Wavelength Assignment Algorithms for Wavelength Division Multiplexing Optical Networks

TSHIAMO TSABONE
STUDENT NUMBER :22009256

SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF MATHEMATICAL AND PHYSICAL SCIENCE
FACULTY OF AGRICULTURE, SCIENCE AND TECHNOLOGY
NORTH WEST UNIVERSITY-MAFIKENG CAMPUS

SUPERVISOR: PROFESSOR OBETEN O. EKABUA

MAY 2014

Declaration

I declare that this research project on the **Wavelength Assignment Algorithms for Wavelength Division Multiplexing Optical Networks** is my work, and has never been presented for the award of any degree in any University. All the information used has been dully acknowledged both in text and in the references.

Signature: Mbaibone Date: 26/09/2014

Approval

Signature..... Date.....

Supervisor: Professor Obeten O. Ekabua
Head of Department
Department of Computer Science
North West University
Mafikeng Campus
South Africa

Dedication

I dedicate this research to my family;

Christopher Ntoayapelo Sekawana, my late grandfather,

Gloria Sekawana, Grandmother,

Grace Tsabone, my late grandmother,

Boitumelo Tsabone, Mother,

Fatlho Tsabone, Father,

Tshepiso Tsabone, Refentse Gill and Tumelo Nako, Sisters,

Onolo Tsabone, Niece,

Lesego Mangwegape Brother,

Ivy Sekawana and Isabella Sekawana, Aunts,

Tsholofelo Lechuti and Koketso Tsikwe, Friend.

Thank you for standing by me and supporting me academically, socially and spiritually.

Acknowledgements

With my deepest sense of gratitude, I would like to thank my supervisor, Professor O.O. Ekabua, the Head of Computer Science Department at the North West University, South Africa. Without his wise counsel, invaluable guidance, support and inspiration, it would have been utterly impossible to complete the present work.

Moreover, I would like to thank Mike Mbougni for guiding me throughout the project. To Mrs Nnenna Eric-Nwonye and Mrs Jane Ifeona Ugochi, your daily check-ups and words of encouragement have greatly helped me to fulfil this work. To Tsholofelo Hope Mogale, I am grateful for all the efforts you have put in my work. OPNET simulator is no child's play and you came to my rescue when my boat was on the verge of sinking. To Mrs D. Mothibi and Dr. B. Muatjetjeja, thank you for being my pillar of strength and having faith in me.

To my friends, Koketso Tsikwe, Mpho Lemphote, Tsholofelo Lechuti, Brian Sejake, Puleng Mokolokolo, Bongani Monama, Freddy Sonakile, and Tshepiso Mere, thank you for being my great support. Your encouraging words showed me that there is light at the end of the tunnel, no matter how unplcasant the challenges of life may be. I will always love you.

To my family, thank you for having faith in me, your support is greatly appreciated.

I am indebted to TELKOM Centre of Excellence for making it possible for me to pursue my degree; their contribution has made a great impact in my life.

Finally, I would like to thank the Almighty, for gracing me the opportunity of life. For I am who I am today because of Him who strengthens me from day to day. It was not by might nor by power but by the Spirit of the Lord that I have achieved so much this far.

List of Acronyms and Abbreviations

AD	Adaptive Routing
ADM	Add/Drop Multiplexer
APON	Automated Passive Optical Network
ATM	Automated Teller Machine
BER	Bit Error Ratio
BoD	Bandwidth on Demand
CO	Central Office
DFB	Distributed Feed Back
DRWA	Dynamic Routing and Wavelength Assignment problem
DWAC	Distinct Wavelength Assignment Constraint
DWDM	Dense Wavelength Division Multiplexing
EMI	Electro Magnetic interference
EMP	Electro Magnetic Pulse
EPON	Ethernet Passive Optical Network
FAR	Fixed Alternate Routing
FDM	Frequency Division Multiplexing
FF	First-Fit algorithm
FPR	Fixed Path Routing
FSM	Finite State Machine
IA-FF	Impairment Aware First Fit
IEEE	Institute of Electrical and Electronics Engineers
IFF	Improved First-Fit algorithm
IRAND	Improved Random algorithm
LCR	Least Congested Routing
LORA	Lexicographical Routing Algorithm

MUW	Most Used Wavelength
NWCC	Non Wavelength Continuity Constraint
OADM	Optical Add/Drop Multiplexer
OEO	Optical Electronic Optical
OPNET	Optimized Network Engineering Tools
OVPN	Optical Virtual Private Network
PABR	Physically Aware Backward Reservation algorithm
PDM	Packet Division Multiplexing
PLD	Permanent Lightpath Demand
PON	Passive Optical Network
QoS	Quality of Service
RAND	Random algorithm
RFI	Radio-Frequency interference
RLD	Random Lightpath Demand
RWA	Routing Wavelength Assignment problem
SDM	Space Division Multiplexing
SLD	Scheduled Lightpath Demand
SLE	Static Lightpath Establishment
TDM	Time Division Multiplexing
URWA	Uniform Random Wavelength Assignment
WAN	Wide Area Network
WCC	Wavelength Continuity Constraint
WDM	Wavelength Division Multiplexing
WRS	Wavelength Routing Switch

Abstract

The rapid growth of internet traffic has been the driving force for faster and more reliable data communication networks. The Wavelength Division Multiplexing (WDM) technique is considered to be one of the best possible techniques in optical network to enhance the capacity of optical fiber in next-generation networks. Wavelength assignment problems are major problems of WDM networks. Therefore, a comprehensive solution on the issues encountered in optical WDM networks needs to be evaluated and resolved using wavelength assignment. This study proposes novel wavelength assignment algorithms in WDM optical networks. The proposed wavelength assignment algorithms are improved first-fit (IFF) algorithm, improved random (IRAND) algorithm and a hybrid algorithm. The uniqueness that IFF possesses is the insertion sort function that assembles wavelengths from the smallest index to the highest. IRAND on the other hand is explored through the link utilization of the path in the network. The hybrid algorithm infuses the concepts of IFF and IRAND. The simulation results are based on OPNET modeler 14.5. This novel algorithm, hybrid, is more efficient in terms of the performance metrics, throughput, delay and utilization.

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

1.1. Introduction and Background

The rapid advancement and evolution experienced in optical network technologies has created the need to minimize the network-wide cost [1]. Wavelength Division Multiplexing (WDM) is a new technology used to promote the rapid growth of internet, telecommunication traffic and expansion capacity in optical networks [2]. An optical network centered on WDM using the wavelength routing technique, is deliberated as a very favourable approach for the recognition of future large bandwidth networks [3]. Furthermore, WDM in optical networks can allow a dozen optical wavelengths (channels) to be multiplexed into a single optical fiber, to promote better transmission at optimal high-speed. Such optical networks are believed to promote data transmission rates that are higher than the existing high level in electronic networks. The speed transmission capacity is 10 Gb/s per wavelength [4, 5].

