware complexity is contrary to our approach.

We thus propose a simple solution whereby a packet is received (assuming that a suitable selection scheme is implemented) and all other colliding/conflicting packets are allowed to do another loop round the ring. In the case of a simple collision, one packet will simply do one more loop and will come back to the destination node and will eventually be received without collision. This mechanism is clearly only suitable with destination-stripping. It may seem as a counter-intuitive scheme but implementing this collision avoidance (CA) mechanism will not affect the average packet transmission delay in a significant manner as the ring latency is very low (691.2 μs). Furthermore, packets may arrive out of sequence but this does not matter in data traffic as higher level layers (TCP/IP) will re-sequence the data.

The FT-TR-CA (FT-TR with collision avoidance) architecture was simulated for the **S-16/4/2.5-Dest-S** network. To evaluate the impact of implementing this scheme, the number of receiver collisions were measured for both the FT-FR^4 and FT-TR-CA architectures. In fact, with the FT-FR^4 architecture, what is called a receiver collision

is simply a case where a node receives more than one packet concurrently but, because in this architecture there is a dedicated fixed receiver for each wavelength, each packet would be received by a separate receiver. We will still consider this event to be a receiver collision but in that specific case (FT-FR^4), the architecture has inherent capabilities to overcome this situation.

Because there are only 4 wavelengths, three possible collisions may occur, i.e. a node may receive 2 to 4 packets at the same time. Figure 6.5 shows the number of simple collisions (2 packets simultaneously received) which occurred among all the nodes within the network. The number of collisions is given as a percentage of all the packet receptions that happened during the course of the simulations. Furthermore, Figure 6.6 shows the number of double collisions (3 packets concurrently received). It is also worth mentioning that, due to their very low occurrence, triple collisions are not considered.

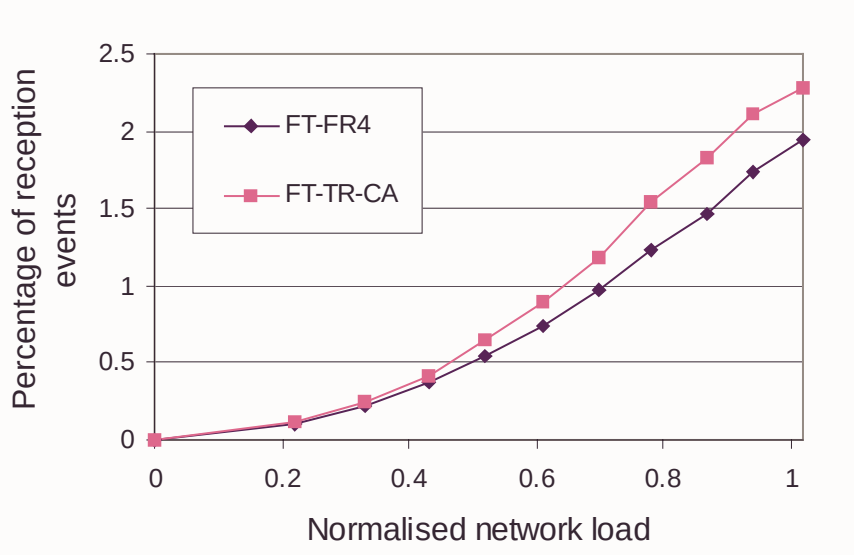


Figure 6.5. Simple receiver collisions

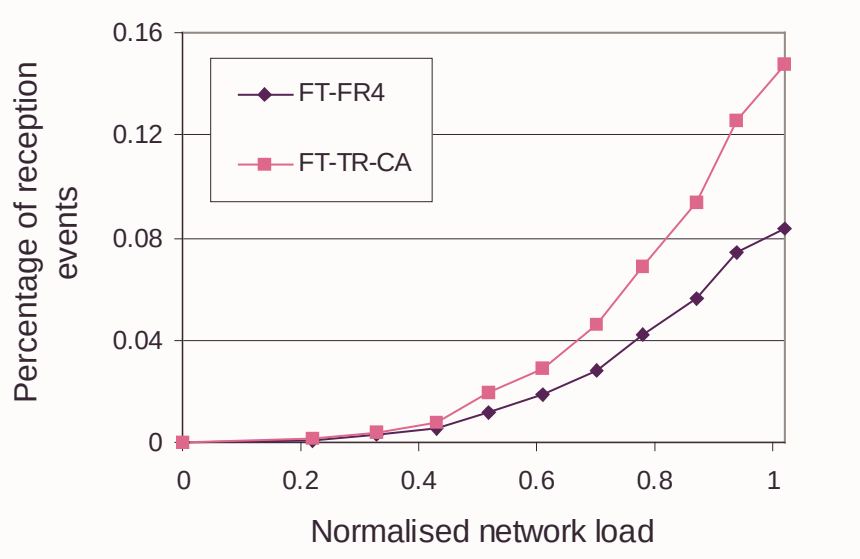


Figure 6.6. Double receiver collisions

It can be seen on both figures that the collision avoidance mechanism induces more collisions but there is very little difference compared to the FT-FR^4 implementation. This is due to the amount of colliding packets (small number) that are allowed to re-circulate around the ring. These packets are added-up to the normal traffic and they induce more receiver collisions as the total network traffic is slightly increased.

For the FT-FR^4, simple receiver collisions are seen to reach a maximum of about 2 % of the total reception events. At the same network load (i.e. $L_N = 1$), the CA mechanism induced more collisions but they only reached up 2.3 %. In the mean time and for $L_N = 1$, double collisions reached 0.08 % for the FT-FR^4 architecture, and 0.15 % with the FT-TR-CA network. In general, the collision avoidance mechanism does not induce a large additional number of receiver collisions when compared to the FT-FR^4 architecture.

Moreover, and through further simulations that implemented our CA mechanism, it was concluded that the previous performance achieved with the FT-FR^4 architecture (throughput, queuing delay, buffer load, and packet dropping) is not affected by the CA mechanism. Namely that the performance of FT-FR^4 and FT-TR-CA are almost identical with respect to these parameters. Because receiver collisions are a small percentage of the total number of reception events, the colliding packets which are allowed to re-circulate around the ring do not significantly increase the overall network load and as a result the performance is not affected. For example, for $L_N = 1$, simple collisions represent 2.3 % of the total number of reception events. But in that case, one packet is correctly received and the other is not removed from the ring, and consequently the normalised network load will be increased by $2.3/2 = 1.15$ % and will only reach $L_N = 1.01$.

The collision avoidance mechanism has, however, an influence on the packet transmission delay, i.e. the total amount of time it takes for a packet to be transmitted from one access network to the intended destination. If the delay introduced at intermediate nodes is considered to be null, the transmission delay is simply the sum of the queuing-delay and the propagation delay (which is proportional to the physical distance between the transmitting and receiving access nodes).

Figure 6.7 shows the average packet transmission delay with both FT-FR^4 and FT-TR-CA architectures (both with destination-stripping).

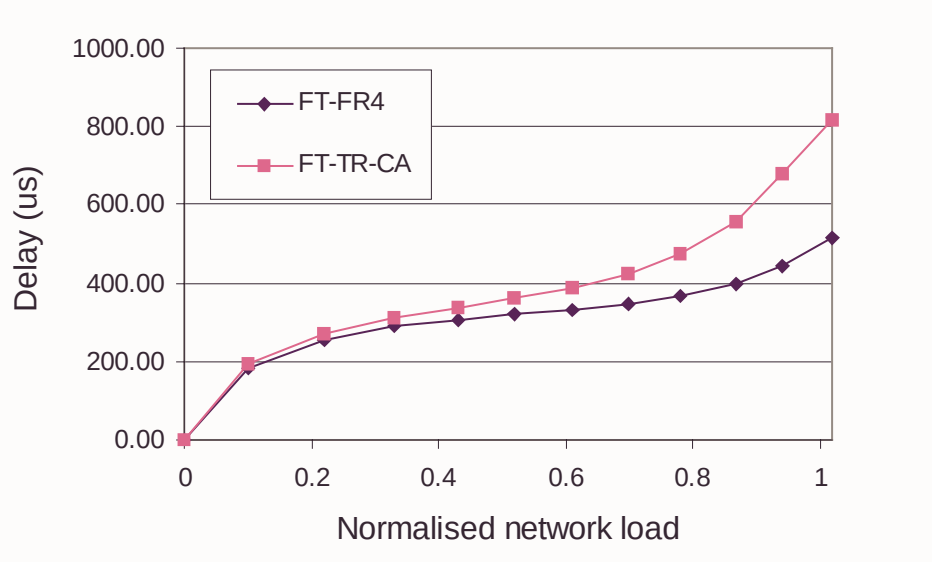


Figure 6.7. Average packet transmission delay

In general, the transmission delay is increased by the collision avoidance mechanism. This is because colliding packets which are allowed to re-circulate around the ring will have a much larger transmission delay when compared with packets that are correctly received. The minimum added delay will indeed be the duration of a complete ring rotation (i.e. 691.2 µs) when a re-circulating packet is correctly received after one extra rotation.

However, the transmission delay is seen to be very acceptable in both architectures. Indeed, in the case where the normalised network load is slightly above 1, the transmission delay reaches 800 µs for the FT-TR-CA and 500 µs for FT-FR^4 network. In the presence of real-time multimedia traffic, the end-to-end user delay (transmission delay + processing time) must usually be below 100 ms [110] and both the FT-FR^4 and FT-TR-CA architectures would be suitable to carry such traffic.

6.4 The effect of unbalanced traffic

So far, all the results presented were obtained when using statistically identical traffic sources at each node. Specifically, the average traffic rate was the same among all the nodes of a specific simulation. Moreover, the efficiency of the fairness mechanism introduced in the design of the MAC protocol was found to be very good even in the presence of highly bursty self-similar sources. However, the assumption that all traffic is symmetrical does not fit with the real networking environment where nodes may transmit at completely different rates.

In this section we consider the case **S-16/4/2.5-Dest-S** with the FT-TR-CA architecture. For each node, the traffic rate is randomly and uniformly chosen between 100 Mb/s and 1 Gb/s. A set of 50 different simulations whose network load was inferior or equal to 1 were studied. Over this set of simulations, the average normalised network load was found to be $L_N = 0.87$. As there are 16 nodes in each simulation, we obtained performance results for $16 \times 50 = 800$ nodes. In order to study the effect of unbalanced traffic, these results were sorted according to the average transmission rate of each node. Nine categories were considered to group the results. Each category is defined as follows. The *i*th category regroups results from nodes whose incoming transmission (from access side) rate was between $i \times 100$ Mb/s and $(i+1) \times 100$ Mb/s. For example, the first category regroups results from nodes whose incoming transmission rate was between 100 Mb/s and 200 Mb/s. In each category, the average throughput per node, the average queuing delay and the average packet dropping probability were computed. These are presented in Figure 6.8, Figure 6.9 and Figure 6.10 respectively.

Figure 6.8 presents the average throughput per node per category as well as the traffic that was received by that node from its access network (applied traffic) and expected to be transferred onto the WDM ring.

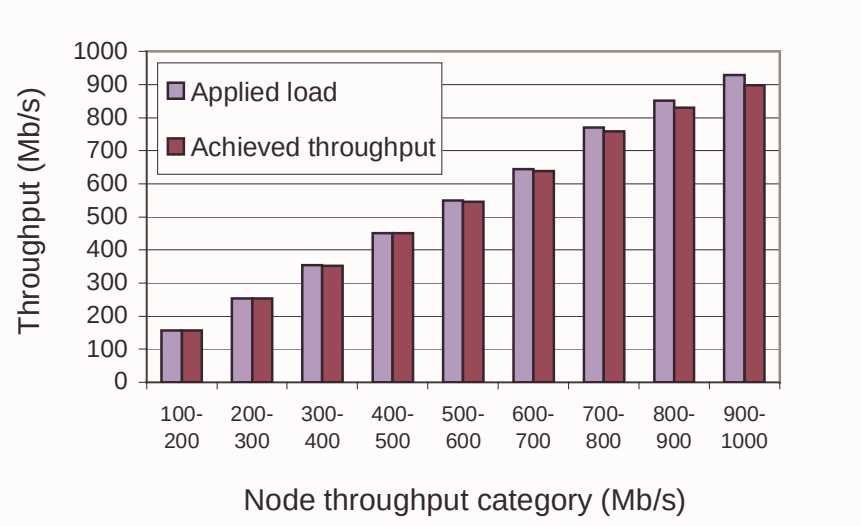


Figure 6.8. Average node throughput per category

In the first category where nodes do not transmit at very high rates (100 Mb/s – 200 Mb/s), the achieved throughput (157.03 Mb/s) is extremely close to the applied load.

For the same category, the average queuing delay (see Figure 6.9) is found to be very low (2.63 μs) and below the duration of a time slot (4.8 μs). This means that these nodes have been able to transmit almost immediately after they had packets in their transmission buffers.

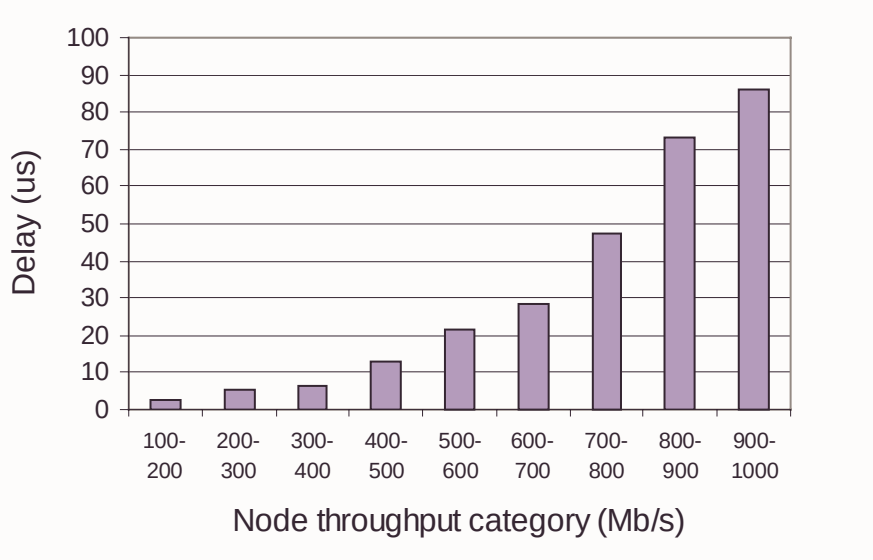


Figure 6.9. Average queuing-delay per category

As a result, their packet dropping probability (see Figure 6.10) is found to be very low (1.1×10^{-4}). In contrast, the ninth category presents a worse performance. The achieved throughput (897.14 Mb/s) is inferior to the applied load (928.03 Mb/s), the queuing delay is 86.17 μs and the packet dropping probability reaches 0.018. As seen on Figure 6.9 and Figure 6.10, the performance of the other categories gradually degrades and does not show an abrupt deterioration.

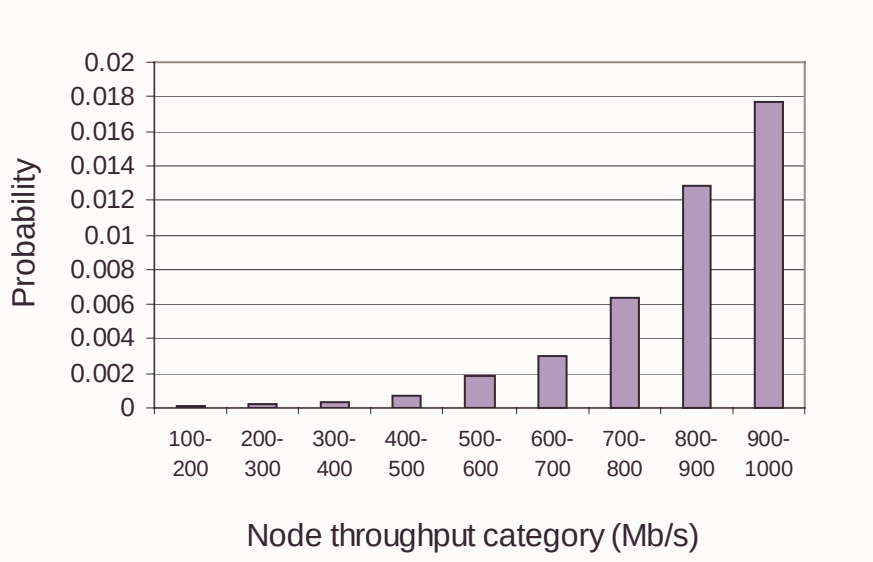


Figure 6.10. Average packet dropping probability per category

These observations can be explained as follows. The network traffic destinations are random and uniformly distributed and destination stripping is used. This implies that the released slots will also be uniformly distributed round the ring, with each node releasing

$$S_{MIN} = \frac{L_N \times R_W \times N_W}{N_T \times S} \tag{6.1}$$

slots per second on average, amounting to an average bit rate of

$$R_{MIN} = \frac{L_N \times R_W \times N_W}{N_T}. \tag{6.2}$$

In a given network (simulation), nodes are mixed in terms of applied load (incoming access bit rate) and each has access to the slots released by the immediate upstream neighbour in addition to any left-over from further up the stream. Therefore, if a node's average applied load is below R_{MIN}, then the achieved throughput, as Figure

6.7 shows, is almost identical to the applied load. On the other hand, if the applied load is above R_{MIN}, then nodes are able to achieve a throughput equal to R_{MIN} plus a rate that depends on the left-over slots from upstream and any unused network capacity (cases with $L_N < 1$). In Figure 6.7, $R_{MIN} = 544$ Mb/s, and the throughput performance is not affected below R_{MIN} as expected. Moreover, the throughput is not significantly affected above R_{MIN} in our scheme. Results for queuing delay and packet dropping probability follow and can be explained on the basis of the observations made.

In general, these results are of the same order compared to the results obtained in previous chapters for a similar normalised network load (0.87). The overall performance is not greatly affected by asymmetrical traffic. However, it is very interesting to note that nodes transmitting at low rates (categories 1-5) are less affected than those transmitting at higher rates (categories 6-9). This is a very attractive feature. Indeed, *resource-hungry* nodes are not able to starve *low-transmitting* nodes as one might have expected. This means that nodes transmitting at high rates can use all of the available bandwidth if they desire, without preventing other nodes from transmitting at smaller rates. The simple fairness mechanism of the MAC protocol is thus found to perform in a satisfactory way.

6.5 Summary

This chapter has presented the simulation results of the ring network with destination-stripping. The performance was compared with the results from the previous chapter where source-stripping was considered. A modified node architecture was also

proposed and the effects of receiver collisions on the network performance were presented and evaluated. Finally, the consequences of using unbalanced traffic sources were described and analysed.

Chapter 7

Conclusions

This project has considered a WDM ring network architecture intended to serve as a metropolitan access network. The research has mainly focused on the network scalability and flexibility, two main implementation issues for telecommunications operators when deploying a metropolitan area network. The different architectures and protocols which have been studied in the past for WDM networks were presented in Chapter 2. Moreover, WDM ring networks and protocols proposed recently were also described and their main features identified.

In Chapter 3, the WDM ring architecture considered was presented and the physical parameters of the ring have carefully been described. Furthermore, the access node architecture has been detailed and some design issues were discussed. The medium-access control protocol was presented and analytical predictions of the expected performance have been derived. These have shown that the network performance (i.e. the bandwidth efficiency and the maximum throughput per node) was very promising

and that it was also expected to be greatly improved by the use of destination-stripping.

A network simulator was specifically designed to simulate the performance of the proposed architecture. The main implementation issues were presented in Chapter 4, alongside with the modular structure that has been employed to organise and facilitate the development of the simulator. The self-similar nature of the traffic in modern data networks has also been introduced and two traffic models have been developed in order to generate the network traffic applied to the simulator. In particular, a self-similar traffic generator was designed and its operation has been validated by measuring and analysing the variance of the generated traffic. The optimal number of aggregated ON-OFF sources has thus been derived and the self-similar behaviour exhibited by the generated traffic has been shown to be very good.

The theoretical predictions derived in Chapter 3 were validated by simulations carried out in Chapter 5 and 6. In both chapters, the impact of using self-similar traffic sources has been discussed and two similar architectures have been compared in order to evaluate the scalability of the proposed network and protocol. Moreover, a complete set of performance results have been presented and they have shown that the network operation was significantly improved by the use of destination-stripping. Moreover, the scalability of the network has been shown to be very attractive as performance also improved when the number of access nodes and the wavelengths rate were proportionally increased.

Furthermore, a modified node architecture has been introduced in Chapter 6. Specifically, the use of a tuneable receiver has increased the flexibility of the network but it has also introduced the possibility of receiver collisions. The frequency and the impact of receiver collisions have therefore been evaluated and it has been shown that collisions have very little effect on the network performance. Finally, unbalanced traffic sources have been implemented in order to analyse the consequence of having different transmission rates among the ring. This experiment has shown that, with destination-stripping, a minimal rate could be achieved by any node and that it was a direct consequence of the simple fairness mechanism introduced in the MAC protocol design.

Chapter 8

Future Work

The work carried out to date has resulted in a number of original contributions to the considered research area. However, there are several potential areas where further work and research could be done. This chapter outlines some possible areas where further work could be carried out.

The traffic sources generated packets whose fixed-size was equal to the length of a slot. While this had been assumed in order to simplify the implementation of both the traffic sources and the simulator, it does not correspond to real networking conditions where the packet size is variable. In order for the network to handle variable packet sizes, two approaches could be considered. First, if the slot size is to remain unchanged, a mechanism must be implemented at each node in order to efficiently fill the slots. A separate queue can be implemented for each destination, and packets could, without altering the arrival order, be simply piggybacked onto the slots. However, a considerable loss of bandwidth efficiency can be induced if slots are not filled optimally. A second alternative is to significantly reduce the size of the slots. In

this case, packets could be sliced in small pieces that would fit into a slot and only the last slice could be smaller that a slot and the bandwidth loss can be reduced. However, reducing the size of the slots would increase the hardware complexity as the duration of a slot would be greatly reduced. Consideration can also be given to variable slot size and random access.

Another limitation of the traffic sources is that each node was assumed to send packets to all other nodes with an equal probability. While this hypothesis was acceptable in order to evaluate the transmission capacities of the network, it does not fit with a real networking environment where traffic can be transmitted randomly among nodes. Nevertheless, it might be interesting to simulate the particular case where a node is the preferred destination of most of the packets. Such a node can for example be the node that links the metropolitan network to the WAN backbone, and a particular architecture might be required for that node to handle such a large amount of traffic. For example, if we consider a network with 4 wavelengths, a FT^4-FR^4 could be considered in order to allow simultaneous transmissions and concurrent receptions.

Also, the proposed architecture does not provide any QoS mechanisms. While it may be considered that implementing complex techniques could quickly become too costly for a metropolitan network, simple queues with a limited number of priorities could easily be implemented. Packets from a high priority queue could be transmitted first while packets from a best-effort queue would be only served on a minimal basis. Moreover, with source-stripping a simple mechanism could be implemented to allow nodes to reserve (i.e. by not releasing them) a small amount of slots for critical traffic. With destination-stripping, a reservation mechanism could also be implemented,

maybe by using a specific sub-carrier tone to declare that a slot is being reserved. Both traffic reservation schemes could simply be regulated by the contract defined between the carrier and the customers.

Finally, a second counter-rotating fibre can be considered. This can obviously introduce a protection mechanism against link failures, but it can also be used to improve transmission fairness among nodes. Moreover, it would double the transmission capacity and undoubtedly improve the network performance.

References

[1] C. Brackett, "Dense wavelength division multiplexing networks: Principles and applications," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 6, pp. 948-964, Aug. 1990.

[2] R. Ramaswami, "Multiwavelength lightwave networks for computer communication," *IEEE Communications*, vol. 31, no. 2, pp. 78-88, Feb. 1993.

[3] R. Cruz *et al*. (Eds), *Feature Topic*, "Special issue on optical networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, June 1996.

[4] J. Elmirghani, and H. Mouftah, "Technologies and architectures for scalable dynamic dense WDM networks," *IEEE Communications Magazine*, vol. 38, no. 2, pp. 58-66, Feb. 2000.

[5] P. Green, Jr., "Optical networking update," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 764-779, June 1996.

[6] B. Mukherjee, "WDM-based local lightwave networks – Part I: Single-hop systems," *IEEE Network*, vol. 6, no. 3, pp. 12-27, May 1992.

[7] B. Mukherjee, "WDM-based local lightwave networks – Part II: Multihop systems," *IEEE Network*, vol. 6, no. 4, pp. 20-32, July 1992.

[8] J. Senior, M. Handley, and M. Leeson, "Developments in wavelength division multiple access networking," *IEEE Communications Magazine*, vol. 36, no. 12, pp. 28-36, Dec. 1998.

[9] H. van As, "Media access techniques: The evolution towards terabit/s LANs and MANs," *Computer Networks and ISDN Systems*, vol. 26, no. 6-8, pp. 603-656, March 1994.

[10] I. Habbab, M. Kavehrad, and C. Sundberg, "Protocols for very high-speed optical fiber local area networks using a passive star topology," *IEEE/OSA Journal of Lightwave Technology*, vol. 5, no. 12, pp. 1782-1794, Dec. 1987.

[11] J. Ryan, "WDM: North american deployment trends," *IEEE Communications Magazine*, vol. 36, no. 2, pp. 40-44, Feb. 1998.

[12] E. Lowe, "Current European WDM deployment trends," *IEEE Communications Magazine*, vol. 36, no. 2, pp. 46-50, Feb. 1998.

[13] B. Mukherjee, "WDM optical communication networks: Progress and Challenges," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1810-1824, Oct. 2000.

[14] D. Stigliani, D. Hanna, and D. Lynch, "Wavelength division multiplexing in light interface technology," *IBM Federal Systems Division Report 71-531-001*, March 1971.

[15] F. Janniello, R. Neuner, R. Ramaswami, and P. Green, Jr., "Multiplex-protocol optical fiber multiplexer for remote computer interconnection," *In Conference Proceedings, OFC'95*, vol. 8, pp. 163-164, San Diego, California, Feb. 1995.

[16] T. Shibagaki, H. Ibe, and T. Ozeki, "Video transmission characteristics in WDM star networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 3, no. 3, pp. 490-495, March 1985.

[17] D. Mestdagh, *Fundamentals of Multiaccess Optical Fiber Networks*, Artech House, 1995.

[18] R. Ramaswami, and K. Liu, "Analysis of effective power budget in optical bus and star networks using erbium-doped fiber amplifiers," *IEEE/OSA Journal of Lightwave Technology*, vol. 11, no. 11, pp. 1863-1871, Nov. 1993.

[19] O. Gerstel, R. Ramaswami, and G. Sasaki, "Fault-tolerant multiwavelength optical rings with limited wavelength conversion," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 7, pp. 1166-1178, Sept. 1998.

[20] C. Dragone, "Efficient $N \times N$ star couplers using Fourier optics," *IEEE/OSA Journal of Lightwave Technology*, vol. 7, no. 3, pp. 479-489, March 1989.

[21] C. Dragone, C. Henry, I. Kaminow, and R. Kistler, "Efficient multichannel integrated optics star coupler on silicon," *IEEE Photonics Technology Letters*, vol. 1, no. 8, pp. 241-243, Aug. 1989.

[22] P. Henry, "High-capacity lightwave local area networks," *IEEE Communications Magazine*, vol. 27, no. 10, pp. 20-26, Oct. 1989.

[23] M. Nassehi, F. Tobagi, and M. Marhic, "Fiber optic configurations for local area networks," *IEEE Journal on Selected Areas in Communications*, vol. 3, no. 11, pp. 941-949, Nov. 1985.

[24] M. Goodman, H. Kobrinski, M. Vecchi, R. Bulley, and J. Gimlett, "The LAMBDANET multiwavelength network: Architecture, applications, and demonstrations," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 6, pp. 995-1004, Aug. 1990.

[25] H. Kobrinski, R. Bulley, M. Goodman, M. Vecchi, C. Brackett, L. Curtis, and J. Gimlett, "Demonstration of high capacity in the LAMBDANET architecture: A multiwavelength optical network," *Electronics Letters*, vol. 23, no. 16, pp. 824-826, July 1987.

[26] E. Arthurs, J. Cooper, M. Goodman, H. Kobrinski, M. Tur, and M. Vecchi, "Multiwavelength optical crossconnect for parallel-processing computers," *Electronics Letters*, vol. 24, no. 2, pp. 119-120., Jan. 1988.

[27] J. Cooper, J. Dixon, M. Goodman, H. Kobrinski, M. Vecchi, E. Arthurs, S. Menocal, M. Tur, and S. Tsuji, "Nanosecond-tunable double-section DFB laser for dynamic wavelength addressing applications," *Electronics Letters*, vol. 24, no. 19, pp. 1237-1239, Sept. 1988.

[28] H. Kobrinski, M. Vecchi, M. Goodman, E. Goldstein, T. Chapuran, J. Cooper, C. Zah, and S. Menocal, "Fast wavelength-switching of laser transmitters and amplifiers," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 6, pp. 1190-1202, Aug. 1990.

[29] I. Chlamtac, and A. Ganz, "Channel allocation protocols in frequency-time controlled high speed networks," *IEEE Transactions on Communications*, vol. 36, no. 4, pp. 430-440, Apr. 1988.

[30] A. Ganz, and Y. Gao, "Time-wavelength assignment algorithms for high performance WDM star based networks," *IEEE Transactions on Communications*, vol. 42, no. 4, pp. 1827-1836, Apr. 1994.

[31] P. Dowd, "Random access protocols for high speed inter-processor communication based on an optical passive star topology," *IEEE/OSA Journal of Lightwave Technology*, vol. 9, no. 6, pp. 799-808, June 1991.

[32] A. Ganz, and Z. Koren, "Performance and design evaluation of WDM stars," *IEEE/OSA Journal of Lightwave Technology*, vol. 11, no. 2, pp. 358-366, Feb. 1993.

[33] M. Karol, and B. Glance, "Performance of the PAC optical packet network," *IEEE/OSA Journal of Lightwave Technology*, vol. 11, no. 8, pp. 1394-1399, Aug. 1993.

[34] E. Arthurs, M. Goodman, H. Kobrinski, and M. Vecchi, "HYPASS: An optoelectronic hybrid packet-switching system," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1500-1510, Dec. 1988.

[35] Y. Kanabar, and N. Baker, "Demonstration of novel optical multichannel grating demultiplexer receiver (MGD) for HDWDM systems," *Electronics Letters*, vol. 25, no. 13, pp. 817-819, June 1989.

[36] H. Kobrinski, M. Vecchi, E. Goldstein, and R. Bulley, "Wavelength selection with nanosecond switching times using DFB laser amplifiers," *Electronics Letters*, vol. 24, no. 15, pp. 969-970, July 1988.

[37] M. Goodman, "Multiwavelength networks and new approaches to packet switching," *IEEE Communications Magazine*, vol. 27, no. 10, pp. 27-35, Oct. 1989.

[38] T. Lee, M. Goodman, and E. Arthurs, "A broadband optical multicast switch," *In Conference Proceedings, ISS'90*, vol. III, pp. 7-13, Stockholm, Sweden, May 1990.

[39] N. Dono, P. Green, Jr., K. Liu, R. Ramaswami, and F. Tong, "A wavelength division multiple access network for computer communication," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 6, pp. 983-994, Aug. 1990.

[40] F. Janniello, R. Ramaswami, and D. Steinberg, "A prototype circuit-switched multi-wavelength optical metropolitan area network," *IEEE/OSA Journal of Lightwave Technology*, vol. 11, no. 5/6, pp. 777-782, May/June 1993.

[41] P. Green, Jr., "An all-optical computer network: Lessons learned," *IEEE Network Magazine*, vol. 6, no. 2, pp. 56-60, Mar. 1992.

[42] E. Hall, J. Kravitz, R. Ramaswami, M. Halvorson, S. Tenbrink, and R. Thomsen, "The Rainbow-II gigabit optical network," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 814-823, June 1996.

[43] I. Habbab, M. Kavehrad, and C. Sundberg, "Protocols for very high-speed optical fiber local area networks using a passive star topology," *IEEE/OSA Journal of Lightwave Technology*, vol. 5, no. 12, pp. 1782-1794, Dec. 1987.

[44] N. Mehravari, "Performance and protocol improvements for very high speed optical fiber local area networks using a passive star topology," *IEEE/OSA Journal of Lightwave Technology*, vol. 8, no. 4, pp. 520-530, Apr. 1990.

[45] G. Sudhakar, N. Georganas, and M. Kavehrad, "Slotted Aloha and reservation Aloha protocols for very high speed optical fiber local area networks using a passive star topology," *IEEE/OSA Journal of Lightwave Technology*, vol. 9, no. 10, pp. 1411-1422, Oct. 1991.

[46] F. Jia, and B. Mukherjee, "The Receiver collision avoidance (RCA) protocol for a single-hop WDM lightwave network," *IEEE/OSA Journal of Lightwave Technology*, vol. 11, no. 5-6, pp. 1053-1065, May/June 1993.

[47] M. Chen, N. Dono, and R. Ramaswami, "A media-access protocol for packet-switched wavelength division multiaccess metropolitan area networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 6, pp. 1048-1057, Aug. 1990.

[48] I. Chlamtac, and A. Fumagalli, "Quadro: A solution to packet switching in optical transmission networks," *Computer Networks and ISDN Systems*, vol. 26, no. 6-8, pp. 945-963, Mar. 1994.

[49] N. Maxemchuk, "Routing in the Manhattan street network," *IEEE Transactions on Communications*, vol. 35, no. 5, pp. 503-512, May. 1987.

[50] M. Hluchyi and M. Karol, "Shufflenet: An application of generalized perfect shuffles to multihop lightwave networks," IEEE/OSA *Journal of Lightwave Technology*, vol. 9, no. 10, pp. 1386-1397, Oct. 1991.

[51] K. Sivarajan, and R. Ramaswami, "Lightwave networks based on de Bruijn graphs," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 70-79, Feb. 1994.

[52] J. Iness, S. Banerjee and B. Mukherjee, "GEMNET: A generalized, shuffle-exchange-based, regular, scalable, modular, multihop, WDM lightwave network," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 470-476, Aug. 1995.

[53] S. Tridandapani, B. Mukherjee, and G. Hallingstad, "Channel sharing in multihop WDM lightwave networks: Do we need more channels?," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 719-727, Oct. 1997.

[54] L. Kazovsky, and P. Poggiolini, "STARNET: A multi-gigabit-per-second optical LAN utilizing a passive WDM star," *IEEE/OSA Journal of Lightwave Technology,* vol. 11, no. 5/6, pp. 1009-1027, May/June 1993.

[55] T. Chiang, S. Agrawal, D. Mayweather, D. Sadot, C. Barry, M. Hickey, and L. Kazovsky, "Implementation of STARNET: A WDM computer communications network," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 824-839, June 1996.

[56] D. Sadot, and L. Kazovsky, "Power budget optimization of STARNET II: An optically amplified direct-detection WDM network with subcarrier control,"

IEEE/OSA Journal of Lightwave Technology, vol. 15, no. 9, pp. 1629-1635, Sept. 1997.

[57] R. Ramaswami, "Multiwavelength lightwave networks for computer communication," *IEEE Communications Magazine*, vol. 31, no. 2, pp. 78-88, Feb. 1993.

[58] G. Hill, "A wavelength routing approach to optical communications networks," in *Conference Proceedings, INFOCOM'88*, pp. 354-362, New York, USA, 1988.

[59] R. Ramaswami, and K. Sivarajan, "Routing and wavelength assignment in all-optical networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 5, pp. 489-500, Oct. 1995.

[60] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath communications: An approach to high bandwidth optical WAN's," *IEEE Transactions on Communications*, vol. 40, no. 7, pp. 1171-1182, July 1992.

[61] R. Barry, and P. Humblet, "On the number of wavelengths and switches in all optical networks," *IEEE Transactions on Communications*, vol. 42, no. 2/3/4, pp. 583-591, Feb./Mar./Apr. 1994.

[62] H. Westlake et al, "Reconfigurable wavelength routed optical networks: A field demonstration," in *Conference Proceedings*, *European Conference on Optical Communications*, *ECOC'91*, vol. 1, pp. 753-756, Valbonne, France, 1991.

[63] F. Kelly, "Blocking probabilities in large circuit-switched networks," *Advances in Applied Probability*, vol. 18, no. 2, pp. 473-505, June 1986.

[64] R. Ramaswami, and K. Sivarajan, "Design of logical topologies for wavelength-routed optical networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 840-851, June 1996.

[65] A. Birman, "Computing approximate blocking probabilities for a class of all-optical networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 852-857, June 1996.

[66] M. Kovačević, and A. Acampora, "Benefits of wavelength translation in all-optical clear-channel networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 868-880, June 1996.

[67] R. Barry, and P. Humblet, "Models of blocking probability in all-optical networks with and without wavelength changers," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 858-867, June 1996.

[68] K. Lee, and V. Li, "A wavelength rerouting algorithm in wide-area all-optical networks," *IEEE/OSA Journal of Lightwave Technology,* vol. 14, no. 6, pp. 1218-1229, June 1996.

[69] D. Banerjee, and B. Mukherjee, "Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 598-607, Oct. 2000.

[70] N. Wauters, and Piet Demeester, "Design of the optical path layer in multiwavelength cross-connected networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 881-892, June 1996.

[71] R. Wagner, R. Alferness, A. Saleh, and M. Goodman, "MONET: Multiwavelength optical networking," *IEEE/OSA Journal of Lightwave Technology*, vol. 14, no. 6, pp. 1349-1355, June 1996.

[72] W. Anderson et al, "The MONET Project- A final report," *IEEE/OSA Journal of Lightwave Technology*, vol. 18, no. 12, pp. 1988-2009, Dec. 2000.

[73] G. Hill et al, "A transport network layer based on optical network elements," *IEEE/OSA Journal of Lightwave Technology*, vol. 11, no. 5/6, pp. 667-678, May/June 1993.

[74] D. K. Hunter et al, "WASPNET: a wavelength switched packet network," *IEEE Communications Magazine*, vol. 37, no. 3, pp. 120-129, March 1999.

[75] A. Carena et al, "OPERA: An optical packet experimental rounting architecture with label swapping capability," *IEEE/OSA Journal of Lightwave Technology*, vol. 16, no. 12, pp. 2135-2145, Dec. 1998.

[76] J. Pierce, "Network for block switches of data," *Bell System Technical Journal*, vol. 51, pp. 1133-1175, July/Aug.1972.

[77] L. Roberts, "Extensions of packet communication technology to a hand-held personal terminal," *in Conference Proceedings, Spring Joint Computer Conference, AFIPS*, pp. 295-298, 1972.

[78] R. Needham, and A. Herbert, "The Cambridge distributed computing Systems," Addison-Wesley, Reading, 1982.

[79] W. Farmer and E. Newhall, "An experimental distributed switching system to handle bursty computer traffic," *in Conference Proceedings, ACM Symposium on Problems in Optimization of Data Communication Systems*, pp. 1-33, New York, USA, Oct. 1969.

[80] R. Gordon, W. Farr, and P. Levine, "Ringnet: A packet switched local network with decentralized control," *Computer Networks*, vol. 3, pp. 373-379, 1980.

[81] W. Bux et al, "A local-area communication network based on a reliable token-ring system," *Local Computer Networks*, pp. 69-82, North-Holland, Amsterdam, 1982.

[82] IEEE, "802.5: Token ring access method," New York: IEEE, 1985c.

[83] E. Hafner, Z. Nenadal, and M. Tscharz, "A digital loop communication system," *IEEE Transactions on Communications*, vol. 22, no. 6, pp. 877-881, June 1974.

[84] D. Paulish, "A fail-soft distributed processing system," *in Conference Proceedings, COMPCOM Fall'80*, pp. 179-184, Sept. 1980.

[85] R. Dixon, "Ring network topology for local data communications," *in Conference Proceedings, COMPCOM Fall'82*, pp. 591-605, Washington, USA, 1982.

[86] M. Liu et al, "Design of the distributed double-loop computer network (DDLCN)," *Journal of Digital Systems*, vol. 5, pp. 3-37, Spring/Summer 1981.

[87] W. Bux, "Local area subnetworks: A performance comparison," *IEEE Transactions on Communications*, vol. 29, no. 10, pp. 1465-1473, Oct. 1981.

[88] T. Lee, K. Lee, and S. Park, "Optimal routing and wavelength assignment in WDM ring networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 2146-2154, Oct. 2000.

[89] G. Li, and R. Simha, "On the wavelength assignment problem in multifiber WDM star and ring networks," *IEEE/ACM Transactions on Networking*, vol. 9, no. 1, pp. 60-68, Feb. 2001.

[90] O. Gerstel, and S. Kutten, "Dynamic wavelength allocation in all-optical ring networks," *in Conference Proceedings, IEEE ICC'97*, vol. 1, pp. 432-436, Montreal, Canada, June 1997.

[91] C. Law, and K. Siu, "Online routing and wavelength assignment in single-hub WDM rings," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 2111-2122, Oct. 2000.

[92] R. Ramaswami, and G. Sasaki, "Multiwavelength optical networks with limited wavelength conversion," *IEEE/ACM Transactions on Networking*, vol. 6, no. 6, pp. 744-754, Dec. 1998.

[93] D. Marcenac, "Benefits of wavelength conversion in optical ring-based networks," *Optical Networks Magazine*, vol. 1, no. 2, pp. 29-35, April 2000.

[94] A. Tucker, "Coloring a family of circular arcs," *SIAM Journal of Applied Mathematics*, vol. 29, no. 3, pp. 493-502, 1975.

[95] L. Bhuyan, D. Ghosal, and Q. Yang, "Approximate analysis of single and multiple ring networks," *IEEE Transactions on Computers*, vol. 38, no. 7, pp. 1027-1040, July 1989.

[96] K. Shrikhande et al, "HORNET: A packet-over-WDM multiple access metropolitan area ring network," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 2004-2016, Oct. 2000.

[97] D. Wonglumsom et al, "Experimental demonstration of an access point for HORNET: A packet-over-WDM multiple-access MAN," *IEEE/OSA Journal of Lightwave Technology*, vol. 18, no. 12, pp. 1709-1717, Dec. 2000.

[98] M. Marsan, A. Bianco, E. Leonardi, M. Meo, and F. Neri, "MAC protocols and fairness control in WDM multirings with tunable transmitters and fixed

receivers," *IEEE/OSA Journal of Lightwave Technology*, vol. 14, no. 6, pp. 1230-1244, June 1996.