Such enormous capacity is overwhelming as it causes a huge burden on the electronic switches and routers at respective nodes in the network, that must in-turn process all information. However, in this case it is not necessary for all the traffic that passes through the node to be automatically processed, because in many instances traffic passing through the node is not aimed at a particular node, neither is it intended to. Consequently, WDM network is considered to be the backbone transmission network. Due to the rapid improvement in WDM technology, WDM optical communication network is likely to be a major development route of network construction. To achieve a successful network transmission, most of the current infrastructure network needs to be built in voice transmission, which corresponds with the current technology [4]. Figure 1.1 shows the overview architecture of a WDM network, which emphasizes the importance of communication networks [6]. There are seven interlinked optical wavelength routers, where each router consists of an access station. These optical wavelength routers are interlinked via a bidirectional fiber link, which allows lightpaths, λ_1 and λ_2, to traverse from one router to another.

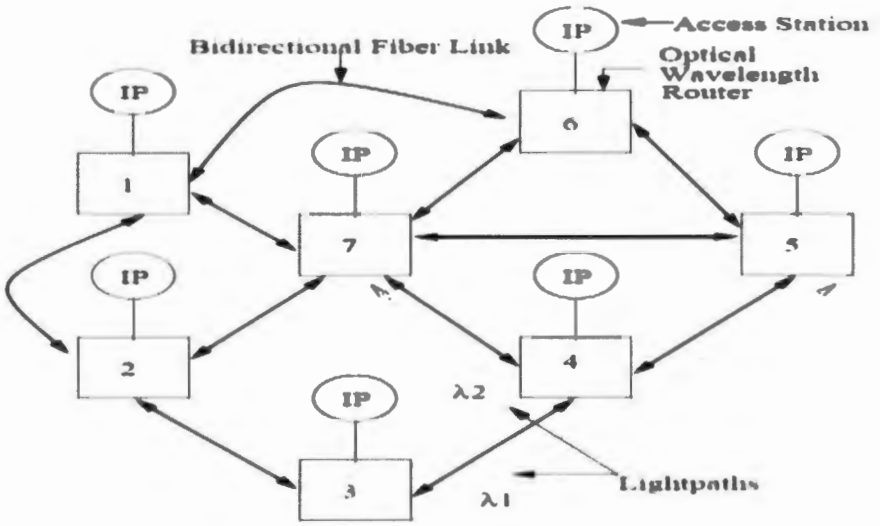


Figure 1.1: Architecture of a WDM Network [6].

The accommodation of several wavelength channels on a fiber is achieved by involving WDM technology, which can be used to enhance the line capacity of the networks [7]. In wavelength routing, data signals are approved on a unique wavelength from a source node to a destination node. WDM optical network comprises of three main constraints which are vital in the promotion of channeling wavelengths. Such constraints fall within the following [8]:

(i) **Wavelength continuity constraint** (WCC): The lightpath is required to occupy the same wavelength that must be used on all fiber links along the designated route, prior to communication between any two nodes.

(ii) **Distinct wavelength assignment constraint** (DWAC): Lightpaths are dispersed to individual wavelengths to avoid any interference in the optical fiber links.

(iii) **Non wavelength continuity constraint** (NWCC): Different wavelengths are utilized on the links beside the designated route, and these wavelengths are required to have the capability of converting wavelengths. Wavelength conversion is the ability to convert the data on one wavelength to another wavelength. Eradicating wavelength conversion considerably decreases the cost of the switch, but it may decrease network efficiency as more wavelengths might be needed for the transmission. Furthermore, the algorithm demands that the nodes neighbouring the protected link have the ability of optical wavelength conversion.

However, the same wavelength can be assigned to two light paths if they are using different fiber links. This property is known as wavelength reuse [9]. More scalable networks can be designed using the wavelength reuse property. By using wavelength converters in OXCs the wavelength continuity constraint can be relaxed. The performance of a wavelength-convertible network is better than wavelength selective networks [9, 10]. Wavelength converters also reduce bandwidth loss which results in better bandwidth utilization. A wavelength converter is a single input single output device that converts the wavelength of an arrived optical signal at its input port to a different wavelength as the signal departs from its output port [9]. Different levels of wavelength conversion capability are possible, as shown in Figure 1.2.

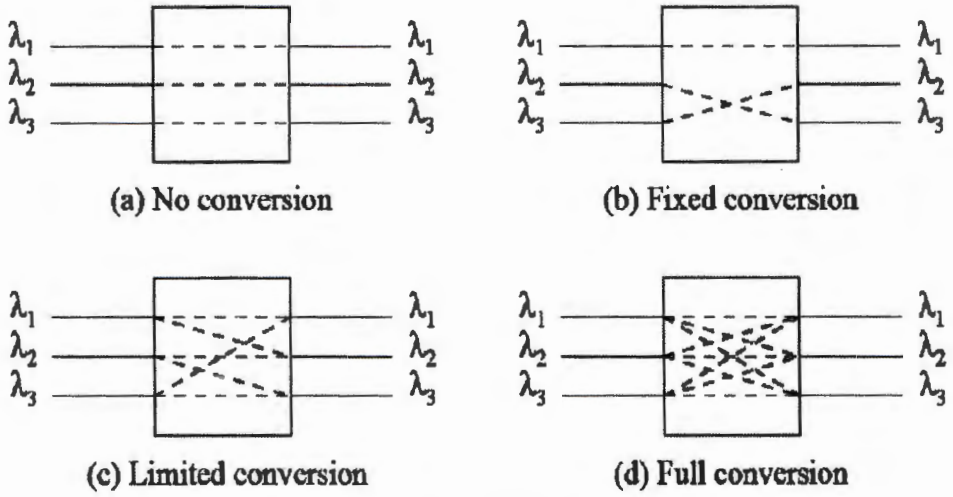


Figure 1.2: Different approaches of wavelength conversion [9].

As is clear from Figure 1.2, full wavelength conversion removes the WCC, making it possible to establish a light path as long as each link along the path from source to destination has a free wavelength (which could be different for different links). However, because of the high cost of converters it is not economically feasible to place converters at all nodes. Therefore, there is a compromise between performance gain and cost. A more cost effective solution is to use only a few converting nodes. This is known as sparse or limited wavelength conversion that can provide the benefits of full wavelength conversion [9].

The essential standard of optical WDM networks is that, if there is more usage in data networks, and a higher pattern usage evolving in bandwidth-intensive networking applications, then a critical necessity for very high-bandwidth transport network facilities

develops, whose proficiencies are much greater than those that present high-speed networks can deliver [11].

Generally, using optical networks and WDM technology involves the following benefits in WDM optical systems [12].

(i) **Transparency**: WDM optical technology is identified by its transparency to the data rate and the format of the data signal. Transmitting numerous systems with diverse protocols and bandwidth requirements that require attentive communication infrastructures is highly advantageous.