[99] M. Marsan, A. Bianco, E. Leonardi, A. Morabito, and F. Neri, "All-optical WDM multi-rings with differenciated QoS," *IEEE Communications Magazine*, vol. 37, no. 2, pp. 58-66, Feb. 1999.

[100] J. Cai, A. Fumagalli, and I. Chlamtac, "The multitoken interarrival time (MTIT) access protocol for supporting variable size packets over WDM ring network," *IEEE Journal on Selected Areas in Commun*ications, vol. 18, no.10, pp. 2094-2104, Oct. 2000.

[101] D. Stoll, P. Leisching, H. Bock, and A. Richter, "Metropolitan DWDM: A dynamically configurable ring for the KomNet field trial in Berlin," *IEEE Communications Mag*azine, vol. 39, no. 2, pp. 106-113, Feb. 2001.

[102] A. Bononi, "Scaling WDM slotted ring networks," *in Conference* Proceedings, *Information Sciences and Systems*, vol. 1, pp. 659-665, Princeton, USA, March 1998.

[103] R. Jain, "The art of computer systems performance analysis," Wiley, 1991.

[104] R. Garzia, and M. Garzia, "Network modeling, simulation, and analysis," Marcel Dekker, New York, 1990.

[105] P. Hartel, and H. Muller, "Functional C," Addison-Wesley, 1997.

[106] W. Willinger, and V. Paxson, "Where mathematics meets the Internet," *Notices of the American Mathematical Society*, vol. 45, no. 8, pp. 961-970, Sept. 1998.

[107] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking*, vol. 2, no.1, pp.1-15, Feb. 1994.

[108] V. Paxson and S. Floyd, "Wide area traffic: The failure of Poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226-244, June 1995.

[109] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson, "Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level," *IEEE/ACM Transactions on Networking*, vol. 5, no.1, pp. 71-86, Feb. 1997.

[110] M. Baldi, and Y. Ofek, "End-to-End delay analysis of videoconferencing over packet-switched networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, pp. 479-492, Aug. 2000.

Appendix 1:

Published articles

Page 156

C. S. Jelger, and J. M. H. Elmirghani, "A Simple MAC Protocol for WDM Metropolitan Access Ring Networks," *in Conference Proceedings, IEEE Globecom'01*, San Antonio, Nov. 2001, in press.

Page 161

C. S. Jelger, and J. M. H. Elmirghani, "Performance Evaluation of a new MAC Protocol for WDM Metropolitan Access Ring Networks," *International Journal of Communication Systems*, in press.

A Simple MAC Protocol for WDM Metropolitan Access Ring Networks

Christophe S. Jelger and Jaafar M. H. Elmirghani
Department of Electrical and Electronic Engineering
University of Wales Swansea
Singleton Park, Swansea SA2 8PP, UK.

***Abstract*- This article considers a WDM slotted-ring network architecture. Used as a metropolitan access network, it is shown through simulation results how a simple MAC protocol can be implemented to achieve fairness between access nodes. Excellent queuing delay and packet dropping probability results are obtained assuming that a maximum load threshold is not exceeded. Further developments are proposed to improve the efficiency, introduce congestion avoidance mechanisms and add QoS capabilities by implementing intelligence within the network access nodes.**

I. INTRODUCTION

The explosion in the popularity of the Internet and its new multimedia applications and services has led to a need for increased bandwidth in the network backbone. In the mean time, Dense-Wavelength-Division-Multiplexing (DWDM) technology is now offering an unprecedented bandwidth as currently available systems support up to about 100 wavelengths per fiber, enabling a single fiber to carry several hundred gigabits of information. WDM-based solutions are therefore expected to appear as the next generation access networks in the metropolitan area [1].

This article presents the results of a simulated WDM-ring network where a minimal MAC protocol is implemented. The complexity of the MAC protocol is reduced on purpose as the intelligence of the network is implemented in the access nodes. This allows for easy software upgrades and wide understanding of the proposed architecture. With Poisson-like traffic, this proposed network provides very good results as long as the overall traffic per wavelength does not reach a threshold value that is given theoretically by a simple equation. The flexibility of the network is very high and attractive and easy physical deployments can be achieved.

New developments are also discussed as they could easily improve the efficiency and introduce congestion avoidance mechanisms and QoS capabilities. Future work will address the impact of non Poisson traffic (i.e. self-similar).

II. NETWORK ARCHITECTURE AND PROTOCOL

A. *The architecture*

We propose a simple multichannel WDM slotted ring topology with a single fiber and a small number of different wavelengths. The simple case of fixed-size slots is considered. The network architecture is illustrated in Fig. 1. It basically consists of a number of access nodes (AN) each of them having add-and-drop capabilities to access the ring slots. Each node has three network ports. A Gigabit Ethernet port is used for transmission between the AN and the access network, the other two OC-like ports are used to access the WDM ring. Each node is also equipped with a transmission and a reception FIFO (first-in first-out) buffer. Table 1. gives detailed values of the architecture that was simulated.

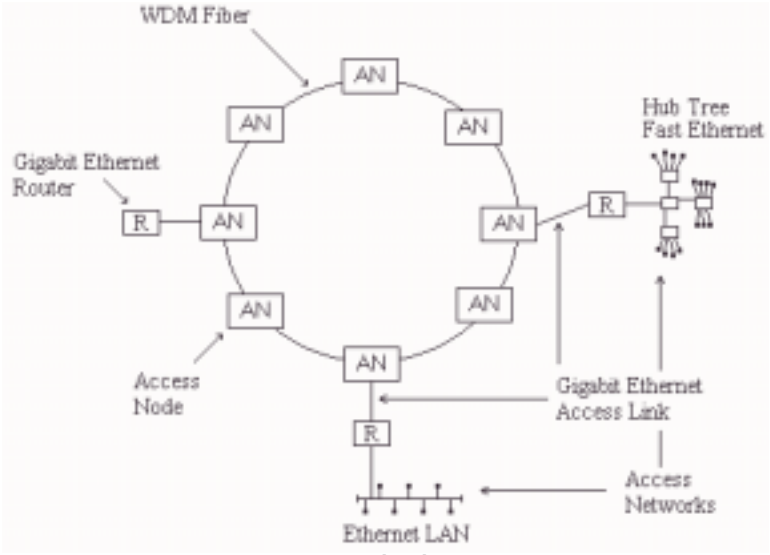


Fig. 1. Network architecture

TABLE 1
NETWORK PARAMETERS

Fiber Length (FBL)	138 240 meters
Wavelength Rate (WVR)	OC-48 2.5 Gb/s
Light Velocity in Fiber (LV)	2×10^8 m/s
Fiber Delay = FBL/LV	691.2 µs
Number of Wavelengths per Fiber	4
Slot Size (S) ≈ Ethernet MTU	12,000 bits
Nb of Slots per Wavelength = (FBL*WVR)/(LV*S)	144

B. The medium access control (MAC) protocol

Each node has a fixed transmitter and a tunable receiver which allow them to transmit on a unique specific wavelength and receive data on any wavelength. A logical architecture is shown in Fig. 2 for a simple case where four wavelengths and four nodes are used. We do not consider addressing capabilities (address resolving and format) nor optical and electronic issues (receiver synchronisation, signal attenuation, chromatic dispersion and others).

When a node wishes to transmit data on a given wavelength it simply inspects (senses) the activity of this channel. An interesting implementation of such a mechanism was proposed in [2] where subcarrier multiplexed tones are transmitted over the wavelength along with the data to identify the destination. If an empty slot is observed data can be transmitted in the slot data unit. If the slot is used the transmission is simply delayed.

In the case of source-stripping operation, the sender is responsible to mark the slot empty after it has completed an entire ring loop. With destination stripping, the destination marks the slot empty once correctly received and thus makes the slot reusable earlier than in the previous scheme. In this simulation the results from the source-stripping scheme are presented as they will serve as a reference set used to assess the efficiency of more complicated implementations (destination-stripping, QoS capabilities, adaptive behaviour when congestion occurs). A restriction is also introduced, which does not allow the sender to immediately reuse the slot it just marked empty. This scheme introduces a very simple fairness mechanism where a node cannot starve the entire network if it never releases any of the slots it uses (this does not apply with the destination-stripping implementation).

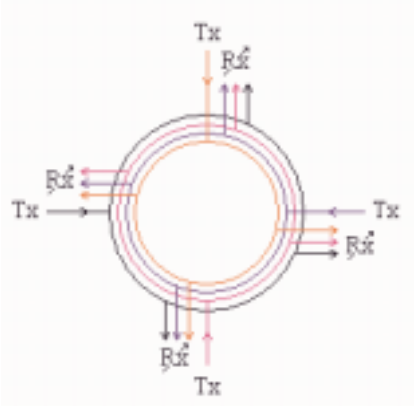


Fig. 2. Logical architecture

Each node has two FIFO (First-In First-Out) buffers (one for transmission and one for reception), each of which can store up to 100 packets. If a node tries to add a packet to a full buffer the packet is simply dropped (discarded) and recovering capabilities are expected to happen at higher level layers (e.g. IP).

III. TRAFFIC SOURCES

The traffic received by each node from its Gigabit Ethernet access link is generated according to a Poisson process. Despite the fact that it was demonstrated that LAN and WAN traffic is better modelled using statistically self-similar processes [3,4], Poisson models are still used in simulations and traffic studies [5,6] as they can be modelled easier than self-similar processes. They are also very attractive as being only defined by a single parameter (in this case the mean interarrival time) and thus they can be quickly tuned to simulate different network loads.

Packet interarrivals (for each Gigabit Ethernet access link) are generated using an inverse transformation [7]. A uniform distribution U(0,1) is generated using a pseudo-random number generator and the exponential interarrivals are computed as $-\lambda \ln(u)$ where λ is the mean interarrival time between two packets. A different seed is used for each access link to initialise the pseudo-random number generator and different network loads are simulated using different values of λ. The size of the packets is fixed to the Ethernet MTU (Maximum Transfer Unit) frame length (i.e. approximately 12,000 bits).

Each node receives packets on its access link port that are intended for one of the other nodes on the ring. The destination node for each packet being transmitted is generated using a uniform distribution. For a network with four nodes, a packet received (for example) by node 1 on the access link port is intended for node 2, 3 or 4 with the same probability (e.g. in this case 1/3). The uniform distribution also uses a pseudo-random number generator with a different seed for each access link.

A file of interarrival times and destination nodes is created for each node at the start of the simulation. A temporary buffer of 100 packets is constantly updated to simulate the packets received from the access link and the packet fields (source node, destination node, time) are computed using the traffic file. The time field (in microseconds) is the packet generation time (given as the time elapsed since the start of the simulation). A discrete global time variable is then used to transfer packets from the temporary buffer to the node transmission buffer. When global time is incremented (the increment is equal to a slot duration), the simulator (for each node) ensures all packets whose generation time is less than the global time variable are moved from the temporary buffer to the transmission buffer. The main advantage of this technique is that it removes the complexity of generating traffic in a real-time manner.

Fig. 3 shows the number of packets received by a given node on its access link ports every 200 µs. The initial load of 25% is increased every 20 ms to reach 75% after 40ms.

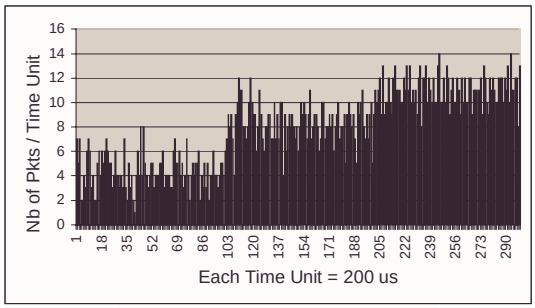


Fig. 3. An access link traffic

IV. SIMULATIONS RESULTS

This section presents the results of discrete event simulations for a 16-node and 4 wavelengths architecture. Different logical topologies were simulated where 2 to 6 nodes shared a wavelength for transmission. All access link traffic loads have the same mean value (for each specific simulation) and they are given as a percentage of the average Gigabit Ethernet link usage. It is very important to note that these results are given when all nodes transmit at the same rate (section V. describes the case when nodes transmit at different rates).

A. *Excellent fairness between nodes*

A very interesting result of the simulations is that an excellent fairness is achieved between the nodes that share a wavelength. The position of the nodes on the ring does not alter the results (i.e. wherever the nodes are situated, results are identical). Despite the fact that each access link traffic is different (but has the same mean interarrival time), the average throughput, queuing delay, buffer load and packet dropping probability are equals (less than 1% difference) among all the nodes sharing a specific wavelength. Each of these sets of data are of course different for each node but the average values are always similar.
Fig. 4 shows the throughput (computed every 691.2 µs = fiber delay) for four nodes sharing the same wavelength, each access link load being 25%. The average throughput for each node is ≈ 250 Mb/s and the average queuing delay (not represented) is 0.57 µs.

B. *Node throughput Vs Access link load – A theoretical threshold*

Fig. 5 shows the average node throughput in three different cases where 2, 3 or 4 nodes share a wavelength for transmission.

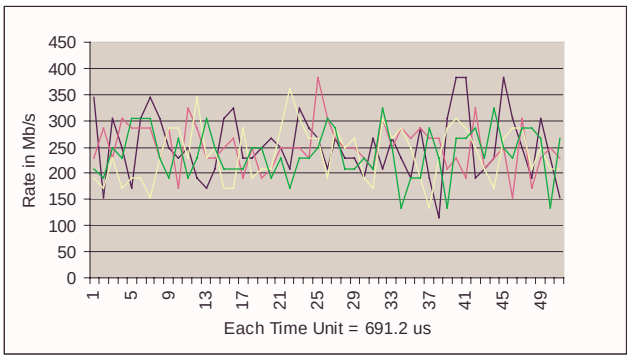


Fig. 4. Throughput for 4 nodes (25% access link load)

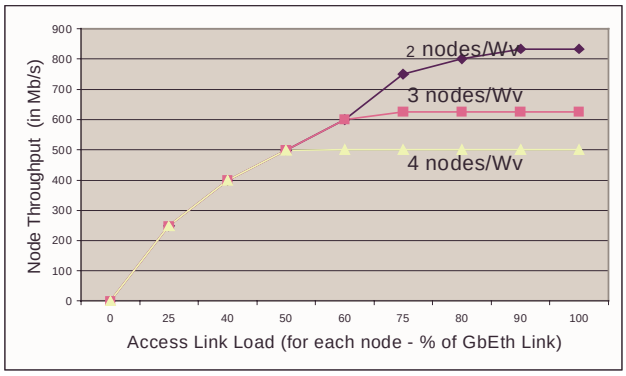


Fig.5. Average throughput Vs access link load

In each case there is a maximum throughput that each node can achieve. As the source-stripping scheme is used and due to the MAC protocol restriction that does not allow a node to reuse slots it marked empty, the *MTpN* is found to be upperbound by a theoretical threshold value given as

$$MTpN = \frac{Wr}{NbN + 1} \qquad (1)$$

where *MTpN* is the maximum throughput per node, *Wr* is the wavelength rate and *NbN* is the number of nodes per wavelength. Equation (1) gives the maximum throughput that can be achieved by each node **when all nodes transmit at the same rate**.

With *N* nodes the existence of the upper bound is explained as follows. A node can transmit packets during one N^{th} of a cycle (we call the time duration of a ring round 'a cycle') and then cannot transmit during the rest of the cycle as the remaining slots have been used by the *N*-1 other nodes. It then releases during the first N^{th} of the next cycle the slots it used but still cannot transmit as the MAC protocol does not allow it to do so. Therefore, during a total time of (1+1/*N*) cycle a node can only transmit for a time of 1/*N* cycle.

Thus during one round a node can transmit during the following fraction of a cycle :

$$T = \frac{1/N}{1+1/N} = \frac{1}{N+1} \quad (2)$$

The *MTpN* is then clearly deduced as being given by (1) as a node can "use" the available wavelength bandwidth during the fraction of time given by (2).

The maximum throughput for the entire wavelength (MTpW) is then easily found as being :

$$MTpW = (1) \times NbN = \frac{Wr \times NbN}{NbN+1} \quad (3)$$

$$\text{Or} \qquad MTpW = \frac{Wr(NbN+1)-Wr}{NbN+1} = Wr - \frac{Wr}{NbN+1} \quad (4)$$

Equation (4) is preferred to (3) as the negative part of the equation gives the amount of bandwidth that is lost due to the MAC protocol restriction on slot reuse. From (4) it can be clearly seen that if a single node uses a wavelength then the restriction introduced reduces the *MTpW* to *Wr*/2. This is because a node that exists on its own on a given wavelength (rare case) will fill all the slots in a cycle (assuming it has sufficient data to transmit) and will subsequently make them empty in the next cycle without being allowed to reuse them. In practice, this will not be a crucial limitation as nodes will usually not transmit data at a faster rate than *Wr*/2 (e.g. Gigabit Ethernet link Vs OC-192 wavelength rate).

Table 2 presents theoretical throughput values and experimental results for the cases where 2 to 6 nodes share a wavelength. Each access link load is 90%. Percentage of the wavelength and the links use are also given. Theoretical values and experimental results are shown to be extremely close.

TABLE 2
THEORY Vs EXPERIMENT

Number of nodes	Theor. Throughput Per Node (Mb/s)	Exper. Throughput Per Node (Mb/s)
2	833.3 (83.3 %)	833.9 (83.3 %)
3	625 (62.5 %)	625.5 (62.6 %)
4	500 (50.0 %)	500.3 (50.0 %)
5	416.7 (41.7 %)	416.9 (41.7 %)
6	357.1 (35.7 %)	357.2 (35.7 %)

Number of nodes	Theor. Throughput per Wavelength (Mb/s)	Exper. Throughput per Wavelength (Mb/s)
2	1666.6 (66.7 %)	1667.8 (66.7 %)
3	1875 (75.0 %)	1876.5 (75.0 %)
4	2000 (80.0 %)	2001.2 (80.0 %)
5	2083.3 (83.3 %)	2084.5 (83.4 %)
6	2142.9 (85.7 %)	2143.2 (85.7 %)

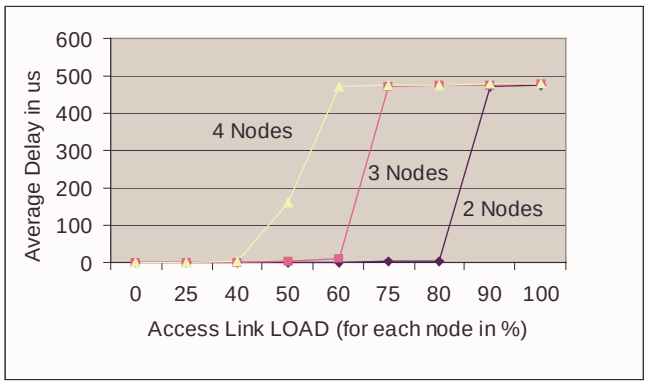


Fig. 6. Average queuing-delay

C. *Queuing-Delay Vs Access Link Load*

In this section we show the average queuing delay experienced by packets before being transmitted on the ring (i.e. the time spent by a packet in a node transmission buffer). Fig. 6 gives the average queuing-delay in three different cases where 2,3 or 4 nodes share a wavelength for transmission. The delay is excellent, being less than a slot duration (i.e. 4.8 µs) if the overall (node) rates do not exceed the maximum rate threshold for the wavelength. In the case where the threshold is exceeded, the queuing-delay increases very quickly to reach a steady value of ≈ 470 µs. Not surprisingly, this value is just inferior to the transmission buffer delay time of 480 µs (i.e. 100 packets at 2.5 Gb/s).

D. *Buffer Load and Packet Dropping Probability*

This set of results presents the average buffer load and the packet dropping probability for the case where 4 nodes share a wavelength for transmission. Each access link load is 60 % and thus it is the case where the maximum wavelength throughput is exceeded.

Initially the buffer is empty but it starts filling up quickly (depending on the access link load). When the buffer load reaches 100% packet dropping starts. As seen in Fig. 7, the packet dropping probability fluctuates as the node is able to transmit in an intermittent manner when the network is congested.

In other cases (threshold not reached) the packet dropping probability is null and the buffer load is less than 5 %.

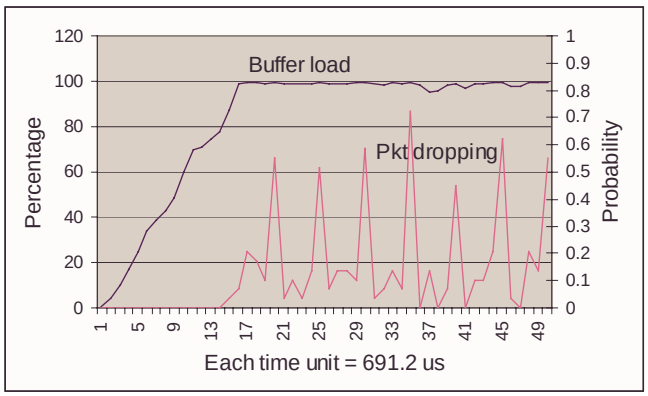


Fig. 7. Buffer load and packet dropping probability

V. Discussion – Developments to Come

As seen previously, the behaviour of the proposed network implementation gives very good results as long as the maximum throughput for a wavelength is not exceeded. A network based on this architecture with 16 wavelengths operating at OC-192 (10 Gb/s) could accommodate 9 nodes (each accessing the ring through a Gigabit Ethernet link) per wavelength and a total of 144 access nodes without experiencing congestion. Due to the simplicity of the MAC protocol being used and the flexibility of the architecture (node positions on the ring do not alter performances), such a network is particularly attractive in the metropolitan access network market.

In the case where nodes do not transmit at the same rate but where the overall traffic does not reach the threshold, the simulation results are still excellent. The average throughput remains very good and the average queuing-delay is just slightly worse than in the case where all nodes transmit at the same constant rate. Simulations where the length of the fiber is changed (e.g. doubled) give very similar results (i.e. all data sets remain almost unchanged).

When considering the traffic sources, it will be interesting to compare the results of the same simulations when self-similar traffic sources are used. A self-similar traffic source is currently under development and the upcoming results will also be published.

Finally, new implementations are also being developed. These include the destination-stripping scheme, an adaptive mechanism to avoid congestion (the nodes equipped with a tunable transmitter can switch from a wavelength to another), and QoS capabilities. These are described with more details in [8].

VI. Conclusion

This article presents a WDM-metropolitan slotted ring architecture using a fairly simple MAC protocol to arbitrate access to the network. Simulation results give very good performances as long as a maximum wavelength load is not exceeded. This value is given by a simple equation which allows for easy provisioning of the network size and scalability. New developments are discussed as they are expected to improve the overall network efficiency and add new functionality such as QoS capabilities.

References

[1] Cisco Limited Technical White Paper, "WDM-Based Metropolitan-Area Deployment of SRP and PoS with the Cisco ONS 15190," October 2000.

[2] S. Gemelos, I. White, D. Wonglumsom, K. Shrikhande, T. Ono, and L. Kazovsky, "WDM Metropolitan Area Network Based on CSMA/CA Packet Switching," IEEE Photon. Technol. Letters, vol. 11, no. 11, November 1999, pp. 1512-1514.

[3] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Trans. Networking*, vol.3, no.3, June 1995, pp.226-244.

[4] W. Willinger, and V. Paxson, "Where Mathematics meets the Internet," *Notices of the American Mathematical Society*, vol. 45, no. 8, Sept. 1998, pp. 961-970.

[5] M. Marsan, A. Bianco, E. Leonardi, A. Morabito, and F. Neri, "All-Optical WDM Multi-Rings with Differentiated QoS," *IEEE Commun. Mag.*, vol. 37, no. 2, Feb.1999, pp. 58-66.

[6] J. Roberts, "Traffic Theory and the Internet," *IEEE Commun. Mag*, vol. 39, no. 1, Jan. 2001, pp. 94-99.

[7] R. Jain, *The Art Of Computer Systems Performance Analysis*, Wiley, 1991, pp. 474.

[8] C. S. Jelger, and J. M. H. Elmirghani, "Intelligent nodes for QoS guarantees in IP over WDM metropolitan network," unpublished. Submitted *to IEEE Commun. Mag.*, Feature Topic "Intelligence in Optical Networks", to appear in Sept. 2001.

Performance Evaluation of a new MAC Protocol for WDM Metropolitan Access Ring Networks

Christophe S. Jelger and Jaafar M. H. Elmirghani

Department of Electrical and Electronic Engineering
University of Wales Swansea
Singleton Park, Swansea SA2 8PP, UK.

SUMMARY

This article considers a WDM slotted-ring network architecture. A slotted MAC protocol was designed to be used in a metropolitan environment. The results of a discrete event simulation are presented. Assuming that a maximum load threshold is not exceeded, fairness between access nodes is achieved by introducing a very simple mechanism. Excellent queuing delay and packet dropping probability results are obtained. Future developments are presented as they are expected to improve the efficiency and introduce QoS capabilities and congestion avoidance mechanisms.

KEY WORDS: Medium access control (MAC) protocol, optical networks, wavelength-division multiplexing (WDM), WDM ring.

1. INTRODUCTION

The explosion in the popularity of the Internet and its new multimedia applications and services has led to a need for increased bandwidth in the network backbone. In the mean time, Dense-Wavelength-Division-Multiplexing (DWDM) technology is now offering an unprecedented bandwidth as currently available systems support up to about 100 wavelengths per fiber, enabling a single fiber to carry several hundred gigabits of information. WDM-based solutions are therefore expected to appear as the next generation access networks in the metropolitan area [1-4].

This article presents the results of a simulated WDM-ring network where a minimal MAC protocol is implemented. The complexity of the MAC protocol is reduced on purpose as the intelligence of the network is implemented in the access nodes.

This allows for easy software upgrades and wide understanding of the proposed architecture. With Poisson-like traffic, this proposed network provides very good results as long as the overall traffic per wavelength does not reach a threshold value that is given theoretically by a simple equation. The flexibility of the network is very high and attractive and easy physical deployments can be achieved.

New developments are also discussed as they could easily improve the efficiency and introduce congestion avoidance mechanisms and QoS capabilities. Future work will address the impact of non Poisson traffic (i.e. self-similar).

2. NETWORK ARCHITECTURE AND PROTOCOL

2.1 Architecture

We propose a simple multichannel WDM slotted ring topology with a single fiber and a small number of different wavelengths. The simple case of fixed-size slots is considered. The network architecture is illustrated in Fig. 1.

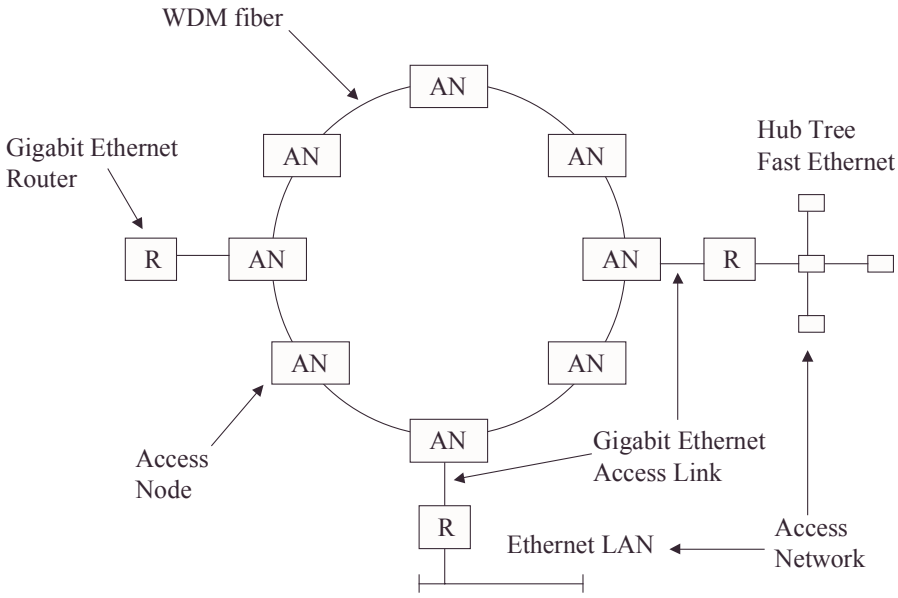


Figure 1. Network architecture

It basically consists of a number of access nodes (AN) each of them having add-and-drop capabilities to access the ring slots. Each node has three network ports. A Gigabit Ethernet (GbE) port is used for transmission between the AN and the access network, the other two OC-like ports are used to access the WDM ring. Data sent from one access network to another is first received by an AN through the GbE link and is then transmitted through the WDM ring to the destination AN which delivers the data to the intended access network. Table 1 gives detailed network parameters of one of the simulated architecture.

Fiber Length (FBL)	138 240 meters
Wavelength Rate (WVR)	OC-48 2.5 Gb/s
Light Velocity in Fiber (LV)	2×10^8 m/s
Fiber Delay = FBL/LV	691.2 µs
Number of Wavelengths per Fiber	4
Slot Size (S) ≈ Ethernet MTU	12,000 bits
Nb of Slots per Wavelength = (FBL*WVR)/(LV*S)	144

Table 1. Network parameters

Each node has a fixed transmitter and a tunable receiver which allow them to transmit on a unique specific wavelength and receive data on any wavelength. This architecture is preferred to the packet switched model where a tunable transmitter (TT) and a fixed receiver (FR) are used [5] or to the one where many fixed transmitters (FT) and fixed receivers are used [6]. In both cases, each wavelength can be seen as a link dedicated to the corresponding receiver and this implementation therefore requires as many wavelengths as there are nodes in the network. This is clearly a scalability limitation [7].
In our proposed architecture (FT-TR), nodes can receive packets on any wavelengths. If there are less wavelengths than there are nodes in the network, the nodes will also have to share all the wavelengths available for transmission. This is not a limitation as the wavelength rate is generally greater than the node access link rate (e.g. wavelengths operating at OC-192/10 Gb/s and Gigabit Ethernet access links) and therefore a wavelength can be shared for transmission by a high number of nodes. The main advantage of this architecture is that the number of nodes can be much larger than the number of wavelengths, therefore satisfying the scalability requirements of the MAN market.

2.2 The medium access control (MAC) protocol

Each node has a fixed transmitter and a tunable receiver which allow them to transmit on a unique specific wavelength and receive data on any wavelength. A logical architecture is shown in Fig. 2 for a simple case where four wavelengths and four nodes are used. In this paper, we do not consider addressing capabilities (address resolving and format) nor optical and electronic issues (receiver synchronisation, signal attenuation, chromatic dispersion and others).

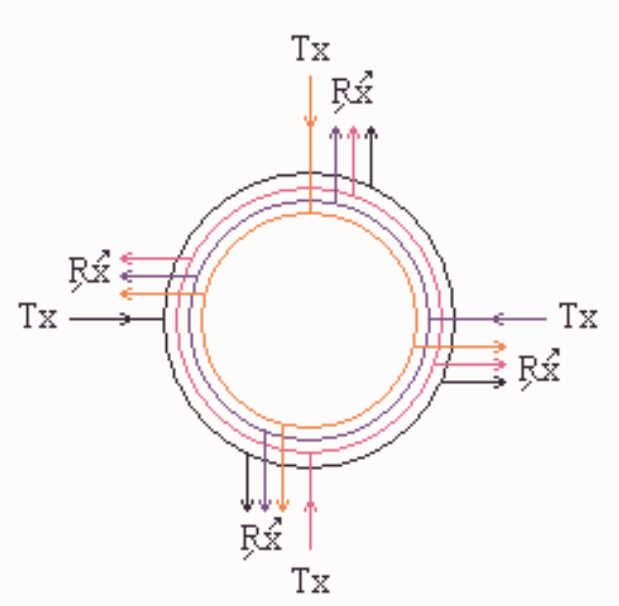


Figure 2. Logical architecture

When a node wishes to transmit data on a given wavelength it simply inspects (senses) the activity of this channel. If an empty slot is observed data can be transmitted in the slot data unit. If the slot is used the transmission is simply delayed. An interesting physical implementation of such a mechanism was proposed in [8] where subcarrier multiplexed tones are transmitted over the wavelength along with the data to identify the destination. This scheme is far cheaper than demultiplexing the wavelengths and monitoring them separately and it does not require any sweeping mechanism which can dramatically slow down the packet detection process [8].
In the case of source-stripping operation, the sender is responsible to mark the slot empty after it has completed an entire ring loop. With destination stripping, the destination marks the slot empty once correctly received and thus makes the slot reusable earlier than in the previous scheme. In this simulation the results from the source-stripping scheme are presented as they will serve as a reference set used to assess the efficiency of more complicated implementations (destination-stripping, QoS capabilities, adaptive behaviour when congestion occurs). A restriction is also introduced, which does not allow the sender to immediately reuse the slot it just marked empty. This scheme introduces a very simple fairness mechanism where a node cannot starve the entire network if it never releases any of the slots it uses (this does not apply with the destination-stripping implementation).
Each node has two FIFO (First-In First-Out) buffers (one for transmission and one for reception), each of which can store up to 100 packets. If a node tries to add a packet to a full buffer the packet is simply dropped (discarded) and recovering capabilities are expected to happen at higher level layers (e.g. IP).

3. SIMULATION ENVIRONMENT

The traffic received by each node from its Gigabit Ethernet access link is generated according to a Poisson process. Despite the fact that it was demonstrated that LAN and WAN traffic is better modeled using statistically self-similar processes [9,10], Poisson models are still used in simulations and traffic studies [11,12] as they can be modeled easier than self-similar processes. They are also very attractive as being only defined by a single parameter (in this case the mean interarrival time) and thus they can be quickly tuned to simulate different network loads.
Packet interarrivals (for each Gigabit Ethernet access link) are generated using an inverse transformation method [13]. The size of the packets is fixed to the Ethernet MTU (Maximum Transfer Unit) frame length (i.e. approximately 12,000 bits).
Each node receives packets on its access link port that are intended for one of the other nodes on the ring. The destination node for each packet being transmitted is generated using a uniform distribution.
It is also important to note that all simulations were ran for a time long enough to reach steady-state results. In general, between two and eight millions packets were transmitted per node for each simulation. Fig. 3 shows the number of packets received by a given node on its access link ports every 200 μs. The initial load of 25% is increased every 20 ms to reach 75% after 40ms.

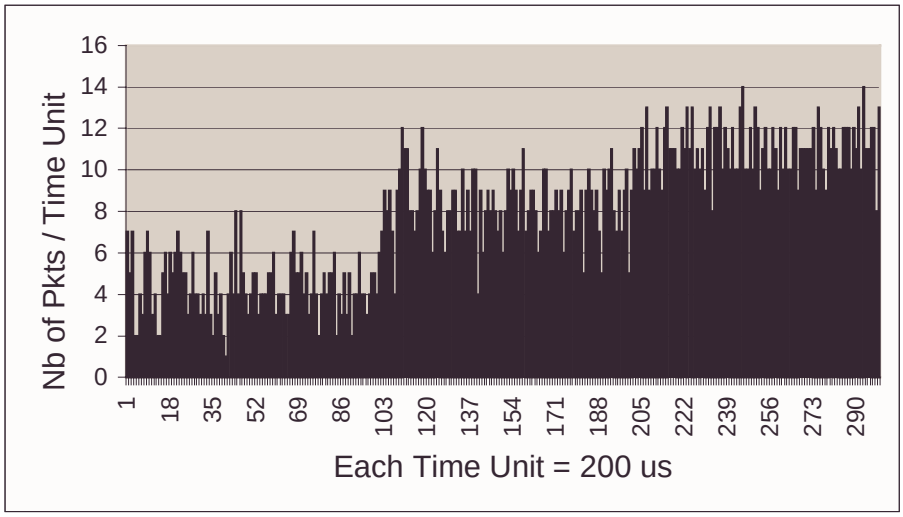


Figure 3. An access link traffic

4. SIMULATIONS RESULTS

This section presents the results of discrete event simulations for a 16-node and 4 wavelengths architecture. Different logical topologies were simulated where 2 to 6 nodes shared a wavelength for transmission. All traffic sources are statistically identical Poisson process. All loads are given as a percentage of the average Gigabit Ethernet link usage. It is very important to note that these results are given when all nodes transmit at the same rate (section V. describes the case when nodes transmit at different rates).

4.1 Excellent fairness between nodes

A very interesting result of the simulations is that an excellent fairness is achieved between the nodes that share a wavelength. The position of the nodes on the ring does not alter the results (i.e. wherever the nodes are situated, results are identical). The average throughput, queuing delay, buffer load and packet dropping probability are equals (less than 1% difference) among all the nodes sharing a specific wavelength. Each of these sets of data are of course different for each node but the average values are always similar.
Fig. 4 shows the throughput (computed every 691.2 μs = fiber delay) for four nodes sharing the same wavelength, each access link load being 25%. The average throughput for each node is ≈ 250 Mb/s and the average queuing delay (not represented) is 0.57 μs.

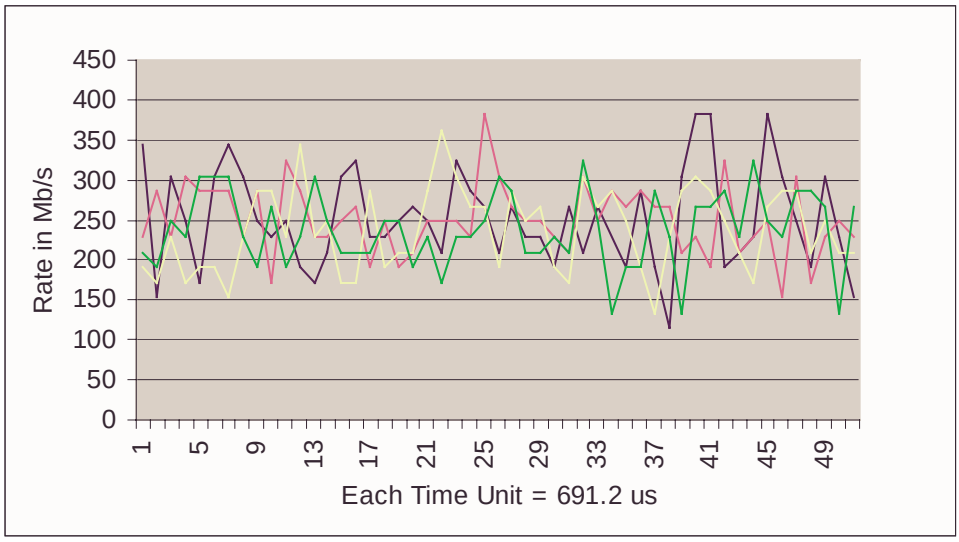


Figure 4. Throughput for 4 nodes (25% load)

4.2 Node throughput Vs Access link load – A theoretical threshold

Fig. 5 shows the average node throughput in three different cases where 2, 3 or 4 nodes share a wavelength for transmission.

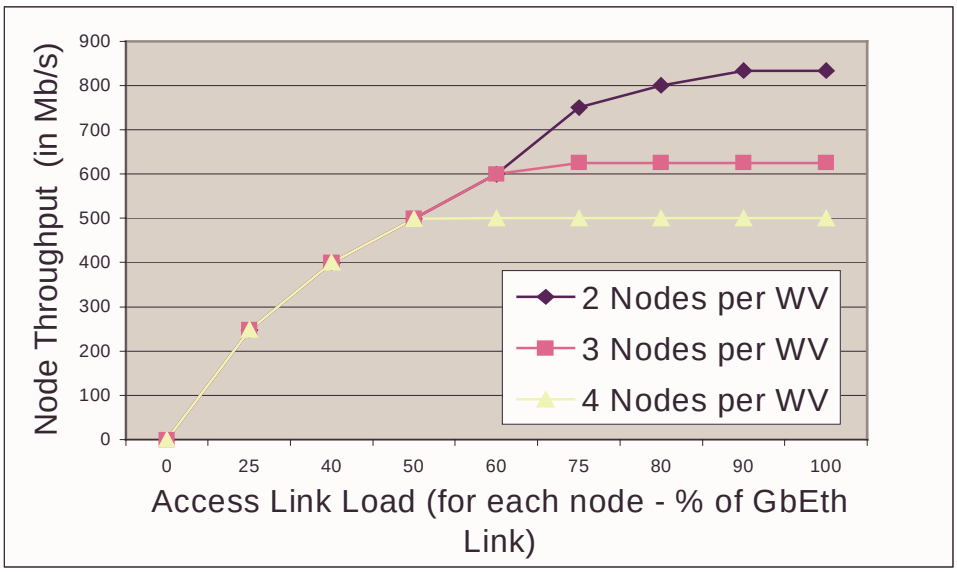


Figure 5. Average node throughput

In each case there is a maximum throughput that each node can achieve. As the source-stripping scheme is used and due to the MAC protocol restriction that does not allow a node to reuse slots it marked empty, the *MTpN* is found to be upperbound by a theoretical threshold value given as

$$MTpN = \frac{Wr}{NbN + 1} \tag{1}$$

where *MTpN* is the maximum throughput per node, *Wr* is the wavelength rate and *NbN* is the number of nodes per wavelength. Equation (1) gives the maximum throughput that can be achieved by each node **when all nodes transmit at the same rate**.

With *N* nodes the existence of the upper bound is explained as follows. A node can transmit packets during one N^{th} of a cycle (we call the time duration of a ring round 'a cycle') and then cannot transmit during the rest of the cycle as the remaining slots have been used by the *N*-1 other nodes. It then releases during the first N^{th} of the next cycle the slots it used but still cannot transmit as the MAC protocol does not allow it to do so. Therefore, during a total time of (1+1/*N*) cycle a node can only transmit for a time of 1/*N* cycle.
Thus during one round a node can transmit during the following fraction of a cycle :

$$T = \frac{1/N}{1+1/N} = \frac{1}{N+1} \tag{2}$$

The *MTpN* is then clearly deduced as being given by (1) as a node can "use" the available wavelength bandwidth during the fraction of time given by (2).

The maximum throughput for the entire wavelength (MTpW) is then easily found as being :

$$MTpW = (1) \times NbN = \frac{Wr \times NbN}{NbN + 1} \tag{3}$$

$$\text{or} \qquad MTpW = \frac{Wr(NbN + 1) - Wr}{NbN + 1} = Wr - \frac{Wr}{NbN + 1}. \tag{4}$$

Equation (4) is preferred to (3) as the negative part of the equation gives the amount of bandwidth that is lost due to the MAC protocol restriction on slot reuse. From (4) it can be clearly seen that if a single node uses a wavelength then the restriction introduced reduces the *MTpW* to *Wr*/2. This is because a node that exists on its own on a given wavelength (rare case) will fill all the slots in a cycle (assuming it has sufficient data to transmit) and will subsequently make them empty in the next cycle without being allowed to reuse them. In practice, this will not be a crucial limitation as nodes will

usually not transmit data at a faster rate than *Wr*/2 (e.g. Gigabit Ethernet link Vs OC-192 wavelength rate).