(ii) **Scalability and flexibility**: The support of a large density of optical wavelengths in WDM optical networks is favourable; additionally, the use of a common fiber network infrastructure in new WDM technologies is promoted.

Furthermore, WDM-based optical architecture also supports the use of a different set and number of wavelength channels on different links which can be effectively configured and modified for different applications.

(iii) **Reliability**: Reliability in optical WDM networks is a priority and it is accomplished by increasing routes and lightpaths which does not affect the physical structure.

(iv) **Resources reuse**: Dynamic distribution and reuse of light paths in WDM optical network is allowed. This condition permits competent management of resources with no supplementary aerial disadvantage.

These benefits of WDM optical networks enhance the functionality of wavelength transmission in the fiber links.

1.2. Problem Statement

Generally, WDM optical networks are the backbone for the rapid growth of internet and telecommunication traffic [13]. Optical networks using WDM offer an enormous bandwidth capacity for the benefit of the next-generation internet. The above-mentioned network is favourable to fulfil the bandwidth conditions from several emerging multimedia applications. Furthermore, WDM networking technology has been recognized as an appropriate factor for

future Wide Area Network (WAN) environments, with regard to its probable ability to satisfy the expanding requirements of high bandwidth and low latency communication. Data transmission over a network is very volatile, and there are various causes of information loss [14]. Additionally, wavelength networks are prone to cost efficiency, unlike packet networks that consistently recover from various failures. The recovery in the packet layer demands additional resources on the network. This becomes a significant problem in the recovery of wavelength routes due to multiple failures in the network [15]. Subsequently, any inaccuracy that results in wavelength assignment causes low network capacity and high connecting blocking probability. This issue of assigning wavelengths correctly in a network is a vital problem in WDM optical networks [16]. In the arena of definite networks, fiber links endure correlated failures due to sharing of collective physical resources, as an outcome of high blocking probability. Additionally, the utilization of wavelength resources in the failed link is not considered as a priority, hence low wavelength resource utilization ratio [8].

There is a developing requirement to proficiently protect critical multicast sessions against link failures such as fiber cuts. These fiber failures are predominant in communication networks, and at any occurrences of a fiber cut, all connections propagating in the direction of a fiber are somewhat interrupted, and the afflicted destinations have to be extended on temporary routes [11]. With such countless challenges in WDM optical networks, the network technology often does not perform well. Therefore, a comprehensive solution on the issues encountered in optical WDM networks needs to be evaluated and resolved using wavelength assignment [17].

1.3. Research Rationale

Wavelength assignment problems are classified as the main challenging issues of WDM networks. They are identified as an exceptional factor in wavelength routed networks that differentiate them from conventional networks. Unfortunately, the wavelength assignment can result in low network capacity and low blocking probability if the results are inaccurate. The existence of optical connections has become an issue of importance to WDM. Furthermore, the wavelength assignment problem is always aligned with the routing problem, hence the Routing Wavelength Assignment (RWA) problem. The collaboration of these two problems is believed to produce productive results as one will discover a route from the

source to the destination, while the other assigns a wavelength to the identified route. This research will propose and enhance a wavelength assignment algorithm for WDM optical networks.

1.4. Research Questions

This research attempts to provide answers to the following research questions (RQ):

RQ1: Is it possible to develop an efficient wavelength assignment algorithm for WDM optical networks with a practical application?

RQ2: Can the developed wavelength assignment algorithm be implemented in WDM optical networks?

RQ3: Does the developed algorithm perform better when compared with existing algorithms for wavelength assignment?

1.5. Research Goal

The main aim of this research is to design, implement and simulate a novel wavelength assignment algorithm for WDM optical network and compare the obtained results with the existing algorithms.

1.6. Research Objectives

In order to achieve the goal for this research, the following research objectives (RO) are formulated:

RO1: To develop a novel wavelength assignment algorithm for WDM optical network

RO2: To implement the proposed algorithm

RO3: To compare the results obtained with existing algorithms.

1.7. Research Methodology

In order to achieve the main goal of this research through the objectives specified, the following research methodology was employed:

(i) **Literature Survey**

An in-depth study of wavelength assignment problem on optical WDM networks was carried out.

(ii) **Algorithm Development**

A hybrid wavelength assignment algorithm for WDM optical network was designed.

(iii) **Algorithm Implementation**

The implementation process of the designed wavelength assignment algorithm for WDM optical network was sketched.

(iv) **Modeling and Simulation of Algorithm**

This method consisted of the modeling process of WDM networks and simulation of the proposed algorithm using a simulator such as Optimized Network Engineering Tools (OPNET) modeler version 14.5.

(v) **Proof of Concept**

Evaluation of proportional similarities and differences of the improved wavelength assignment algorithm against the existing traditional algorithm from other researchers was done.

1.8. Research Contributions

The main contribution of this dissertation to academia, the research community and network engineers is the development and the implementation of the wavelength assignment algorithms, namely, improved first-fit (IFF), improved random (IRAND), and the hybrid algorithm; and the evaluation of their results with respect to their performance metrics.

1.9. Scope of the Research

The Research work described in this dissertation focuses mainly on modelling Wavelength Assignment Algorithms for WDM Optical Networks. Other challenges in WDM optical networks, such as blocking probability, delay and jitter, are not within the scope of this research. Furthermore, this research is also restricted to simulation implementation; hence no test bed was performed.

1.10. Chapter Summary

Chapter 1: This chapter presents an overview and general background for this thesis. The remainder of this thesis is structured as follows:

Chapter 2: The literature review relevant to this dissertation is presented. A broader description of wavelength assignment in WDM optical network is presented in this chapter. In addition, popular heuristics of wavelength assignment are also presented and elaborated.

Chapter 3: The methodology used for this research work is presented. The algorithm development, algorithm implementation, modeling and simulation of the proposed algorithm are shown.

Chapter 4: The results obtained from the simulation setup used in this dissertation are presented and discussed.

Chapter 5: This is the concluding chapter of this research. A summary of the whole dissertation is firstly presented followed by the concluding remarks and finally suggestions for future work.

CHAPTER 2

LITERATURE REVIEW

2.1. Chapter Overview

This chapter discusses literature material on wavelength assignment in WDM optical networks. It looks at the impact and the importance of wavelength routing techniques in wavelength assignment algorithms. In addition, popular heuristics of wavelength assignment algorithms are discussed and presented.

2.2. Optical Fiber Systems

Network providers are shifting toward utilizing optical networks for provision of increased bandwidth and improvement in fiber performance. In this regard, the evolution of optical systems has developed significantly [18], while the continuous application of WDM technology in systems generates increased complexity [19]. The environment of wireless communication is compositely developing into a communication method, which explores and classifies signals as vital challenges. The transition of an optical fiber communication signal is able to identify non-cooperated communication assignment, such as signal identification, interferer identification, and frequency supervision [20].

The driving force for the development of optical fiber communication systems evolved during the invention of lasers in the early 1960s. This built an environment for examining optical spectrums in relation to radio and microwave spectrums to supply transmission links with enormously high capacities. Numerous composite challenges were identified in the process of accomplishing a steady communication system [21]. Nevertheless, the advances in the technology to date have exceeded even the most optimistic predictions, producing supplementary advantages. These advantages fall within the following [18, 22]:

(i) **Enormous potential bandwidth**: The data capacity carrier of a transmission system is directly proportional to the frequency carrier of the transmitted signals. The range of optical carrier frequency is from 10^{13} to 10^{15} Hz, while radio wave frequency is approximately 10^{6} Hz and microwave frequency is roughly 10^{10} Hz. Therefore, the

optical fiber generates a better transmission bandwidth than the conventional communication systems, including the data rate in the optical fiber communication system. Additionally, the WDM technique of the data rate and information carrying capacity of optical fibers is, therefore, greater by many orders of magnitude.

(ii) **Small size and weight:** Optical fibers have very small widths which never exceed the thickness of a human hair. Therefore, these fibers are very reliable as they are covered with protective coatings and they are far smaller and much lighter than corresponding copper cables. This is a tremendous advantage towards the improvement of channel congestion as it permits expansion of signal transmission within mobiles.

(iii) **Electrical isolation:** Optical fibers are generated from glass, and are therefore electrical insulators, which do not exhibit earth loop and interface problems. Additionally, this attribute means that optical fiber transmission is ideal for communication in electrically hazardous environments.

(iv) **Immunity to interference and crosstalk:** Optical fibers form a dielectric waveguide and are, therefore, unrestricted from electromagnetic interference (EMI), radio-frequency interference (RFI), or switching transients giving electromagnetic pulses (EMPs). The procedure of optical fiber communication is therefore uninterrupted by electrically noisy environments.

(v) **Signal security:** The signal conducted through the fibers does not radiate. Moreover, the signal cannot be employed easily from a fiber. Hence, optical fiber communication significantly provides a high degree of signal security.

(vi) **Low transmission loss:** Transmission loss is guaranteed with ultra-low loss fibers, this enables the employment of communication links with wide optical repeater, hence reduction in both system cost and complexity.

(vii) **Ruggedness and flexibility:** Optical fibers are made from highly flexible material. Taking its weight and size into account, it is notable that optical fiber cables are superior in terms of storage, transportation, handling and installation to corresponding copper cables, while presenting at least equivalent strength and durability.

(viii) **System reliability and ease of maintenance:** These characteristics are primarily important to optical fiber cables, which decreases the requirement for intermediate repeaters or line amplifiers for the improvement of transmitted signal strength.

(ix) **Potential low cost:** The glass which commonly delivers the optical fiber transmission medium is made from sand, which is not a scarce resource. Consequently, in comparison with copper conductors, optical fibers provide better potential for low-cost line communication.

2.3. Optical Network Properties

Optical technologies have three promising attributes for the next-generation access networks. These attributes are as follows [18, 23]:

(i) **Point-to-point topologies:** Point-to-point dedicated fiber links can connect each node to the telecom central office (CO), as showed in Figure 2.1. This architecture is basic but expensive due to the extensive fiber distribution. An alternative method is applying a dynamic star topology, where a switch is positioned near to the nodes so that signals can be multiplexed/de-multiplexed between the subscribers and the CO. This alternative, shown in Figure 2.2, is prone to cost efficiency in terms of the amount of fiber used. A disadvantage of this approach is that the switch is an active component that expects electrical power as well as backup power at the curb-unit location.

(ii) **Passive optical networks:** Passive optical networks (PONs) substitute the switch with a passive optical component such as an optical splitter (see Figure 2.3). This is one of the several possible topologies appropriate for PONs including tree-and-branch, ring, and bus. Using a PON reduces the total amount of fiber deployed, the total number of optical transceivers in the system, and electrical power consumption. Presently, two PON technologies are being considered: Automated Teller Machine (ATM) PON (APON) and Ethernet PON (EPON). APON uses ATM as their layer- 2 protocol; hence, they can provide quality-of-service features. EPON summarizes all data in Ethernet frames and can deliver a comparatively low-priced solution compared to APONs. Furthermore, EPON is becoming very popular and is being standardized as a solution for access networks in the IEEE 802.3ah group.

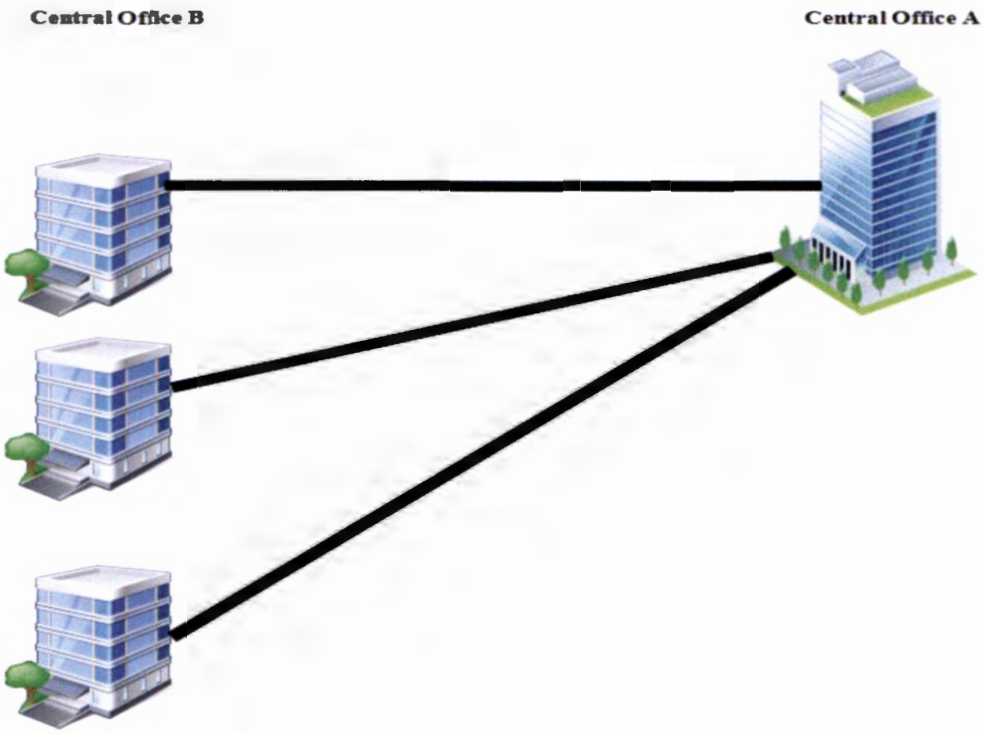


Figure 2.1: Point-to-point topology [18, 23].