Table 2 presents theoretical throughput values and experimental results for the cases where 2 to 6 nodes share a wavelength. Each access link load is 90%. Percentage of the wavelength and the links use are also given. Theoretical values and experimental results are shown to be extremely close.

Number of nodes per wavelength	Theor. ***MTpN*** (Mb/s)		Exper. ***MTpN***(Mb/s)	
2	833.3	(83.3 %)	833.9	(83.3 %)
3	625	(62.5 %)	625.5	(62.6 %)
4	500	(50.0 %)	500.3	(50.0 %)
5	416.7	(41.7 %)	416.9	(41.7 %)
6	357.1	(35.7 %)	357.2	(35.7 %)

Number of nodes per wavelength	Theor. ***MTpW***(Mb/s)		Exper. ***MTpW***(Mb/s)	
2	1666.6	(66.7 %)	1667.8	(66.7 %)
3	1875	(75.0 %)	1876.5	(75.0 %)
4	2000	(80.0 %)	2001.2	(80.0 %)
5	2083.3	(83.3 %)	2084.5	(83.4 %)
6	2142.9	(85.7 %)	2143.2	(85.7 %)

Table 2. Theoritical and experimental results comparison

4.3 Queuing-Delay Vs Access Link Load

In this section we show the average queuing delay experienced by packets before being transmitted on the ring (i.e. the time spent by a packet in a node transmission buffer). Fig. 6 gives the average queuing-delay in three different cases where 2,3 or 4 nodes share a wavelength for transmission. The delay is excellent, being less than a slot duration (i.e. 4.8 μs) if the overall (node) rates do not exceed the maximum rate threshold for the wavelength. In the case where the threshold is exceeded, the queuing-delay increases very quickly to reach a steady value of $\approx$ 470 μs. Not surprisingly, this value is just inferior to the transmission buffer delay time of 480 μs (i.e. 100 packets at 2.5 Gb/s).

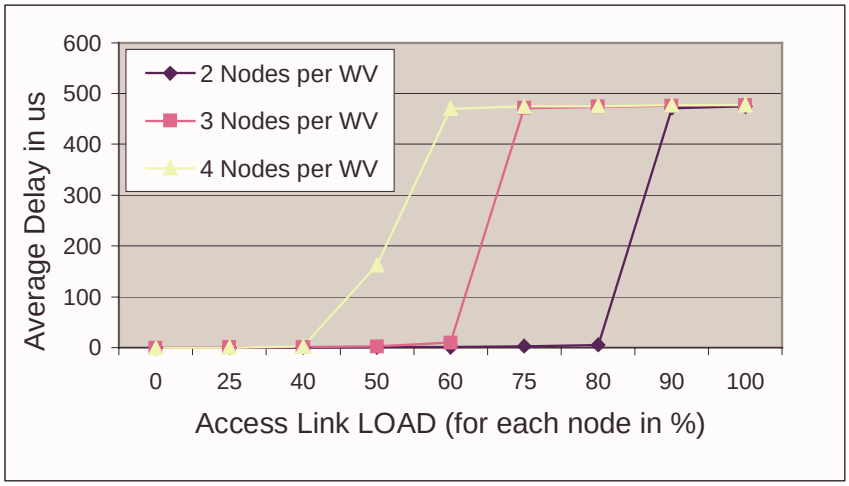


Figure 6. Average queuing delay

4.4 Buffer Load and Packet Dropping Probability

This set of results presents the average buffer load and the packet dropping probability for the case where 4 nodes share a wavelength for transmission. Each access link load is 60 % and thus it is the case where the maximum wavelength throughput is exceeded.
Initially the buffer is empty but it starts filling up quickly (depending on the access link load). When the buffer load reaches 100% packet dropping starts. As seen in Fig. 7, the packet dropping probability fluctuates as the node is able to transmit in an intermittent manner when the network is congested.
In other cases (threshold not reached) the packet dropping probability is null and the buffer load is less than 5 %.

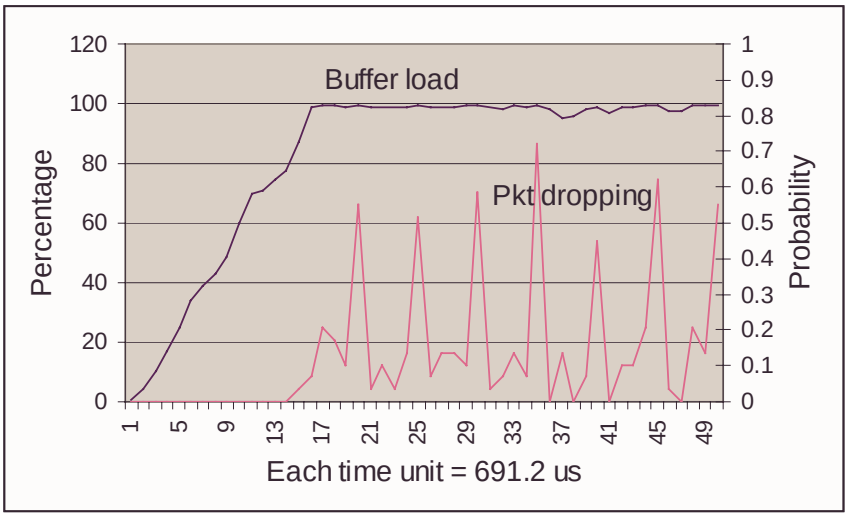


Figure 7. Buffer load and packet dropping probability

4.5 Efficiency comparison

Compared with the results in [6] where a different and more complex MAC protocol is used, the proposed implementation achieves a better performance with a much simpler protocol. The proposed implementation can achieve a *MTpN* of 8.88 Gb/s (equation (1)) with a bandwidth efficiency (η) of 0.89 when *Wr*=80 Gb/s, *NbN*=8 and only one wavelength is used. In [6] (Poisson traffic) and under comparable constraints, the *MTpN* is 1.7 Gb/s and η=0.18. Also in [6] with *Wr*=40 Gb/s, two wavelengths and *NbN*=4, the *MTpN* is 4.8 Gb/s and η=0.55. Our implementation achieves a *MTpN* of 8 Gb/s and η=0.8. It is worth mentioning that comparable packet sizes are used in both implementations.

5. RECEIVER COLLISIONS

The main problem of using tunable receiver is that it introduces receiver collisions. Indeed, a node may receive two or more packets at the same time on different wavelengths. This problem can be avoided by simply replacing the tunable receiver by as many fixed-tuned receivers as there are wavelengths in the network as it is done in [6]. However, this solution involves a more costly hardware implementation and the scalability of the network is reduced as adding wavelengths to the network requires a hardware update of all the access nodes. Further in this paper we will call this solution the Multi-Rx architecture. Therefore, all the results presented up to this point are only valid in the case of a Multi-Rx architecture. If a node receives two or more packets at the same time, these are simply buffered once they were received by each autonomous fixed-tuned receiver. Therefore, receiver collisions do not affect the performance.

It is worth noting that, for a fixed number of wavelengths, as the number of nodes increases in the network, receiver collisions are less likely to occur. From Fig. 8 (for the Multi-Rx architecture) and out of more than 10 millions packets received among all the nodes, simple collisions (two packets received at the same time) reach up to 2% of all reception events and it reaches 1% for a network load of 0.7. In the mean time, double collisions (three packets received) are only 0.08% of receiver events (this is not shown in Fig. 8) at the same network load of 0.7. The network load represents the normalized overall network load (i.e. a network load of 1 means that the added nodes throughputs reach the total transmission capacity of 10 Gb/s).

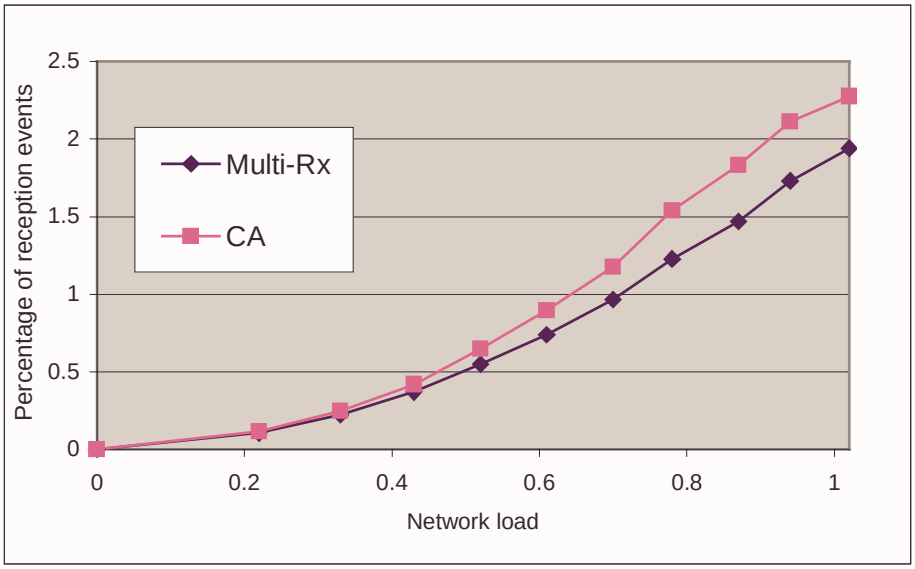


Figure 8. Receiver Collisions

However, if a tunable receiver is to be used, a mechanism must be introduced to handle receiver collisions. Solving the problem by implementing a more complex hardware architecture (e.g. by using optical fiber delays) is not desirable in our proposed architecture because it adds up to the hardware complexity and this is contrary to our approach. We propose a simple solution whereby a packet is received and all other colliding/conflicting packets are allowed to do another loop round the ring. In the case of a simple collision, one packet will simply do one more loop and will come back to the destination node and will eventually be received without collision. This may seem as a counter-intuitive scheme but implementing this collision avoidance (CA) mechanism will not affect the average packet transmission delay in a significant manner as the ring latency is very low (691.2 μs). It does indeed induce more collisions as this can be seen in Fig. 8 (CA) but there is very little difference compared to the Multi-Rx implementation. Moreover, through simulations it is concluded that the previous performance achieved with the Multi-Rx architecture (throughput, queuing delay, packet dropping) is not affected by the CA mechanism. The overall packet transmission delay is of course affected by the CA scheme as shown in Fig. 9 but in the worst case (network load of 1), this value reaches 800 μs which is a very good delay even in the presence of real-time multimedia traffic where the end-to-end user delay (transmission delay + processing time) must be below 100 ms. Furthermore, packets may arrive out of sequence because of the CA scheme but this does not matter in data traffic as higher level layers (TCP/IP) will re-sequence the data.

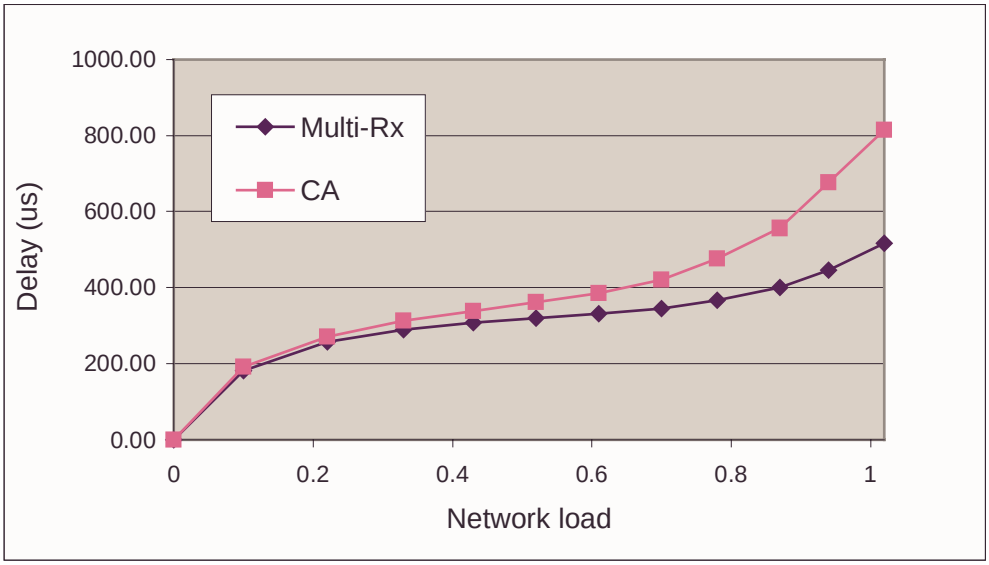


Figure 9. Transmission Delay

6. DISCUSSION – DEVELOPMENTS TO COME

As seen previously, the behaviour of the proposed network implementation gives very good results as long as the maximum throughput for a wavelength is not exceeded. A network based on this architecture with 16 wavelengths operating at OC-192 (10 Gb/s) could accommodate 9 nodes (each accessing the ring through a Gigabit Ethernet link) per wavelength and a total of 144 access nodes without experiencing congestion. Due to the simplicity of the MAC protocol being used and the flexibility of the architecture (node positions on the ring do not alter performances), such a network is particularly attractive in the metropolitan access network market.

In the case where nodes do not transmit at the same rate but where the overall traffic does not reach the threshold, the simulation results are still excellent. The average throughput remains very good and the average queuing-delay is just slightly worse than in the case where all nodes transmit at the same constant rate. Simulations where the length of the fiber is changed (e.g. doubled) give very similar results (i.e. all data sets remain almost unchanged).

When considering the traffic sources, it will be interesting to compare the results of the same simulations when self-similar traffic sources are used. A self-similar traffic source is currently under development and the upcoming results will also be published.

Finally, new implementations are also being developed. These include the destination-stripping scheme, an adaptive mechanism to avoid congestion (the nodes equipped with a tunable transmitter can switch from a wavelength to another), and QoS capabilities.

7. CONCLUSION

This article presents a WDM-metropolitan slotted ring architecture using a fairly simple MAC protocol to arbitrate access to the network. Simulation results give very good performances as long as a maximum wavelength load is not exceeded. This value is given by a simple equation which allows for easy provisioning of the network size and scalability. New developments are discussed as they are expected to improve the overall network efficiency and add new functionality such as QoS capabilities.

REFERENCES

[1] *IEEE Commun. Mag*. "Optical Switching Networks: From Circuits to Packets", vol. 39, no.3, March 2001.

[2] N. Ghani, S. Dixit, and T. Wang, "On IP-over-WDM Integration," *IEEE Commun. Mag*., vol. 38, no. 3, pp. 72-84, March 2000.

[3] M. Kuznetsov, N. Froberg, S. Henion, H. Rao, J. Korn, K. Rauschenbach, E. Modiano, and V. Chan, "A Next-Generation Optical Regional Access Network," *IEEE Commun. Mag*., vol. 38, no. 1, pp. 66-72, January 2000.

[4] "Metropolitan DWDM: A Dynamically Configurable Ring for the KomNet Field Trial in Berlin," *IEEE Commun. Mag*., vol. 39, no. 2, pp. 106-113, Feb. 2001.

[5] M. Marsan, A. Bianco, E. Leonardi, A. Morabito, and F. Neri, "All-Optical WDM Multi-Rings with Differentiated QoS," *IEEE Commun. Mag*., vol. 37, no. 2, pp. 58-66, Feb.1999.

[6] J. Cai, A. Fumagalli, and I. Chlamtac, "The Multitoken Interarrival Time (MTIT) Access Protocol for Supporting Variable Size Packets Over WDM Ring Network," IEEE *J. Sel. Areas Commun*., vol. 18, no.10, pp. 2094-2104, Oct. 2000.

[7] A. Bononi, "Scaling WDM Slotted Ring Networks," in Proc. *Conf. On Information Sciences and Systems*, Princeton, vol. 1, pp. 659, March 1998.

[8] S. Gemelos, I. White, D. Wonglumsom, K. Shrikhande, T. Ono, and L. Kazovsky, "WDM Metropolitan Area Network Based on CSMA/CA Packet Switching," IEEE Photon. Technol. Letters, vol. 11, no. 11, November 1999, pp. 1512-1514.

[9] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Trans. Networking*, vol.3, no.3, June 1995, pp.226-244.

[10] W. Willinger, and V. Paxson, "Where Mathematics meets the Internet," *Notices of the American Mathematical Society*, vol. 45, no. 8, Sept. 1998, pp. 961-970.

[11] M. Marsan, A. Bianco, E. Leonardi, A. Morabito, and F. Neri, "All-Optical WDM Multi-Rings with Differentiated QoS," *IEEE Commun. Mag*., vol. 37, no. 2, Feb.1999, pp. 58-66.

[12] E. Modiano and R. Barry, "A Novel Medium Access Control Protocol for WDM-based LAN's and Access Networks Using a Master/Slave Scheduler," *J. Lightwave Technol.*, vol. 18, no. 4, pp. 461-468, April 2000.

[13] R. Jain, *The Art Of Computer Systems Performance Analysis*, Wiley, 1991, pp. 474.

Appendix 2:

C code of the network simulator

Appendix 1.1 Makefile

```
#
#  Christophe Jelger - MPhil Student
#  University of Wales Swansea
#
#  Makefile for WDM Simulator
#

#  Choose the compiler (cc or gcc)

CC = gcc

CFLAGS = -O

OBJS = fiber.o wavelength.o packet.o buffer.o node.o dac.o log.o wasp.o init.o display.o result.o

TEST_OBJS = packet.o

simulator : $(OBJS) simulator.o
	$(CC) $(CFLAGS) $(OBJS) -lm simulator.o -o simulator

test : $(TEST_OBJS) test.o
	$(CC) $(CFLAGS) $(TEST_OBJS) -lm test.o -o test


# The object files

fiber.o : fiber/fiber.c fiber/fiber.h
	$(CC) $(CFLAGS) -c fiber/fiber.c

wavelength.o : wavelength/wavelength.c wavelength/wavelength.h
	$(CC) $(CFLAGS) -c wavelength/wavelength.c

packet.o : packet/packet.c packet/packet.h
	$(CC) $(CFLAGS) -c packet/packet.c

buffer.o : buffer/buffer.c buffer/buffer.h
	$(CC) $(CFLAGS) -c buffer/buffer.c

node.o : node/node.c node/node.h
	$(CC) $(CFLAGS) -c node/node.c

dac.o : dac/dac.c dac/dac.h main.h
	$(CC) $(CFLAGS) -c dac/dac.c

log.o : log/log.c log/log.h
	$(CC) $(CFLAGS) -c log/log.c

wasp.o : wasp/wasp.c wasp/wasp.h
	$(CC) $(CFLAGS) -c wasp/wasp.c

init.o : init/init.c init/init.h
	$(CC) $(CFLAGS) -c init/init.c

display.o : display/display.c display/display.h
	$(CC) $(CFLAGS) -c display/display.c

result.o : result/result.c result/result.h
	$(CC) $(CFLAGS) -c result/result.c
```

```
simulator.o : simulator.c global.h main.h
	$(CC) $(CFLAGS) -c simulator.c

test.o : test.c global.h main.h
	$(CC) $(CFLAGS) -c test.c


# to delete the object files and the exe files

clean:
	rm -f simulator simulator.o $(OBJS)

cleantest:
	rm -f test test.o
```

Appendix 1.2 C modules: simulator.c (main)

```
/************************************
```