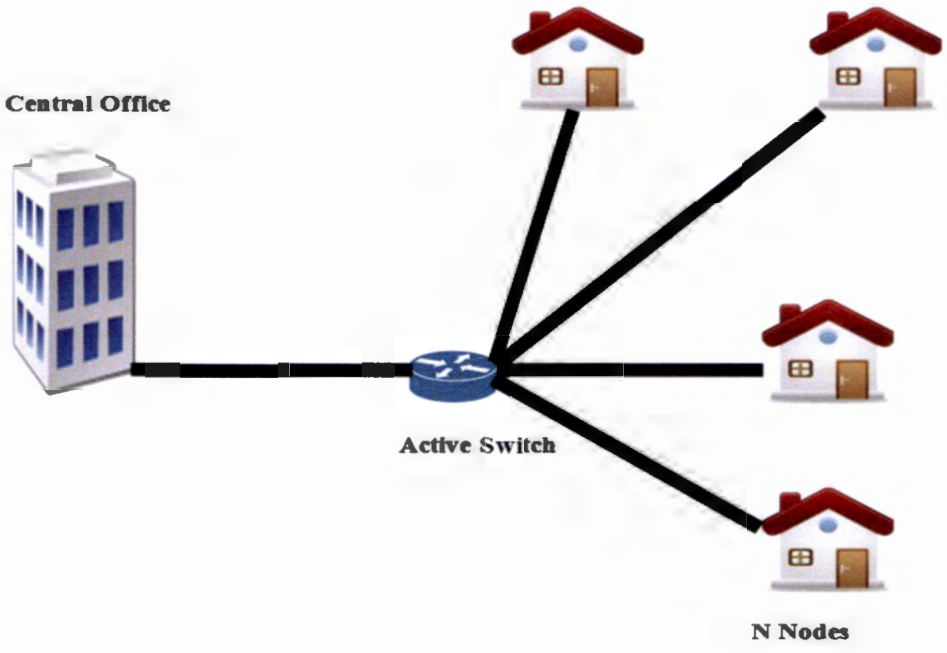


Figure 2.2: Active star topology [18, 23].

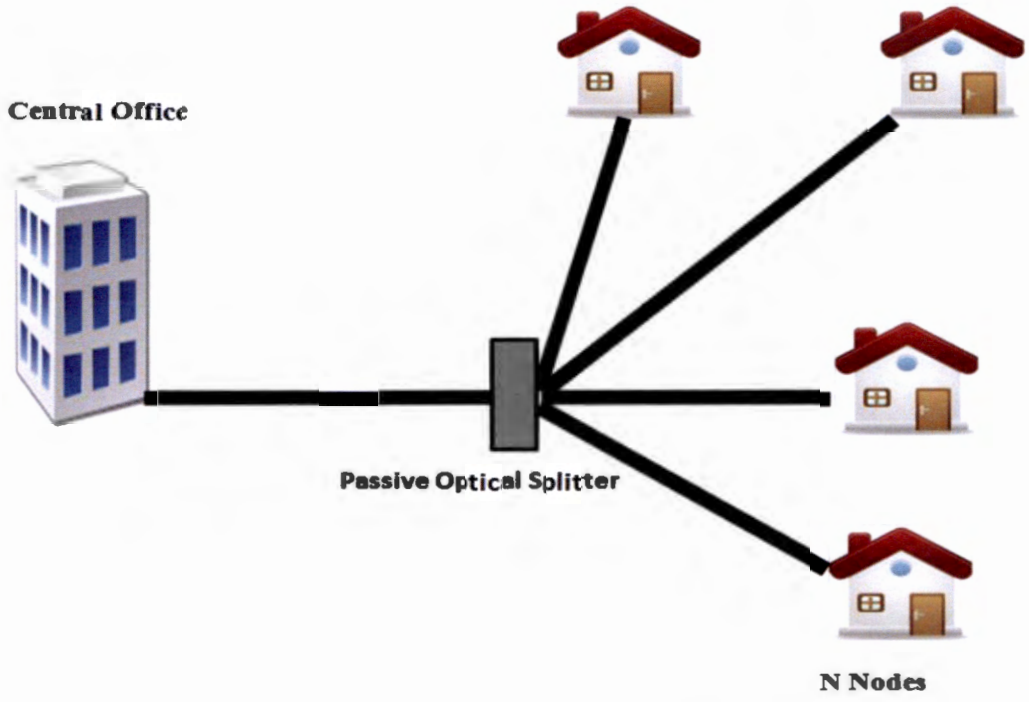


Figure 2.3: Passive Optical Network Topology [18, 23].

(iii) **Optical wireless technology (free space optics):** Low-power infrared lasers can be used to transfer high-speed data via point-to-point (up to 10 Gbps) or meshed (up to 622 Mbps) topologies. An optical data connection can be established through the air via lasers sitting on rooftops targeted at a receiver. Under ideal atmospheric conditions, this technology can deliver a transmission range of up to 4 km. Several challenges need to be addressed for optical wireless technology, including weather conditions, movement of buildings, flying objects, and safety considerations.

The main goal in the development of optical networks is to move toward dynamic all-optical networks, which are also called transparent networks. These include circuit-switched, burst switched and packet-switched networks. In all-optical networks, information is transmitted from sender to recipient entirely in the optical domain without Optical Electronic Optical (OEO) conversions in intermediate nodes. These networks have many advantages. A large number of devices for OEO conversion are not needed and this significantly reduces costs. The decreased number of components in a network decreases the amount of required intervention and probability of network elements failing [18].

2.4. Optical network components

The critical goal of the optical signal transmission is accomplishing the predetermined Bit-Error Ratio (BER) between any two nodes in an optical network. The optical transmission system has to be designed in such a way that it is able to provide consistent operation during its lifetime, which comprises of the management of key engineering parameters [24]. Optical communication systems are comprised of physical principles which support the operations of the components involved in promoting a better network. These major components used in modern optical networks fall within the following [18, 25]:

(i) **Transmitters:** Transmitters consist of many different types of light sources, and the most vital one is a laser. A laser is an essential optical amplifier tool that is encircled within a reflective cavity that causes it to oscillate via positive response. Optical transmitters generally make use of a semiconductor laser diode as a light source. Furthermore, its operational principle is centred on the physical occurrence of stimulated emission. Figure 2.4 shows a structure of a Distributed Feed-Back (DFB) laser diode.

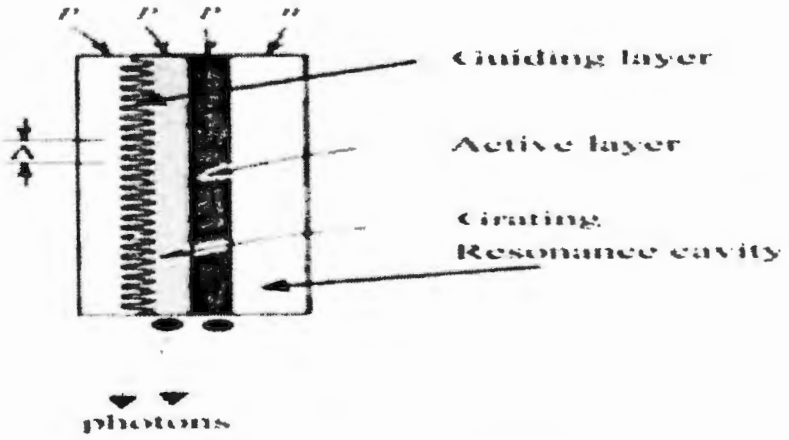


Figure 2.4: Structure of a DFB laser [18].

(ii) **Multiplexers and Demultiplexers:** Multiplexers and demultiplexers are significant components for wavelength-based networks. Both of these components are used to multiplex several channels onto one fiber for transmission and demultiplex signals into distinct channels for routing and detection, respectively. Figure 2.5 shows the structure of a simple optical multiplexer and demultiplexer.

These two components are generally described by the following important key characteristics:

1. Effective optical filters should have low insertion losses.
2. The loss should be independent of the state of polarization of the input signals.
3. The pass-band of a filter should be unresponsive to variations in closing temperature.
4. As more and more filters descend in a WDM system, the pass-band becomes increasingly narrower; the reason for this is to accommodate small changes in operating wavelengths of the laser over time.

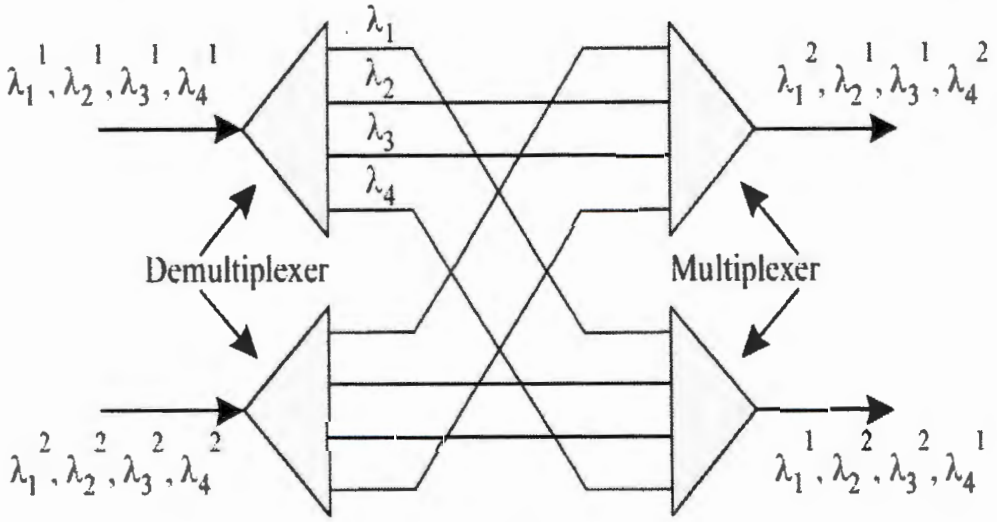


Figure 2.5: A static wavelength crosses-connect [25].

(iii) **Fiber Properties**: Single-mode fibers are extensively employed in today's optical communication networks. Such fibers have reductions as low as 0.2 dB/km in the 1550 nm wavelength range and are made out of silica glass, which is more affordable than other transmission mediums, such as copper coax cable. Figure 2.6 shows the cross-sectional structure of an optical fiber and the geometric optics view of wave propagation in a single-mode fiber. In Figure 2.6, the fiber has cylindrical geometry. It has a core with refractive index n1 and an outer cladding layer with a smaller refractive index n2. Plastic protective layers form part of the cladding, which are not displayed in Figure 2.6. One fundamental principle that monitors the light in the optical fiber is total internal refection. When the incident

angle is smaller than the critical angle, light in the fiber will incur total internal refection, and the entire signal energy will be confined in the core.

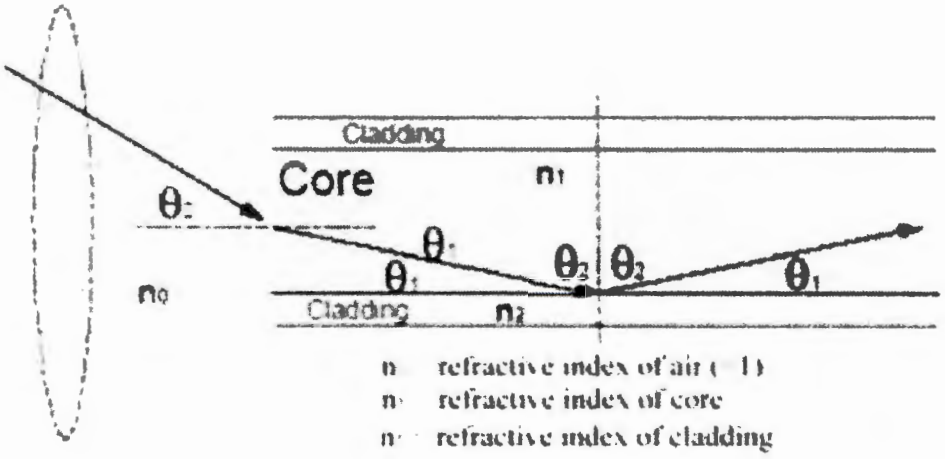


Figure 2.6: Structure of a single-mode fiber and geometric optics theory of wave guides [18].

(iv) **Optical Amplifiers:** In an optical communication system, the optical signals which originate from the transmitter are weakened by the optical fiber as they circulate through it. Adding to this, the accumulated loss of signal strength causes the signal to become too weak to be distinguished. Before this phenomenon takes place, the signal strength has to be restored. Optical amplifiers offer several advantages over regenerators. Moreover, regenerators are specific to the bit rate and modulation format used by the communication system.

(v) **Optical Switches:** Optical switches are significant components for all-optical networks. One of their elementary functions is switching input signals with one wavelength to another output fiber. If the wavelength converters are not channelled properly in the switch, then the input wavelength and the output wavelength become identical. These switches are the origin of the WCC in the RWA problem of optical routing. Through optical switches, wavelengths from distinct links can be linked and a lightpath can be constructed. There are different types of optical switches. Figure 2.7 shows the general structure of an optical switch. In Figure 2.7, all the wavelengths of an input fiber are first de-multiplexed and linked to an array of Wavelength Routing Switches (WRSs). Each WRS is given the responsibility of switching one particular wavelength of each input fiber.

Furthermore, a wavelength on an input fiber can be switched to any output fiber. There are different technologies to implement the WRSs, and thus, different types of optical switches exist [18].