Christophe Jelger
MPhil Student
University of Wales Swansea

WDM Simulator - Version 1.02

Simulator.c – MAIN module

```
************************************/

#include "global.h"
#include "error.h"
#include "wavelength/wavelength.h"
#include "packet/packet.h"
#include "node/node.h"
#include "buffer/buffer.h"
#include "fiber/fiber.h"
#include "dac/dac.h"
#include "log/log.h"
#include "wasp/wasp.h"
#include "init/init.h"
#include "display/display.h"
#include "result/result.h"
#include "predict/predict.h"

#include <stdio.h>
#include <string.h>


/***********  GLOBAL VARIABLES **************/


fiber fiber1;                                   /* The physical layer */

wavelength wave[NB_WV];                         /* array of wavelengths */

wc_counter load_counter[NB_NODES][NB_WV];       /* array of wavelengths load counters */

FILE* traffic_file[NB_NODES];           /* array of traffic sources files */

FILE *wv_snapshot;

FILE* buffer_file[NB_NODES];            /* array of files to log buffers load */

FILE* rate_file[NB_NODES];              /* array of files to log nodes throughput */

FILE* delay_file[NB_NODES];             /* array of files to log pkt delays in buffers */

FILE* linkrate_file[NB_NODES];                  /* array of files to log access link rate */
```

```
FILE* pkts_dropped_file[NB_NODES];      /* array of files to log PKTS DROPPED stats */

FILE* results_file[NB_NODES];           /* array of files for final results stats */

FILE* wcounters_file[NB_NODES];                 /* array of counters log files */

FILE* wcounters_by_slots_file[NB_NODES];        /* array for real-time log of w_counters */

FILE* receiver_collisions_stats;        /* stats for receiver collisions */

packet* node_current[NB_NODES];                 /* array of pointers to wavelength packets */

buffer node_buffer[NB_NODES];                   /* array of node buffers */

buffer node_tmp_buffer[NB_NODES];               /* array of temp buffers for traffic */

node node_array[NB_NODES];              /* the array of nodes */

int node_tx_pkts[NB_NODES];             /* array of tx pkts per frame per node */

int node_rx_from_link[NB_NODES];                /* array of rx pkts from ACCESS LINK per node */

int node_allow_tx[NB_NODES];            /* a flag to allow or not transmission */

float node_delay[NB_NODES];             /* used to compute average queuing delay time */

float node_global_delay[NB_NODES];              /* used to compute average TOTAL transmission time */

int pkts_rx[NB_NODES];

int node_buffer_load[NB_NODES];                 /* used to compute the average buffer load */

result final_results[NB_NODES];                 /* for automatic stat results */

long collisions_stats_array[5];

double global_time = 0;                         /* the GLOBAL time in us */
double frame_time = 0;                          /* used to display items per frame time */
int rotation_counter = 0;                       /* used by the wavelengths LOAD counters */

double frame_counter = 0;

matrix node1_matrix;                    /* temp test */

/***********  END GLOBAL VARIABLES **************/


int main(void)
{

        int command = 1;                        /* for user command */
        double target_time = 0;                 /* how long last the simulation ? */
        int node_init = 0;

        int log_wcounters_flag = 0;     /* allow (=1) or not the real-time log */

        int test;
        double precision = 691.2;                               /* the precision for RATE output files */
        double pkt_dropped_precision = 691.2 ;  /* the precision for PKTS dropping stats */

        char traffic_file_name[32];
        char buffer_file_name[] = "../../results/buffer_node";
        char rate_file_name[] = "../../results/rate_node";
        char delay_file_name[] = "../../results/delay_node";
        char linkrate_file_name[] = "../../results/linkrate_node";
        char pkts_dropped_file_name[] = "../../results/pkts_dropped_node";
        char results_file_name[] = "../../results/results_node";
        char wcounters_file_name[] = "../../results/wcounters_node";
        char wcounters_by_slots_file_name[]="../../results/Wcounters_by_slots_node";
        char receiver_collisions[]="../../results/receiver_collisions";

        printf("\n\nEnter the traffic file name : ");
        scanf("%s", &traffic_file_name);

        /* we open the traffic files */
```

```
log_open_files( traffic_file_name, traffic_file, "r");

/* we open the buffer log files */
log_open_files( buffer_file_name, buffer_file, "w");

/* we open the THROUGHPUT log files */
log_open_files( rate_file_name, rate_file, "w");

/* we open the DELAY log files */
log_open_files( delay_file_name, delay_file, "w");

/* we open the LINK RATE  log files */
log_open_files( linkrate_file_name, linkrate_file, "w");

/* we open the PKTS DROPPED log files */
log_open_files( pkts_dropped_file_name, pkts_dropped_file, "w");

/* we open the FINAL RESULTS log files */
log_open_files( results_file_name, results_file, "w");

/* we open the WAVELENGTH COUNTERS log files */
log_open_files( wcounters_file_name, wcounters_file, "w");

/* we open the WAVELENGTH COUNTERS log files */
log_open_files( wcounters_by_slots_file_name, wcounters_by_slots_file, "w");

/* we open the RECEIVER COLLISION STATISTICS FILE */
receiver_collisions_stats = fopen(receiver_collisions, "w");

for( node_init=0; node_init<5; node_init++ )
        collisions_stats_array[node_init] = 0;

/* Fiber */
init_fiber();

/* We initialise the NB_WV wavelengths */
init_wave();

/* We initialize the NB_NODES nodes - one loop for each node */
init_node();

/* we initialize the array of load counters */
init_counter();

/* we initialise the matrix */
mx_init_zero(&node1_matrix);
mx_init_XtX(&node1_matrix);
mx_compute_det_XtX(&node1_matrix);
mx_compute_Adj_XtX(&node1_matrix);
mx_compute_Inv_XtX(&node1_matrix);

while(command != 0)
{
        printf("\n\n-------------------------------------------------------");
        printf("\nIP over WDM simulator ... version 0.15");
        printf("\n\nUniversity of Wales Swansea");
        printf("\nDepartment of Electrical and Electronic Engineering");
        printf("\n-------------------------------------------------------\n");

        printf("\n 0. Exit");
        printf("\n 1. Display physical layer (fiber + wavelength)");
        printf("\n 2. Display node parameters (node + buffers)");
        printf("\n 3. Create wavelength snapshot");
        printf("\n 4. Read traffic files (initial sequence)");
        printf("\n 5. Transfer traffic from TEMP buffers");
        printf("\n 6. Transmit to wavelength");
        printf("\n 7. Rotate ...");
        printf("\n 8. Steps 4,5,6 and 7 a given number of time");
        printf("\n 9. Precision parameter for RATE output files");
        printf("\n10. Display load counter");
        printf("\n11. New wavelength for node ...");
        printf("\n12. Allow Real-time log of w_counters");
        printf("\n13. Display MATRIX ...");
        printf("\n\nEnter your choice : ");
```

```
scanf("%d", &command);

switch( command )
{
        case 0 : break;

        case 1 :
        {       display_phy();
                break;      }

        case 2 :
        {       display_node();
                break;      }

        case 3 :
        {       wv_snapshot = fopen("../results/wv_snapshot", "w");
                printf("\nEnter the wavelength ID : ");
                scanf("%d", &test);
                wv_display_to_file( wave+test-1, wv_snapshot);
                fclose( wv_snapshot );
                break;      }

        case 4 :
        {       init_buffers();
                break;      }

        case 5 :
        {       dac_transfer_traffic();         /* transfer to buffers */
                break;      }

        case 6 :
        {       dac_transmit_to_wave();                 /* transmit to wavelength */
                break;      }

        case 7 :
        {       dac_rotate();
                global_time += 4.8;
                break;      }

        case 8 :
        {
                printf("\nEnter the target time (in us) : ");
                scanf("%lf", &target_time);

                init_buffers();

                while( global_time < target_time )
                {
                global_time += 4.8;             /* time increases */

                dac_transfer_traffic();         /* traffic from TEMP to BUFFERS */
                log_buffers_to_files();                 /* log buffers LOAD */

                dac_remove_from_wave();                 /* Rx traffic */
                dac_transmit_to_wave();                 /* Tx traffic */

                wc_update_load();               /* update counters */

                /* the matrix XtY is also updated by wc_update_load() */

                if( log_wcounters_flag == 1)            /* real-time log */
                        log_wcounters_by_slots();       /* of w_counters */

                log_rates_to_files(precision);  /* log nodes THROUGHPUT */

                dac_rotate();                           /* network rotation */

                }

                break;
        }

        case 9 :
        {
                printf("\nEnter your value (in us - default = 691.2) : ");
```

```
                                        scanf("%lf", &precision);
                                        break;
                                }

                                case 10 :
                                {
                                        display_counter();
                                        break;
                                }

                                case 11 :
                                {
                                        wasp_manual_chg_wave();
                                        break;
                                }

                                case 12 :
                                {
                                        log_wcounters_flag = 1;
                                        printf("\nReal-time log for w_counters is ON\n\n");
                                        break;
                                }

                                case 13 :
                                {
                                        mx_display_XtX_and_XtY(&node1_matrix);
                                        break;
                                }

                        }
                }

                res_compute( final_results, frame_counter-1 );

                res_display_to_file( final_results, results_file );

                for(node_init=0; node_init < 5; node_init++)
                        fprintf(receiver_collisions_stats, "%ld ", collisions_stats_array[node_init]);

                /* we close all the files : TRAFFIC, BUFFER, RATE, DELAY, W_COUNTERS */

                log_close_files( traffic_file );
                log_close_files( buffer_file );
                log_close_files( rate_file );
                log_close_files( delay_file );
                log_close_files( linkrate_file );
                log_close_files( pkts_dropped_file );
                log_close_files( results_file );
                log_close_files( wcounters_file );
                log_close_files( wcounters_by_slots_file );
                fclose(receiver_collisions_stats);

                /* free the memory used for the wavelength slots/pkts */

                for(node_init=0; node_init < NB_WV; node_init++)
                        wv_free_table(wave+node_init);

                return 0;
}
```

Appendix 1.3 C modules: main.h

```
/************************************
```

Christophe Jelger
MPhil Student
University of Wales Swansea

WDM Simulator - Version 1.02

Main.h

**********/

```
#ifndef MAIN_H
#define MAIN_H

#include "wavelength/wavelength.h"
#include "fiber/fiber.h"
#include "packet/packet.h"
#include "buffer/buffer.h"
#include "node/node.h"
#include "wasp/wasp.h"
#include "global.h"
#include "result/result.h"
#include "predict/predict.h"


/***********  GLOBAL VARIABLES *************/


extern fiber fiber1;                    /* The physical layer */

extern wavelength wave[NB_WV];                                  /* array of wavelengths */

extern wc_counter load_counter[NB_NODES][NB_WV];            /* array of wavelength load counters */

extern FILE* traffic_file[NB_NODES];

extern FILE *wv_snapshot;                               /* file for wavelength snapshot */

extern FILE* buffer_file[NB_NODES];                     /* array of files to log buffers load */

extern FILE* rate_file[NB_NODES];                       /* array of files to log nodes throughput */

extern FILE* delay_file[NB_NODES];                      /* array of files to log pkt delays in buffers */

extern FILE* linkrate_file[NB_NODES];                   /* array of files to log access link rate */

extern FILE* pkts_dropped_file[NB_NODES];               /* array of files to log PKTS DROPPED stats */

extern FILE* results_file[NB_NODES];                    /* array of files for final results stats */

extern FILE* wcounters_file[NB_NODES];                  /* array of counters log files */

extern FILE* wcounters_by_slots_file[NB_NODES];     /* array for real-time log of w_counters */

extern FILE* receiver_collisions_stats;                 /* stats for receiver collisions */

extern packet* node_current[NB_NODES];                  /* array of pointers to wavelength */

extern buffer node_buffer[NB_NODES];                    /* node buffers */

extern buffer node_tmp_buffer[NB_NODES];                /* temp buffers for traffic */

extern node node_array[NB_NODES];                       /* the nodes */

extern int node_tx_pkts[NB_NODES];                      /* tx pkts per frame */

extern int node_rx_from_link[NB_NODES];  /* array of rx pkts from ACCESS LINK per node */

extern int node_allow_tx[NB_NODES];                     /* a flag to allow or not transmission */

extern float node_delay[NB_NODES];                      /* used to compute average delay time */

extern float node_global_delay[NB_NODES];               /* used to compute average TOTAL transmission time */

extern int pkts_rx[NB_NODES];

extern int node_buffer_load[NB_NODES];                  /* used to compute the average buffer load */

extern result final_results[NB_NODES];                  /* for automatic stat results */

extern long collisions_stats_array[5];
```

```
extern double global_time;                          /* time in us ... EXTRA CARE !!! */

extern double frame_time;                           /* used to display items per frame time */

extern int rotation_counter;                /* used by the wavelengths LOAD counters */

extern double frame_counter;

extern matrix node1_matrix;                         /* temp test */

/***********  END GLOBAL VARIABLES *************/


#endif /* MAIN_H */
```

Appendix 1.4 C modules: global.h

```
/***********************************

            Christophe Jelger
            MPhil Student
            University of Wales Swansea

            WDM Simulator - Version 1.02

            Global.h

************************************/

#ifndef GLOBAL_H
#define GLOBAL_H


/* The number of nodes in the network */
#define NB_NODES   16

/* The number of wavelengths per fiber */
#define NB_WV                   4

/* To be used temporarily to simplify the implementation */
#define NB_SLOTS_INT            144


/* The total fiber/ring length (in meters) */
#define FIBER_LENGTH            138240

/* The light velocity in the fiber (in m/s) */
#define LIGHT_VELOCITY      200000000

/* The propagation delay of the fiber */
#define FIBER_DELAY             FIBER_LENGTH/LIGHT_VELOCITY

/* The capacity (in packets) of a buffer */
#define BF_MAX_PACKET_STORE         100

/* The size of a slot (in bits) */
#define SLOT_SIZE               12000

/* The rate for ONE wavelength (in b/s) */
#define WAVELENGTH_RATE             2.5e+9

/* the time (in us) for one slot */
#define SLOT_TIME  SLOT_SIZE/WAVELENGTH_RATE*1e+6

/* The number of slots by wavelength */
#define NB_SLOTS    FIBER_DELAY*WAVELENGTH_RATE/SLOT_SIZE
```

```
/* Misc */
#define STATUS_ON 1
#define STATUS_OFF              0
#define CLK_WISE    1
#define ANTI_CLK_WISE         -1


#endif /* GLOBAL_H */
```

Appendix 1.5 C modules: error.h

```
/***********************************
```

Christophe Jelger
MPhil Student
University of Wales Swansea

WDM Simulator - Version 1.02

Error.h

```
************************************/

#ifndef ERROR_H
#define ERROR_H

/* Global defintion */

#define ERR_OK                  0

/* Definitions for BUFFER */

#define     ERR_BF_FULL         1
#define ERR_BF_EMPTY            2


/* Definitions for PACKET */

#define ERR_PK_SET              1
#define ERR_PK_EMPTY            2


/* Definitions for NODES and DAC*/

#define TX_ALLOWED              1
#define TX_NOT_ALLOWED    2

/* Definitions for MATRIX */

#define ERR_DET_NULL            1

#endif /* ERROR_H */
```

Appendix 1.6 C modules: buffer.h and buffer.c

```
/***********************************
```

Christophe Jelger
MPhil Student
University of Wales Swansea

WDM Simulator - Version 1.02

Buffer.h

```
**********************************/

#ifndef BUFFER_H
#define BUFFER_H

#include "../global.h"
#include "../packet/packet.h"

#include <stdio.h>

typedef struct
{
        int bf_id;
        int bf_capacity;
        int bf_peak_load;
        int bf_average_load;
        int bf_current_load;
        int bf_dropped_packets;
        packet* bf_origin;
        packet* bf_current;
        packet bf_table[BF_MAX_PACKET_STORE];

} buffer;

extern int bf_display(buffer*);

extern int bf_init(buffer*, int);

extern int bf_add_pk(buffer*, packet*);

extern int bf_rmv_pk(buffer*);

extern int bf_move_pk(buffer*, packet*);

extern int bf_move_bf(buffer*, buffer*);

extern int bf_display_to_file(buffer*, FILE*);

#endif /* BUFFER_H */



/**********************************

        Christophe Jelger
        MPhil Student
        University of Wales Swansea

        WDM Simulator - Version 1.02

        Buffer.c

**********************************/

#include "buffer.h"
#include "../global.h"
#include "../error.h"
#include <stdio.h>

int bf_init(buffer* bf, int id)
{
        bf->bf_id = id;
        bf->bf_capacity = BF_MAX_PACKET_STORE;
        bf->bf_peak_load = 0;
        bf->bf_average_load = 0;
        bf->bf_current_load = 0;
        bf->bf_dropped_packets = 0;
        bf->bf_origin = bf->bf_table;
        bf->bf_current = bf->bf_table;
        return ERR_OK;
}

int bf_display(buffer* bf)
{
```

```
		printf("\nBuffer parameters :");
		printf("\nID = %d", bf->bf_id);
		printf("\nBuffer capacity = %d packets", bf->bf_capacity);
		printf("\nCURRENT_LOAD = %d packets", bf->bf_current_load);
		printf("\nAVERAGE_LOAD = %d packets", bf->bf_average_load);
		printf("\nPEAK_LOAD = %d packets", bf->bf_peak_load);
		printf("\nPACKETS DROPPED = %d packets\n", bf->bf_dropped_packets);
		return ERR_OK;
}

int bf_add_pk(buffer* bf, packet* pk)
{
		if( bf->bf_current_load == BF_MAX_PACKET_STORE )
		{
			bf->bf_dropped_packets++;	/* Buffer full - packet is dropped */
			return ERR_BF_FULL;
		}
		else
		{
			bf->bf_current_load++;			/* One more packet in the buffer */
			pk_copy(pk, bf->bf_current);

			/* Move the current position MODULO size of buffer*/

			if( (bf->bf_current - bf->bf_table) < (BF_MAX_PACKET_STORE - 1) )
				bf->bf_current++;
			else
				bf->bf_current = bf->bf_table;

			return ERR_OK;
		}

}

int bf_rmv_pk(buffer* bf)
{
		if( bf->bf_current_load == 0)		/* Buffer empty - return error */
			return ERR_BF_EMPTY;
		else
		{
			bf->bf_current_load--;			/* One packet removed from the buffer */

			/* Move the start position MODULO size of buffer*/

			if( (bf->bf_origin - bf->bf_table) < (BF_MAX_PACKET_STORE - 1) )
				bf->bf_origin++;
			else
				bf->bf_origin = bf->bf_table;

			return ERR_OK;
		}
}

int bf_move_pk(buffer* bf, packet* pk)
{
		if( bf->bf_current_load == 0)		/* Buffer empty - return error */
			return ERR_BF_EMPTY;
		else
		{
			pk_copy( bf->bf_origin, pk);
			bf_rmv_pk( bf );

			return ERR_OK;
		}
}

int bf_move_bf(buffer* src, buffer* dest)
{
		if( src->bf_current_load == 0)		/* SRC Buffer empty - return error */
			return ERR_BF_EMPTY;
		else
		{
			bf_add_pk( dest, src->bf_origin );		/* move SRC origin pk to DEST */
			bf_rmv_pk( src );				/* remove SRC origin pk */
			return ERR_OK;
```

```
		}
}

int bf_display_to_file(buffer* bf, FILE* bf_file)
{
		fprintf( bf_file, "%d\n", bf->bf_current_load);

		return ERR_OK;
}
```

Appendix 1.7 C modules: dac.h and dac.c

```
/************************************
```

Christophe Jelger
MPhil Student
University of Wales Swansea

WDM Simulator - Version 1.02

Dac.h

```
************************************/

#ifndef DAC_H
#define DAC_H

#include "../buffer/buffer.h"

extern int dac_read_traffic_init(buffer*, int);   /* read the traffic files at time = 0 */

extern int dac_check_traffic(buffer*);                  /* ensure temp traffic buffers are not empty */

extern int dac_transfer_traffic(void);                  /* transfer traffic from temp to buffers */

extern int dac_transmit_to_wave(void);                  /* transmit to wavelength */

extern int dac_rotate(void);                            /* rotate the network */

extern int dac_remove_from_wave(void);                  /* reception procedure */

#endif /* DAC_H */



/************************************
```

Christophe Jelger
MPhil Student
University of Wales Swansea

WDM Simulator - Version 1.02

Dac.c

```
************************************/

#include "dac.h"
#include "../error.h"
#include "../global.h"
#include "../main.h"
#include "../wavelength/wavelength.h"
#include "../packet/packet.h"
#include "../buffer/buffer.h"
#include "../log/log.h"
#include "../predict/predict.h"

#include <stdio.h>
```

```
int dac_read_traffic_init(buffer* bf, int nb_pkts)          /* read nb_pkts packets from traffic files */
{                                                                                   /* and stores them to TEMP buffer */
        int i;
        packet pk_temp;
        int src_id, dest_id;
        double time;

        for(i=0; i<nb_pkts; i++)
        {
                /* read in traffic_file ARRAY at position BF_ID%11*/

                fscanf( traffic_file[(bf->bf_id)%101], "%d %d %lf", &src_id, &dest_id, &time );
                pk_init( &pk_temp, src_id, dest_id, time );

                /* transfer the pkt to TEMP buffer */

                bf_add_pk( bf, &pk_temp );
        }

        return ERR_OK;
}

int dac_check_traffic(buffer* bf)   /* ensures TEMP traffic buffers are not empty */
{
        if( (bf->bf_current_load) < (BF_MAX_PACKET_STORE/2) )
                dac_read_traffic_init( bf, (int)(BF_MAX_PACKET_STORE/2) );


        return ERR_OK;
}

int dac_transfer_traffic(void)              /* transfers pkts from TEMP buffers to NODE */
{                                                                                   /* buffers at a given time */
        int i;

        for(i=0; i<NB_NODES; i++)       /* for each node */
        {
                if( (( node_tmp_buffer+i )->bf_origin)->pk_time <= global_time )
                {
                        /* transfer the origin packet if pkt time <= GLOBAL TIME (above) */

                        bf_move_bf( node_tmp_buffer+i, node_buffer+i);

                        node_rx_from_link[i]++;                     /* +1 pkt received from access link */

                        /* ensures the buffer is not empty */

                        dac_check_traffic( node_tmp_buffer+i );
                }
        }

        return ERR_OK;
}

int dac_transmit_to_wave()                          /* transmit from tx_buffer to wavelength */
{
        int i;

        for(i=0; i<NB_NODES; i++)       /* for each node */
        {
                if( (node_buffer+i)->bf_current_load != 0 && pk_test_empty(node_current[i]) ==
                ERR_PK_EMPTY && node_allow_tx[i] == TX_ALLOWED)
                {
                        /* transmit pkt to wave if LOAD != 0 AND slot = EMPTY */

                        bf_move_pk( node_buffer+i, node_current[i] );

                        log_delays_to_files(i);             /* log the time the pkts spent in buffer */

                        /* one more pkt was transmitted ;-) */

                        node_tx_pkts[i]++;
                }
```

```
	}

	return ERR_OK;
}

int dac_rotate()		/* rotate the node current pointer to the wavelength */
{
	int i,j;

	for(i=0; i<NB_NODES; i++)	/* each node "rotates" around its current wavelength */
	{
		node_current[i] = wv_rotate( (node_array+i)->nd_wave, node_current[i] );
	}

	rotation_counter++;			/* for wavelength LOAD counters */

	if( rotation_counter == NB_SLOTS_INT )
	{

		/* compute REAL load */

		for(i=0; i<NB_NODES; i++)	/* each node */
		{
			for(j=0; j < NB_WV ; j++)		/* for each wavelength */
			{
				wc_compute_load( &load_counter[i][j] );
			}
		}

		rotation_counter = 0;

		frame_counter++;

		/* temp : compute predicted load */

		mx_compute_final(&node1_matrix);

		log_wcounters();			/* log the w_counters values */



	}

	return ERR_OK;
}

int dac_remove_from_wave()	/* each node removes its own pkts from the wavelength */
{
	int i,j,offset;
	int collisions;
	packet* pk_current;

	for(i=0; i<NB_NODES; i++)	/* for each node */
	{
		/* allow each node to TX (init process) */

		node_allow_tx[i] = TX_ALLOWED;

		/* set collision stat to 0 */

		collisions = 0;

		/* the destination stripping */

		offset = ( node_current[i] - ((node_array+i)->nd_wave)->wv_table );

		for(j=0; j<NB_WV; j++)
		{
			pk_current = ( (wave+j)->wv_table + offset );

			if( pk_test_empty(pk_current) == ERR_PK_SET )		/* if pkt set */
			{
				if( pk_current->pk_dest_id == (node_array+i)->nd_id )
				{
					collisions++;
```

```
                                                if( collisions < 2 )
                                                {
                                                        node_global_delay[i] += (global_time-(pk_current-
>pk_time));

                                                        pkts_rx[i]++;

                                                        pk_set_empty( pk_current );

                                                        if( (node_array+i)->nd_wave == (wave+j) )
                                                        {
                                                                node_allow_tx[i] = TX_NOT_ALLOWED;
                                                        }
                                                }
                                        }
                                }
                        }

                        collisions_stats_array[collisions]++;
                }

                return ERR_OK;
}
```