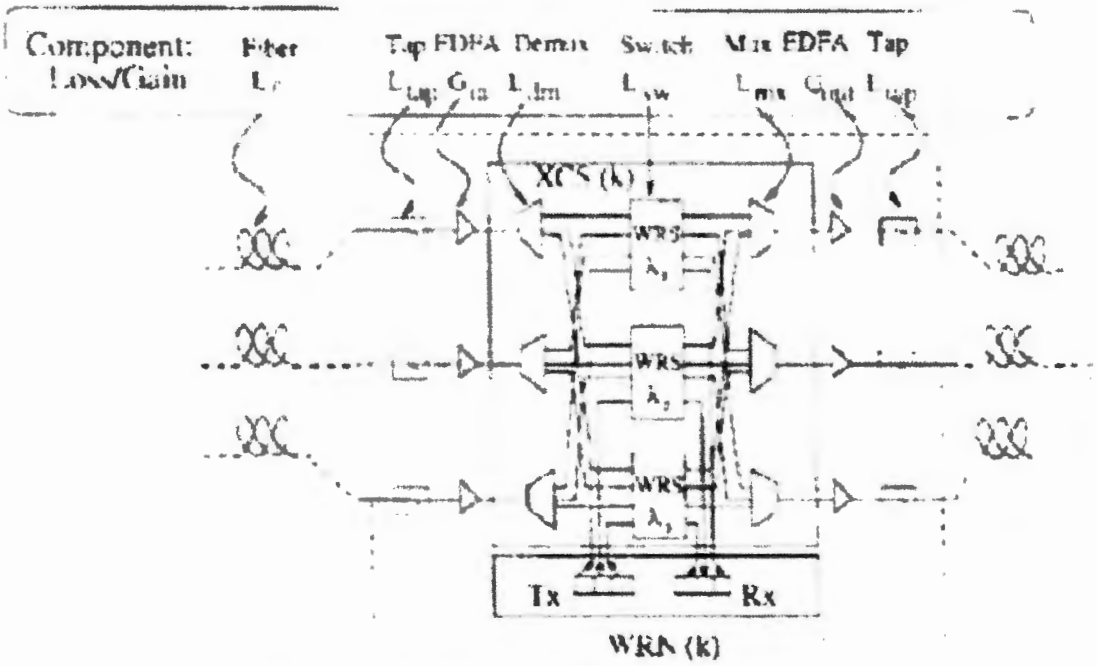


Figure 2.7: The general structure of an optical switch [18].

The procedure of optical signals traversing through the optical fiber is made possible by the passive and active components [18]. These devices, include fibers, light sources, photo detectors and many other components, and are used in a complex optical communication network to split, route, process, or otherwise manipulate light signals [26]. The devices can be categorized as either passive or active components. Passive optical components do not hum or wink or blink, since they require no external source of energy to perform an operation or transformation on an optical signal. Passive components carry out their unique processes without any physical or electrical action. For example, a passive optical filter will allow only a certain wavelength to pass through it while absorbing or reflecting all others, and an optical splitter divides the light entering it into two or more smaller optical power streams. Active components require some type of external energy either to perform their functions or to be used over a wider operating range than a passive device, thereby

offering greater flexibility [26, 27]. Passive components include optical couplers, isolators, circulators, filters, gratings, and wavelength multiplexers [26].

2.5 Multiplexing With WDM

WDM is a vital method in optical networks, which has the ability to manipulate the huge opto-electronic bandwidth incongruity by requiring that every end-user's equipment functions only at electronic rate, but multiple WDM channels from diverse end-users may be multiplexed on the same fiber [28]. The purpose of the wavelength is identified by firstly acknowledging that every WDM channel is unique and is seen as an address that gives direction to network signals [29]. Figure 2.8 shows how different wavelengths are multiplexed.

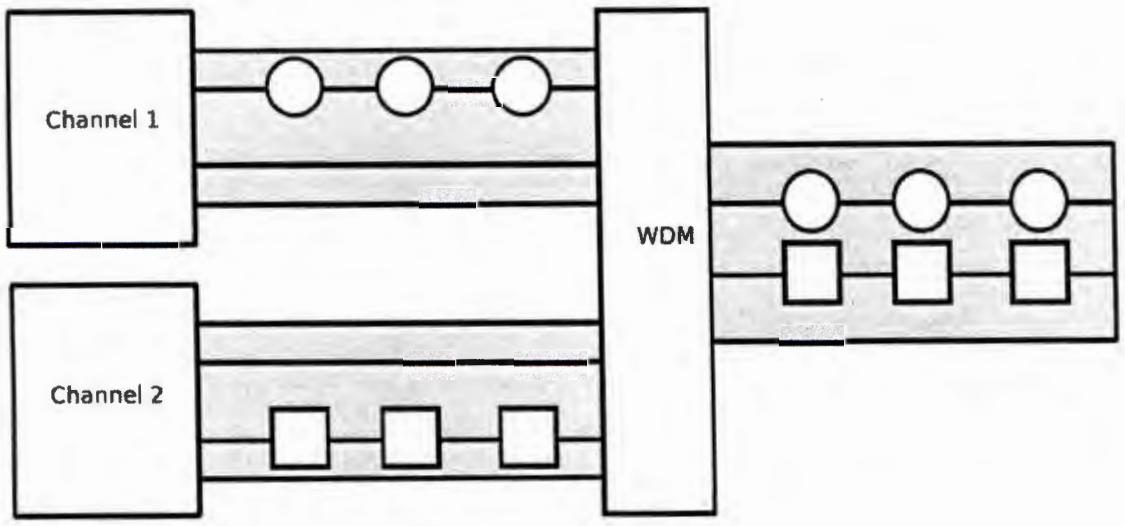


Figure 2.8: Simple schematic of WDM system [29].

In WDM the optical transmission spectrum, depicted in Figure 2.9, shows a carved up motion into a number of non-overlapping wavelength bands, with each wavelength supporting a single communication channel functional at whatever rate one desires [28].

Therefore, by permitting multiple WDM channels to exist on a single fiber, one can tap into the huge fiber bandwidth, with the consistent challenges being the design and development of appropriate network architectures, protocols, and algorithms. Furthermore, WDM devices are easier to implement since, commonly, all components in a WDM device are required to

operate only at electronic speed [30]; as a result, several WDM devices are accessible in the marketplace today, and more are emerging. WDM delivers the way to partition the optical bandwidth into a large number of channels functioning at different carrier frequencies on a single optical fiber. Multiple users, spread over a geographical area, can utilise the optical fiber concurrently using different wavelengths, which in turn leads to a dramatic increase in the total capacity of the network [28]. It is expected that the next generation of the internet will employ WDM-based optical backbones. A primary approach for WDM networks is depicted in Figure 2.10.