Appendix 1.8 C modules: display.h and display.c

```
/************************************

                Christophe Jelger
                MPhil Student
                University of Wales Swansea

                WDM Simulator - Version 1.02

                Display.h

************************************/

#ifndef DISPLAY_H
#define DISPLAY_H


extern int display_phy(void);                   /* display physical layer (fiber+wave) */

extern int display_node(void);                  /* display the nodes */

extern int display_counter(void);   /* display the load counters */


#endif /* DISPLAY_H */



/************************************

                Christophe Jelger
                MPhil Student
                University of Wales Swansea

                WDM Simulator - Version 1.02

                Display.c

************************************/

#include "../error.h"
#include "../global.h"
#include "../main.h"
```

```
#include "../wavelength/wavelength.h"
#include "../node/node.h"
#include "../buffer/buffer.h"
#include "../fiber/fiber.h"
#include "../wasp/wasp.h"

#include <stdio.h>


int display_phy(void)
{
	int wave_id;

	printf("\nEnter the wavelength ID : ");
	scanf("%d", &wave_id);

	wave_id--;				/* we want the offset in wave array */

	fb_display(&fiber1);		/* displays the fiber parameters */
	wv_display(wave + wave_id);	/* displays the wavelength parameters */

	return ERR_OK;
}

int display_node(void)		/* displays a node according to user selection */
{
	int node_id;

	printf("\nEnter the node ID : ");
	scanf("%d", &node_id);

	node_id--;  /* we want the offset in the different arrays */

	nd_display( node_array + node_id );				/* node parameters */
	bf_display( node_buffer + node_id );				/* buffer parameters */
	bf_display( node_tmp_buffer + node_id );	/* TEMP buffer parameters */

	return ERR_OK;
}

int display_counter(void)
{
	int counter_id;
	int node_id, wave_id;

	printf("\nEnter the counter ID : ");
	scanf("%d", &counter_id);

	wave_id = counter_id % 10;

	node_id = (counter_id - wave_id)/10;

	wc_display( &load_counter[node_id-1][wave_id-1] );

	return ERR_OK;
}
```

Appendix 1.9 C modules: fiber.h and fiber.c

```
/************************************
```

Christophe Jelger
MPhil Student
University of Wales Swansea

WDM Simulator - Version 1.02

Fiber.h

```
************************************/
```

```
#ifndef FIBER_H
#define FIBER_H

typedef struct
{
        char fb_name[32];
        int fb_id;
        long fb_length;
        float fb_delay;
        int  fb_direction;
} fiber;

extern int fb_display(fiber*);
extern int fb_init(fiber*, char*, int, long, float, int);

#endif /* FIBER_H */
```

```
/***********************************

        Christophe Jelger
        MPhil Student
        University of Wales Swansea

        WDM Simulator - Version 1.02

        Fiber.c

************************************/

#include "fiber.h"
#include <stdio.h>
#include <string.h>

int fb_init(fiber *fb, char* name, int id, long length, float delay, int direction)
{
        strcpy( fb->fb_name, name);
        fb->fb_id = id;
        fb->fb_length = length;
        fb->fb_delay  = delay;
        fb->fb_direction = direction;
        return 0;
}

int fb_display(fiber *fb)
{
        printf("\nFiber parameters :");
        printf("\nNAME = %s", fb->fb_name);
        printf("\nID = %d", fb->fb_id);
        printf("\nLENGTH = %ld Meters", fb->fb_length);
        printf("\nDELAY  = %f Seconds", fb->fb_delay);
        printf("\nDIRECTION = %d\n", fb->fb_direction);
        return 0;
}
```

Appendix 1.10 C modules: init.h and init.c

```
/***********************************

        Christophe Jelger
        MPhil Student
        University of Wales Swansea

        WDM Simulator - Version 1.02

        Init.h
```

```
**********************************/

#ifndef INIT_H
#define INIT_H

extern int init_fiber(void);

extern int init_wave(void);

extern int init_node(void);

extern int init_counter(void);

extern int init_buffers(void);                  /* first load of traffic buffers */

#endif /* INIT_H */



/**********************************
```

Christophe Jelger
MPhil Student
University of Wales Swansea

WDM Simulator - Version 1.02

Init.c

```
**********************************/

#include "../error.h"
#include "../global.h"
#include "../main.h"
#include "../fiber/fiber.h"
#include "../wavelength/wavelength.h"
#include "../log/log.h"
#include "../node/node.h"
#include "../buffer/buffer.h"
#include "../dac/dac.h"
#include "../result/result.h"

#include <stdio.h>


int init_fiber()
{
        fb_init(&fiber1, "Fiber One", 1, FIBER_LENGTH , (float)FIBER_DELAY , CLK_WISE );

        return ERR_OK;
}

int init_wave()
{
        int init;
        int wave_id=1;
        char tmp_string[32];
        char tmp2[3];
        char wave_name[]="Wavelength number ";

        for(init=0; init < NB_WV ; init++)
        {
                sprintf( tmp2, "%d", wave_id);

                log_return_file_name( wave_name, tmp_string, tmp2 );

                wv_init(wave+init, tmp_string, (init+1), WAVELENGTH_RATE, STATUS_ON, SLOT_SIZE,
                (float)SLOT_TIME, (float)NB_SLOTS, &fiber1);
                wv_init_table(wave+init);

                wave_id++;
        }
```

```
	return ERR_OK;
}

int init_node()
{
	int init;
	int node_id=1;
	char tmp_string[32];
	char tmp2[3];
	char node_name[]="Node number ";
	int i,j;

	int node_per_wave = NB_NODES/NB_WV;		/* how many nodes per wavelength ? */
	int count_per_wave = 0;				/* nb of nodes assigned per wavelength */
	int wv_offset = 0;				/* offset in wavelength array */

	for(init=0; init < NB_NODES; init++)
	{
		/* initialize the pointer to the wavelength with offset (node position) */

		(node_current[init]) = (wv_reset_current(wave+wv_offset) + (int)NB_SLOTS_INT/NB_NODES*init );


		/* initialise the node buffers : POSITION(in buffer array) + ID */

		bf_init( (node_buffer + init), (init + 1) );

		bf_init( (node_tmp_buffer + init), (init + 101) );


		/* initialize the nodes' parameters */

		sprintf( tmp2, "%d", node_id);

		log_return_file_name( node_name, tmp_string, tmp2 );

		nd_init( (node_array + init), tmp_string, (init + 1), wave+wv_offset,(node_buffer + init) );


		/* initialise the count of transmitted pkts per frame */

		node_tx_pkts[init] = 0;
		node_rx_from_link[init] = 0;	/* idem for rx pkts from access link */
		node_delay[init] = 0;
		node_global_delay[init] = 0;
		pkts_rx[init] = 1;

		/* init the result array */

		res_init(final_results + init);

		/* allow each node to transmit */

		node_allow_tx[init] = TX_ALLOWED;

		node_id++;

		count_per_wave++;	/* one node added to wavelength N */

		if(count_per_wave == node_per_wave)	/* nb of nodes per wave achieved */
		{
			count_per_wave = 0;
			wv_offset++;			/* next wavelength */
		}
	}

	return ERR_OK;
}

int init_counter()
{
	int i,j;

	for(i=0; i < NB_NODES; i++)
	{
```

```
                for(j=0; j < NB_WV ; j++)
                {
                        wc_init( &load_counter[i][j], (i+1)*10+j+1, wave+j);
                }
        }

        return ERR_OK;
}

int init_buffers()              /* loads TEMP buffers with traffic (BF_MAX pkts) */
{
        int i;                          /* for each node */

        for(i=0; i<NB_NODES; i++)
                dac_read_traffic_init( node_tmp_buffer + i, BF_MAX_PACKET_STORE);

                /* we read BF_MAX pkts from the traffic file */


        return ERR_OK;
}
```

Appendix 1.11 C modules: log.h and log.c

```
/************************************

        Christophe Jelger
        MPhil Student
        University of Wales Swansea

        WDM Simulator - Version 1.02

        Log.h

************************************/

#ifndef LOG_H
#define LOG_H

#include <stdio.h>

extern int log_return_file_name( char*, char*, char* );     /* creates file "extensions" */

extern int log_buffers_to_files(void);                      /* log buffers load to files */

extern int log_rates_to_files(double);                      /* log the nodes throughput to files */

extern int log_delays_to_files(int);                        /* log the delays in us - pkts time in buffers */

extern int log_open_files( char*, FILE**, char*);           /* open file names for each nodes */

extern int log_close_files( FILE** );                       /* close all files with prefix (arg) */

extern int log_wcounters();                                 /* log w_counters values */

extern int log_wcounters_by_slots();                        /* log real-time w_counters values*/

#endif /* LOG_H */




/************************************

        Christophe Jelger
        MPhil Student
```

University of Wales Swansea

WDM Simulator - Version 1.02

Log.c

**********/

```
#include "log.h"
#include "../error.h"
#include "../buffer/buffer.h"
#include "../main.h"
#include "../result/result.h"
#include "../predict/predict.h"
#include <stdio.h>
#include <string.h>


int log_return_file_name( char* base, char* temp, char* suffix)
{
        strcpy( temp, base);

        strcat( temp, suffix );

        return ERR_OK;
}


int log_buffers_to_files(void)      /* prints buffers information to BUFFER files */
{
        int i;

        /* for each node */

        for(i=0; i < NB_NODES; i++)
                /* bf_display_to_file( node_buffer + i, buffer_file[i] ); */

                node_buffer_load[i] += ( (node_buffer + i)->bf_current_load );

                /* we print the buffer load in the corresponding file */

        return ERR_OK;
}


int log_rates_to_files(double precision)        /* prints tx pkts per frame to RATE files */
{
        int i;

        if( (global_time - frame_time) >= precision )
        {
                frame_time += precision;            /* the frame counter is "updated" */

                for(i=0; i < NB_NODES; i++)     /* for each node */
                {
                        /* we print the nb of pkts transmitted to the corresponding file */

                        fprintf( rate_file[i], "%d\n", node_tx_pkts[i] );

                        fprintf( linkrate_file[i], "%d\n", node_rx_from_link[i] );

                        fprintf( pkts_dropped_file[i], "%d\n",
                        (node_array+i)->nd_tx_buffer->bf_dropped_packets );

                        fprintf( delay_file[i], "%.1lf\n", (float)node_delay[i]/NB_SLOTS_INT );

                        fprintf( buffer_file[i], "%.1f\n", (float)node_buffer_load[i]/NB_SLOTS_INT);

                        res_add_to(final_results+i, node_rx_from_link[i], node_tx_pkts[i],
                        (double)node_delay[i]/NB_SLOTS_INT, (double)node_buffer_load[i]/NB_SLOTS_INT,
                        (node_array+i)->nd_tx_buffer->bf_dropped_packets,
                        (double)node_global_delay[i]/pkts_rx[i] );

                        (node_array+i)->nd_tx_buffer->bf_dropped_packets = 0;
                        node_delay[i] = 0;
```

```
                                node_global_delay[i] = 0;
                                node_buffer_load[i] = 0;
                                node_tx_pkts[i] = 0;
                                node_rx_from_link[i] = 0;
                                pkts_rx[i] = 1;
                        }
                }

                return ERR_OK;
}

int log_delays_to_files(int i)
{
                /* prints the time spent in a buffer by a pkt before TX */

                node_delay[i] += ( global_time - (node_current[i])->pk_time );

                /* fprintf( delay_file[i], "%.1lf\n", (global_time - (node_current[i])->pk_time) ); */

                return ERR_OK;
}

int log_open_files( char* path, FILE** file_array, char* mode)
{
                char tmp_string[64];
                char tmp2[3];
                int file_extension = 1;
                int init;

                for(init=0; init < NB_NODES; init++)
                {
                        sprintf( tmp2, "%d", file_extension);

                        log_return_file_name( path, tmp_string, tmp2 );

                        *(file_array + init) = fopen( tmp_string, mode );

                        file_extension++;
                }

                return ERR_OK;
}

int log_close_files( FILE** file_array )
{
                int init;

                for(init=0; init < NB_NODES; init++)
                {
                        fclose( *(file_array + init) );
                }

                return ERR_OK;
}

int log_wcounters(void)
{
                int i,j;

                for(i=0; i < NB_NODES; i++)                     /* for each node */
                {
                        for(j=0; j < NB_WV; j++)            /* for each wavelength */
                        {
                                /* the load for each counter [NODE][WV] */

                                fprintf( wcounters_file[i], "%.2f %.2f ", load_counter[i][j].wc_load,
                                load_counter[i][j].wc_own_load ,node1_matrix.mx_predicted_load);
                        }

                        fprintf( wcounters_file[i], "\n");      /* an CR char */
                }

                return ERR_OK;
}
```

```
int log_wcounters_by_slots(void)
{
	int i,j;

	for(i=0; i < NB_NODES; i++)		/* for each node */
	{
		for(j=0; j < NB_WV; j++)	/* for each wavelength */
		{
			/* the load for each counter [NODE][WV] */

			fprintf( wcounters_by_slots_file[i], "%d %d ", load_counter[i][j].wc_sum,
			load_counter[i][j].wc_own_sum );
		}

		fprintf( wcounters_by_slots_file[i], "\n");	/* an CR char */
	}

	return ERR_OK;
}
```

Appendix 1.12 C modules: node.h and node.c

```
/************************************
```

Christophe Jelger
MPhil Student
University of Wales Swansea

WDM Simulator - Version 1.02

Node.h

```
************************************/

#ifndef NODE_H
#define NODE_H

#include "../wavelength/wavelength.h"
#include "../buffer/buffer.h"

typedef struct
{
	char nd_name[32];
	int nd_id;
	wavelength* nd_wave;
	buffer* nd_tx_buffer;

} node;

extern int nd_display(node*);

extern int nd_init(node*, char*, int, wavelength*, buffer*);

extern packet* nd_chg_wave(node*, wavelength*, packet*);		/* assign new wavelength to node */

#endif /* NODE_H */



/************************************
```

Christophe Jelger
MPhil Student
University of Wales Swansea

WDM Simulator - Version 1.02

Node.c

```
**********************************/

#include "node.h"
#include "../error.h"
#include "../main.h"
#include <stdio.h>
#include <string.h>

int nd_init(node* nd, char* name, int id, wavelength* wv, buffer* tx_buffer)
{
        strcpy( nd->nd_name, name);
        nd->nd_id = id;
        nd->nd_wave = wv;
        nd->nd_tx_buffer = tx_buffer;
        return ERR_OK;
}

int nd_display(node* nd)
{
        printf("\nNode parameters :");
        printf("\nNAME : %s", nd->nd_name);
        printf("\nID : %d", nd->nd_id);
        printf("\nWavelength(s) %d : %s", (nd->nd_wave)->wv_id, (nd->nd_wave)->wv_name);
        printf("\nTX Buffer ID : %d", (nd->nd_tx_buffer)->bf_id);
        printf("\nOffset : %d\n", node_current[(nd->nd_id)-1] - (nd->nd_wave)->wv_table );
        return ERR_OK;
}

packet* nd_chg_wave(node* nd, wavelength* wv, packet* current)
{
        int offset;

        offset = ( current - (nd->nd_wave)->wv_table);          /* offset from origin */

        nd->nd_wave = wv;                                       /* assign new wave to node */

        return (wv->wv_table + offset);     /* return position in new wavelength */
}
```

Appendix 1.13 C modules: packet.h and packet.c

```
/**********************************

        Christophe Jelger
        MPhil Student
        University of Wales Swansea

        WDM Simulator - Version 1.02

        Packet.h

**********************************/

#ifndef PACKET_H
#define PACKET_H

#include <stdio.h>

typedef struct
{
        int pk_source_id;
        int pk_dest_id;
        double pk_time;

} packet;

extern int pk_display(packet*);

extern int pk_init(packet*, int, int, double);
```

```
extern int pk_copy(packet*, packet*);

extern int pk_test_empty(packet*);

extern int pk_set_empty(packet*);

extern int pk_display_to_file(packet*, FILE*);

#endif /* PACKET_H */



/***********************************

        Christophe Jelger
        MPhil Student
        University of Wales Swansea

        WDM Simulator - Version 1.02

        Packet.c

***********************************/

#include "packet.h"
#include "../error.h"
#include <stdio.h>


/* initialise the packet with src and dest ID and packet creation time */

int pk_init(packet* pk, int source, int dest, double time)
{
        pk->pk_source_id = source;
        pk->pk_dest_id = dest;
        pk->pk_time = time;

        return ERR_OK;
}

int pk_display(packet* pk)
{
        printf("\nPacket parameters :");
        printf("\nSOURCE ID = %d", pk->pk_source_id);
        printf("\nDEST ID = %d", pk->pk_dest_id);
        printf("\nENTRY TIME = %f\n", pk->pk_time);

        return ERR_OK;
}


/* src packet values are copied to dest packet values */

int pk_copy(packet* src, packet* dest)
{
        dest->pk_source_id = src->pk_source_id;
        dest->pk_dest_id = src->pk_dest_id;
        dest->pk_time = src->pk_time;

        return ERR_OK;
}


/* test if a packet is empty or not */

int pk_test_empty(packet* pk)
{
        if(pk->pk_source_id == 0)
                return ERR_PK_EMPTY;
        else
                return ERR_PK_SET;
}
```

```
/* set a packet to EMPTY */

int pk_set_empty(packet* pk)
{
	pk_init(pk, 0, 0, 0);

	return ERR_OK;
}


int pk_display_to_file(packet* pk, FILE* wv_file)
{
	if( pk_test_empty(pk) == ERR_PK_EMPTY )
		fprintf( wv_file, "EMPTY\n");
	else
		fprintf( wv_file, "%d %d %lf\n", pk->pk_source_id, pk->pk_dest_id, pk->pk_time);

	return ERR_OK;
}
```

Appendix 1.14 C modules: result.h and result.c

```
/************************************
```

Christophe Jelger
MPhil Student
University of Wales Swansea

WDM Simulator - Version 1.02

Result.h

```
************************************/

#ifndef RESULT_H
#define RESULT_H

#include <stdio.h>

typedef struct
{
	double link_th;
	double node_th;
	double q_delay;
	double bf_load;
	double pk_dropped;
	double g_delay;
} result;

extern int res_add_to(result*, double, double, double, double, double, double);

extern int res_init(result*);			/* all values to 0 */

extern int res_compute(result*, double);

extern int res_display_to_file(result*, FILE**);

#endif /* RESULT_H */



/************************************
```

Christophe Jelger
MPhil Student
University of Wales Swansea

WDM Simulator - Version 1.02

Result.c

**********/

```
#include "result.h"
#include "../error.h"
#include "../global.h"

int res_init(result* res)
{
	res->link_th = 0;
	res->node_th = 0;
	res->q_delay = 0;
	res->bf_load = 0;
	res->pk_dropped = 0;
	res->g_delay = 0;

	return ERR_OK;
}

int res_add_to(result* res, double lk, double nd, double q, double bf, double pk, double g)
{
	res->link_th += lk;
	res->node_th += nd;
	res->q_delay += q;
	res->bf_load += bf;
	res->pk_dropped += pk;
	res->g_delay += g;

	return ERR_OK;
}

int res_compute(result* res, double div)
{
	int i;

	for(i=0; i < NB_NODES; i++)
	{

		(res+i)->link_th = (res+i)->link_th/div*12000/0.0006912/1000000;
		(res+i)->node_th = (res+i)->node_th/div*12000/0.0006912/1000000;
		(res+i)->q_delay /= div;
		(res+i)->bf_load /= div;
		(res+i)->pk_dropped /= div;
		(res+i)->g_delay /= div;
	}

	return ERR_OK;
}

int res_display_to_file(result* res, FILE** file_array)
{
	int i;

	FILE* all_results;

	all_results = fopen("../../results/All_results", "w");

	for(i=0; i < NB_NODES; i++)
	{

	fprintf( *(file_array+i), "%.2f\n%.2f\n%.2f\n%.2f\n%.2f\n", (res+i)->link_th, (res+i)->node_th,
	(res+i)->q_delay,(res+i)->bf_load, (res+i)->pk_dropped);

	}

	for(i=0; i < NB_NODES;i++)
		fprintf(all_results, "%.2f ", (res+i)->link_th);
	fprintf(all_results, "\n");

	for(i=0; i < NB_NODES;i++)
		fprintf(all_results, "%.2f ", (res+i)->node_th);
	fprintf(all_results, "\n");
```

```
		for(i=0; i < NB_NODES;i++)
			fprintf(all_results, "%.2f ", (res+i)->q_delay);
		fprintf(all_results, "\n");

		for(i=0; i < NB_NODES;i++)
			fprintf(all_results, "%.2f ", (res+i)->bf_load);
		fprintf(all_results, "\n");

		for(i=0; i < NB_NODES;i++)
			fprintf(all_results, "%.2f ", (res+i)->pk_dropped);
		fprintf(all_results, "\n");

		for(i=0; i < NB_NODES;i++)
			fprintf(all_results, "%.2f ", (res+i)->g_delay);
		fprintf(all_results, "\n");

		fclose(all_results);

		return ERR_OK;
}
```

Appendix 1.15 C modules: wasp.h and wasp.c

```
/************************************

		Christophe Jelger
		MPhil Student
		University of Wales Swansea

		WDM Simulator - Version 1.02

		Wasp.h

************************************/

#ifndef WASP_H
#define WASP_H

#include "../wavelength/wavelength.h"

typedef struct			/* structure for wavelength load counter */
{
		int wc_id;
		wavelength* wc_wave;
		int wc_sum;
		float wc_load;
		int wc_own_sum;
		float wc_own_load;
		int wc_state;

} wc_counter;

extern int wc_init(wc_counter*, int, wavelength*);

extern int wc_display(wc_counter*);

extern int wc_add_to_sum(wc_counter*, int);

extern int wc_compute_load(wc_counter*);

extern int wc_update_load(void);

extern int wasp_manual_chg_wave(void);

#endif /* WASP_H */
```

```
/************************************

        Christophe Jelger
        MPhil Student
        University of Wales Swansea

        WDM Simulator - Version 1.02

        Wasp.c

************************************/

#include "../error.h"
#include "../global.h"
#include "../main.h"
#include "../node/node.h"
#include "../predict/predict.h"

#include <stdio.h>


int wc_init(wc_counter* wc, int id, wavelength* wave)
{
        wc->wc_id = id;
        wc->wc_wave = wave;
        wc->wc_sum = 0;
        wc->wc_load = 0;
        wc->wc_own_sum = 0;
        wc->wc_own_load = 0;
        wc->wc_state = 0;

        return ERR_OK;
}

int wc_display(wc_counter* wc)
{
        printf("\nWavelength Counter ...");
        printf("\nID = %d", wc->wc_id);
        printf("\nWavelength : %s", (wc->wc_wave)->wv_name);
        printf("\nTemp sum = %d slots set", wc->wc_sum);
        printf("\nLoad = %.2f %%", wc->wc_load);
        printf("\nTemp own sum = %d slots", wc->wc_own_sum);
        printf("\nOwn load = %.2f %%", wc->wc_own_load);
        printf("\nState = %d\n", wc->wc_state);

        return ERR_OK;
}

int wc_add_to_sum(wc_counter* wc, int wc_flag)
{
        wc->wc_sum++;       /* sum is increased by 1*/

        if( wc_flag == 1 )
                wc->wc_own_sum++;               /* the OWN sum is also increased */

        return ERR_OK;
}

int wc_compute_load(wc_counter* wc)
{
        float previous;

        previous = wc->wc_load;

        wc->wc_load = (float)(wc->wc_sum)/NB_SLOTS_INT*100;
        wc->wc_sum = 0;

        wc->wc_own_load = (float)(wc->wc_own_sum)/NB_SLOTS_INT*100;
        wc->wc_own_sum = 0;

        if(wc->wc_load >= previous)
                wc->wc_state = 1;
        else
                wc->wc_state = -1;
```

```
	return ERR_OK;
}

int wc_update_load(void)
{
	int i,j;
	int wc_flag=0;
	int offset;
	packet* pk_current;

	for(i=0; i < NB_NODES; i++)	/* each node */
	{
		offset = ( node_current[i] - ((node_array+i)->nd_wave)->wv_table );

		for(j=0; j < NB_WV ; j++)		/* for each wavelength */
		{
			pk_current = ( (wave+j)->wv_table + offset );

			if( pk_test_empty(pk_current) == ERR_PK_SET )		/* if pkt set */
			{
				if( pk_current->pk_source_id == (node_array+i)->nd_id )
					wc_flag = 1;
				else
					wc_flag = 0;

				wc_add_to_sum( &load_counter[i][j], wc_flag );
			}
		}
	}

	/* temp test for node 1*/

	offset = ( node_current[0] - ((node_array+0)->nd_wave)->wv_table );
	pk_current = ( (wave+0)->wv_table + offset );

	mx_update_XtY(&node1_matrix, &load_counter[0][0]);

	return ERR_OK;
}

int wasp_manual_chg_wave(void)
{
	int wave_id, node_id;

	printf("\nEnter the Node ID : ");
	scanf("%d", &node_id);

	printf("\nEnter the wavelength ID : ");
	scanf("%d", &wave_id);

	node_current[node_id-1] = nd_chg_wave( (node_array+node_id-1), (wave+wave_id-1),
	node_current[node_id-1] );

	return ERR_OK;
}
```

Appendix 1.16 C modules: wavelength.h and wavelength.c

```
/************************************

	Christophe Jelger
	MPhil Student
	University of Wales Swansea

	WDM Simulator - Version 1.02

	Wavelength.h

************************************/
```

```
#ifndef WAVELENGTH_H
#define WAVELENGTH_H

#include "../fiber/fiber.h"
#include "../packet/packet.h"
#include <stdio.h>

typedef struct
{
	char wv_name[32];
	int wv_id;
	double wv_rate;
	int wv_status;
	int wv_slotsize;
	float wv_slottime;
	float wv_slotnumber;
	fiber* wv_link;
	packet* wv_table;
} wavelength;

extern int wv_display(wavelength*);

extern int wv_init(wavelength*, char*, int, double, int, int, float, float, fiber*);

extern int wv_init_table(wavelength*);

extern int wv_free_table(wavelength*);

extern packet* wv_rotate(wavelength*, packet*);

extern packet* wv_reset_current(wavelength*);

extern int wv_display_to_file(wavelength*, FILE*);

#endif /* WAVELENGTH_H */
```

```
/************************************

	Christophe Jelger
	MPhil Student
	University of Wales Swansea

	WDM Simulator - Version 1.02

	Wavelength.c

************************************/

#include "wavelength.h"
#include "../global.h"
#include "../error.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int wv_init(wavelength* wv, char* name, int id, double rate, int status, int slotsize, float
slottime, float slotnumber, fiber* link)
{
	strcpy( wv->wv_name, name);
	wv->wv_id = id;
	wv->wv_rate = rate;
	wv->wv_status = status;
	wv->wv_slotsize = slotsize;
	wv->wv_slottime = slottime;
	wv->wv_slotnumber = slotnumber;
	wv->wv_link = link;
	return ERR_OK;
}

int wv_display(wavelength* wv)
{
```

```
		printf("\nWavelength parameters :");
		printf("\nNAME = %s", wv->wv_name);
		printf("\nID = %d", wv->wv_id);
		printf("\nRATE = %.3e Gbps", wv->wv_rate);
		printf("\nSTATUS = %d", wv->wv_status);
		printf("\nSLOT SIZE = %ld Bits", wv->wv_slotsize);
		printf("\nSLOT TIME = %.1f us", wv->wv_slottime);
		printf("\nNb of SLOTS = %.0f", wv->wv_slotnumber);
		printf("\nAttached to fiber : %s\n", (wv->wv_link)->fb_name);
		return ERR_OK;
}


/* return the next slot/packet of a given wavelength from a current position */

packet* wv_rotate(wavelength* wv, packet* current)
{
		if( (current - wv->wv_table) < (NB_SLOTS_INT - 1) )
					return (++current);
			else
					return (wv->wv_table);
}


/* return the wavelength packet table BASE pointer*/

packet* wv_reset_current(wavelength* wv)
{
		return wv->wv_table;
}

int wv_init_table(wavelength* wv)
{
		wv->wv_table = calloc( (int)NB_SLOTS_INT, sizeof(packet) );

		return ERR_OK;
}

int wv_free_table(wavelength* wv)
{
		free(wv->wv_table);

		return ERR_OK;
}

int wv_display_to_file(wavelength* wv, FILE* wv_file)
{
		int i;

		for(i=0; i<NB_SLOTS_INT; i++)
			pk_display_to_file( (wv->wv_table + i) , wv_file );

		return ERR_OK;
}
```

Appendix 3:

C code of Poisson traffic generator

```
/************************************

        Christophe Jelger
        MPhil Student
        University of Wales Swansea

        WDM Ring Simulator

        POISSON TRAFFIC GENERATOR

************************************/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define DEST_ID_NB      63      /* number of destination nodes */
#define FRAME_TIME      691.2


int traffic();


FILE *result;                   /* File for traffic results */
FILE *stat;                     /* File for statistics */

long loops = 0;                 /* counter for loops */
double time_limit;              /* time for simulation in us */
double rnd_nb;                  /* the uniform random number */
double limit;                   /* the max random number */
double variate;                 /* the final exponenetial variate (interarrivals) */
double sum_variate = 0;         /* used to compute the experimental mean for variate */
double sum_rnd = 0;             /* used to compute the experimental mean for UD */
double time = 0;                /* our simulated time in us */
double unit_time_counter = 0;   /* a local time counter for file results */
long pkt_per_time_unit = 0;     /* nb of pkts per unit of time */
double mean;                    /* as named ... */
double pkt_time = 12;
int seed;                       /* seed for random generator */

char file_real[32];             /* the REAL result file name */
char file_stat[32];             /* the STAT file name */

int dest_ids[DEST_ID_NB];       /* array of destination nodes */
int src_id;                     /* the source node ID */
int i;
float j,k;

int control = 1;                /* to stop or carry on with new values */

int main(void)
{
        limit = pow(2,31);      /* the max random number we can generate */

        printf("\n\n-------------------------------------------------------------");
        printf("\nPoisson Traffic Generator ... version 1.02");
        printf("\n\nUniversity of Wales Swansea");
        printf("\nDepartment of Electrical and Electronic Engineering");
        printf("\n-------------------------------------------------------------\n");

        printf("\n\nEnter the time origin (in us) : ");
        scanf("%lf", &time);
        printf("Enter seed : ");
        scanf("%d", &seed);
        printf("Enter node source ID : ");
        scanf("%d", &src_id);
        printf("Enter file name for REAL traffic (max 32) : ");
```

```
        scanf("%s", file_real);
        printf("Enter file name for STAT traffic (max 32) : ");
        scanf("%s", file_stat)

        result = fopen(file_real, "w");       /* the file for traffic results */
        stat = fopen(file_stat, "w");         /* the file for statistics */

        srandom(seed);

        printf("\nDestination nodes :");     /* create the destination table */

        for(i=0; i<DEST_ID_NB; i++)
        {       dest_ids[i] = ((src_id+i)%(DEST_ID_NB+1)) + 1;
                printf(" %d", dest_ids[i]);
        }


        while( control != 0 )
        {
                printf("\n 0. Exit");
                printf("\n 1. Traffic");

                scanf("%d", &control);

                switch( control )
                {
                        case 0 : break;

                        case 1 :
                        {       traffic();
                                break;      }
                }

        } /* close main loop */

        fclose(result);
        fclose(stat);

        return 0;
}


int traffic()
{
        double start_pktime, end_pktime, ref_time;

        printf("\nEnter the target time (in us) : ");
        scanf("%lf", &time_limit);

        printf("Enter start pkt time : ");
        scanf("%lf", &start_pktime);

        printf("Enter start pkt time : ");
        scanf("%lf", &end_pktime);

        mean = start_pktime;

        ref_time = time;

        while( time < time_limit )
        {
                if(unit_time_counter >= FRAME_TIME)
                {
                        mean += (end_pktime - start_pktime)/((time_limit - ref_time)/FRAME_TIME);

                        unit_time_counter -= FRAME_TIME;

                        /* prints the nb of pkts / unit time */

                        fprintf(stat, "\n%ld", pkt_per_time_unit);

                        pkt_per_time_unit = 0;
                }

                rnd_nb = (random()/limit);          /* the random number 0<= x <=1 */
```

```
                /* compute the IAT */

                variate = (-mean)*log(rnd_nb);      /* Poisson IAT */
                sum_variate += variate;
                sum_rnd += rnd_nb;

                time += variate;

                /* compute the destination node ID */

                j=1;
                for(i=1; i<=DEST_ID_NB; i++)
                {
                        k = j/DEST_ID_NB;

                        if( rnd_nb <= k )       /* prints the SRC_ID and DEST_ID */
                        {
                                fprintf(result, "%d\t%d\t", src_id, dest_ids[i-1]);
                                rnd_nb = 2;
                        }

                        j++;
                }

                fprintf(result, "%.1lf\n", time); /* prints the time the pkt was issued */

                unit_time_counter += variate;

                time += pkt_time;
                unit_time_counter += pkt_time;
                pkt_per_time_unit += 1;
                loops += 1;

        }

        printf("\n\nConstant rate statistics ... \n");
        printf("\nMean random : %lf", sum_rnd/loops);
        printf("\nMean value : %lf", sum_variate/loops);
        printf("\nTime elapsed : %lf", time-time_limit);

        return 0;
}
```

Appendix 4:

C code of self-similar traffic generator

```
/*****************************************

        Christophe Jelger
        MPhil Student
        University of Wales Swansea

        WDM Ring Simulator

        ON/OFF aggregated TRAFFIC sources

*****************************************/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define DEST_ID_NB          63          /* number of destination nodes */


int traffic();
int dtoi(double);

FILE *result;                           /* File for traffic results */
FILE *stat;                             /* File for statistics */

double time_limit;          /* time for simulation in us */
double rnd_nb;                          /* the uniform random number */
double limit;                           /* the max random number */
double variate;                         /* the final exponenetial variate (interarrivals) */

double unit_time;
double unit_time_counter=0;

double hurst_on, mean_on, alpha_on;     /* PARETO OFF parameters ... */

double hurst_off, mean_off, alpha_off;  /* PARETO OFF parameters ... */

double beta_on, beta_off, smallest, correction;

double off_value, on_value;     /* generated values by EXP and PARETO disctribution */

double load;                            /* the overall LOAD */

int nb_sources;                         /* the number of ON-OFF sources */

char file_real[32];             /* the REAL result file name */
char file_stat[32];             /* the STAT file name */

int dest_ids[DEST_ID_NB];       /* array of destination nodes */
int src_id=1;                           /* the source node ID */


int main(void)
{
        int i;

        limit = pow(2,31);

        smallest = 1/limit;

        printf("\n\n-------------------------------------------------------");
        printf("\nTraffic Generator ... ON-OFF version 0.15");
        printf("\n\nUniversity of Wales Swansea");
        printf("\nDepartment of Electrical and Electronic Engineering");
        printf("\n-------------------------------------------------------\n");
```

```
	printf("\nEnter file name for REAL traffic (max 32) : ");
	scanf("%s", file_real);
	printf("\nEnter file name for STAT traffic (max 32) : ");
	scanf("%s", file_stat);
	printf("\nEnter precision for STAT : ");
	scanf("%lf", &unit_time);
	printf("\nEnter the target time (in us) : ");
	scanf("%lf", &time_limit);
	printf("Enter the NODE SOURCE ID : ");
	scanf("%d", &src_id);
	printf("Enter the HURST PARAMETER (0.5<= H <1): ");
	scanf("%lf", &hurst_on);
	printf("Enter the expected LOAD : ");
	scanf("%lf", &load);
	printf("Enter the number of sources : ");
	scanf("%d", &nb_sources);

	result = fopen(file_real, "w");		/* the file for traffic results */
	stat = fopen(file_stat, "w");		/* the file for statistics */


	/* first we compute the parameters of the pareto distributions */


	alpha_on = 3-(2*hurst_on);
	beta_on = 12;
	mean_on = (alpha_on*beta_on)/(alpha_on-1);

	alpha_off = 1.2;
	hurst_off = (3-alpha_off)/2;

	beta_off = ((nb_sources-load)*alpha_on*beta_on*(alpha_off-1))/(load*(alpha_on-1)*alpha_off);
	mean_off = (alpha_off*beta_off)/(alpha_off-1);

	correction = (1-pow(smallest,(alpha_on-1)/alpha_on))/
		(1-pow(smallest,(alpha_off-1)/alpha_off));

	printf("\nDestination nodes :");	/* create the destination table */

	for(i=0; i<DEST_ID_NB; i++)
	{	dest_ids[i] = ((src_id+i)%(DEST_ID_NB+1)) + 1;
		printf(" %d", dest_ids[i]);
	}

	printf("\n\nPARETO (ON) parameters : HURST=%.2lf ALPHA=%.2lf MEAN=%.2lf BETA=%.2f", hurst_on,
	alpha_on, mean_on, beta_on);

	printf("\n\nPARETO (OFF) parameters : HURST=%.2lf ALPHA=%.2lf MEAN=%.2lf BETA=%.2f
BETA(CORR)=%.2f\n\n",
	hurst_off,alpha_off, mean_off, beta_off, beta_off*correction);


	beta_off = beta_off*correction;			/* we use the corrected value of beta_off */

	/* create the destination table */

	for(i=0; i<DEST_ID_NB; i++)
		dest_ids[i] = ((src_id+i)%(DEST_ID_NB+1)) + 1;

	traffic();	/* the traffic is generated in this function */

	fclose(result);
	fclose(stat);

	return 0;
}


int traffic()
{
	int i,j, tx, dest_id;

	float k,h;

	long pkts_tx = 0;
```

```
double tmp_time=0;

double *data_set;

double smaller_exp=time_limit+100;

double time=0;

int current_pkt, nb_pkts_to_tx;


/* initialise the array of OFF-ON sources (times) */

data_set = calloc( (int)(nb_sources*2), sizeof(double) );

srandom(src_id);

/* generate the first PAIR (ON-OFF) for ALL sources */

for(i=0; i<nb_sources; i++)
{
        rnd_nb = (random()/limit);
        off_value = beta_off/( pow(rnd_nb,(1/alpha_off)) );

        rnd_nb = (random()/limit);
        on_value = 1/( pow(rnd_nb,(1/alpha_on)) );

        *(data_set + i) = off_value;                    /* OFF */

        *(data_set + i + nb_sources) = on_value;        /* ON */
}

while(time < time_limit)
{
        /* finds the smallest OFF value */

        for(i=0; i<nb_sources; i++)
        {
                if( *(data_set+i) < smaller_exp)
                {
                        smaller_exp = *(data_set+i);
                        current_pkt = i;
                }
        }

        /* rounds up the nb of packets being transmited */

        if( *(data_set+current_pkt+nb_sources)-dtoi(*(data_set+current_pkt+nb_sources)) >= 0.5)
                nb_pkts_to_tx = dtoi(*(data_set+current_pkt+nb_sources))+1;
        else
                nb_pkts_to_tx = dtoi(*(data_set+current_pkt+nb_sources));


        /* creates the destination table */

        h=1;
        rnd_nb = (random()/limit);

        for(i=1; i<=DEST_ID_NB; i++)
        {
                k = h/DEST_ID_NB;

                if( rnd_nb <= k )       /* computes the DEST_ID */
                {
                        dest_id = dest_ids[i-1];
                        rnd_nb = 2;
                }

                h++;
        }


        /* if the OFF value is negative there has been superposition of packets */
```

```
                        if(smaller_exp < 0)
                        {
                                smaller_exp = 0;        /* in this case OFF time is set to zero */
                        }

                        tmp_time += smaller_exp;        /* updates TIME */
                        time += smaller_exp;

                        for(tx=0; tx<nb_pkts_to_tx; tx++)               /* for each packet generated */
                        {
                                if( tmp_time  >= unit_time )        /* do we write in STAT file ? */
                                {
                                        fprintf( stat, "%d\n", pkts_tx );

                                        pkts_tx = 0;
                                        tmp_time = tmp_time - unit_time;
                                }

                                /* writes the packet parameters in the TRAFFIC file */

                                fprintf( result, "%d\t%d\t%.1lf\n", src_id, dest_id,time+(tx*beta_on) );

                                tmp_time += beta_on;

                                pkts_tx ++;
                        }

                        time += (beta_on*nb_pkts_to_tx);                /* time is increased by NB pkts_time */

                        for(i=0; i<nb_sources; i++)         /* updates other pkts OFF time */
                        {
                                *(data_set + i) = *(data_set + i)-(smaller_exp+nb_pkts_to_tx*beta_on);

                        }

                        /* generates NEW ON-OFF pair for source which just transmitted */

                        rnd_nb = (random()/limit);
                        *(data_set + current_pkt) = ( beta_off/( pow(rnd_nb,(1/alpha_off))));

                        rnd_nb = (random()/limit);
                        *(data_set + current_pkt + nb_sources) = 1/( pow(rnd_nb,(1/alpha_on)) );

                        /* we set the smallest EXP to its highest value possible */

                        smaller_exp = time_limit+100;
                }

                return 0;
}

int dtoi(double d)      /* converts a double into an integer */
{
                char tmp[8];

                sprintf( tmp, "%.2lf", d);

                return atoi(tmp);
}
```