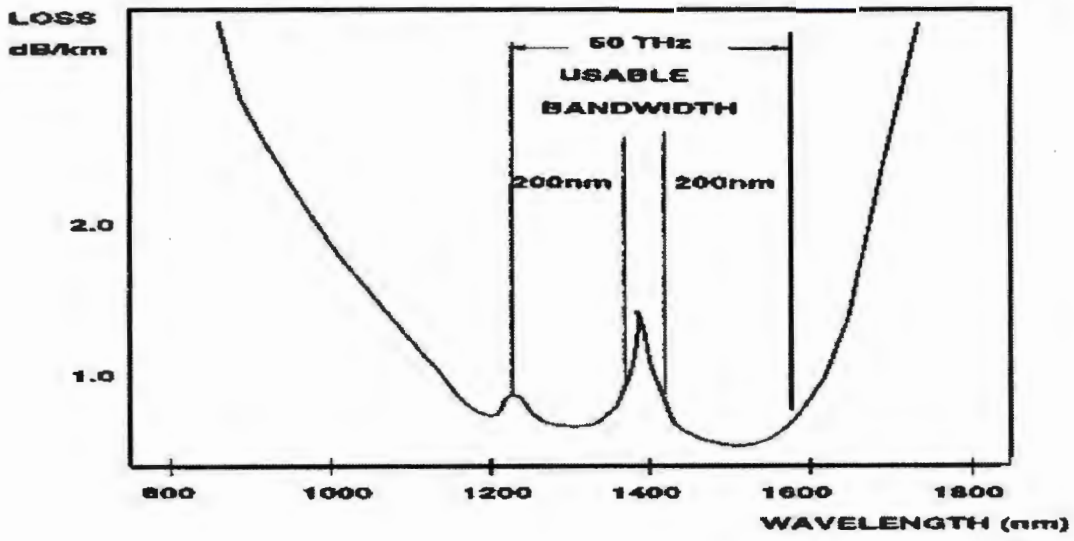


Figure 2.9: The low- attenuation regions of an optical fiber [28].

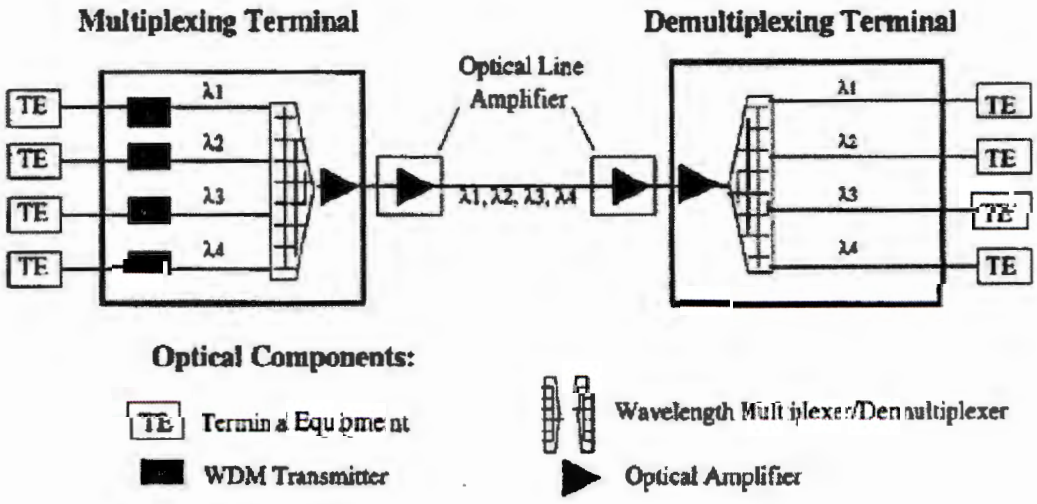


Figure 2.10: WDM Approach [28].

There are various multiplexing techniques used for optical networks. These techniques are as follows [28]:

(i) **Space-division multiplexing (SDM)** – This involves partitioning the physical space to increase transport bandwidth, e.g. bundling a set of fibers into a single cable, or using several cables within a network link.

(ii) **Frequency-division multiplexing (FDM)** – Here the available frequency spectrum is partitioned into a set of independent channels. The use of FDM within an optical network is termed (dense) wavelength-division multiplexing (DWDM or WDM), which enables a given fiber to carry traffic on many distinct wavelengths. WDM divides the optical spectrum into coarser units, called wavebands, which are further divided into wavelength channels.

(iii) **Time-division multiplexing (TDM)** – This divides the bandwidth's time domain into repeated time-slots of fixed length. Using TDM, multiple signals can share a given wavelength if they are non-overlapping in time.

(iv) **Dynamic statistical multiplexing or packet-division multiplexing (PDM)** – This provides "virtual circuit" service in an IP/MPLS over WDM network architecture. The bandwidth of a WDM channel is shared between multiple IP traffic streams (virtual circuits).

These approaches are designed in a way that considers cost minimization and promotes efficient utilization of network resources.

2.6. Benefits of WDM

Wavelength division multiplexing has several advantages over the other presented approaches in increasing the capacity of a link. The advantages of WDM include the following [29]:

(i) It works with existing single mode communication fiber.
(ii) It works with low speed equipment.
(iii) It is transparent: it does not depend on the protocol that has to be transmitted.

(iv) It is scalable: instead of switching to a new technology, a new channel can easily be added to existing channels. Companies only have to pay for the bandwidth they actually need.

(v)

(vi) It is easy for network providers to add additional capacity in a few days if customers need it. This gives companies using WDM an economical advantage. Parts of a fiber can be leased to a customer, who then gets fast network access without having to share the connection with others. The telecommunication company on the other hand still has an independent part of the fiber available for other customers.

2.7. Optical WDM Networks

Due to its enormous capacity and flexibility, optical fiber technology plays a vital role in telecommunication networks. An optical fiber encompasses multiple channels, each using distinctive wavelengths of light which can have high capacities ranging between 10 Gbps and 100 Gbps. In conventional optical networks, WDM multiplexes numerous optical signals in a single optical fiber using distinctive wavelengths. Furthermore, WDM depends on the fact that optical fibers can transmit numerous wavelengths of light simultaneously without allowing any communication between each wavelength. Thus, a single fiber can transmit numerous separate wavelength signals or channels simultaneously [12]. Additionally, an overview of WDM into the existing telecommunications infrastructure signifies the first serious deployment of optical networking in the evolution of the modern day network.

The need for increasing throughput is crucial and pressing, driven by the necessity to deliver a range of high-speed data and video services and the explosion in Internet use [31]. Furthermore, WDM networks confirm the reality of an all-optical information highway that is capable of delivering a comprehensive scope of applications, including the transport of massive amounts of data that demands high speed response times [18]. Optical networking has a variety of characteristics that creates a primary key factor to meet the needs in the network. Allowing a translucent optical physical layer creates a complex platform for system upgrade whilst supporting the existing electronic infrastructure. WDM optical networking has evolved efficiently by accommodating the use of single mode fibers through carrying multiple wavelength signals [31]. Furthermore, photonics were adopted to implement other

functions such as switching and add/drop directly in the optical domain. At present, it is virtually certain that the network will advance as in Figure 2.11 [31]:

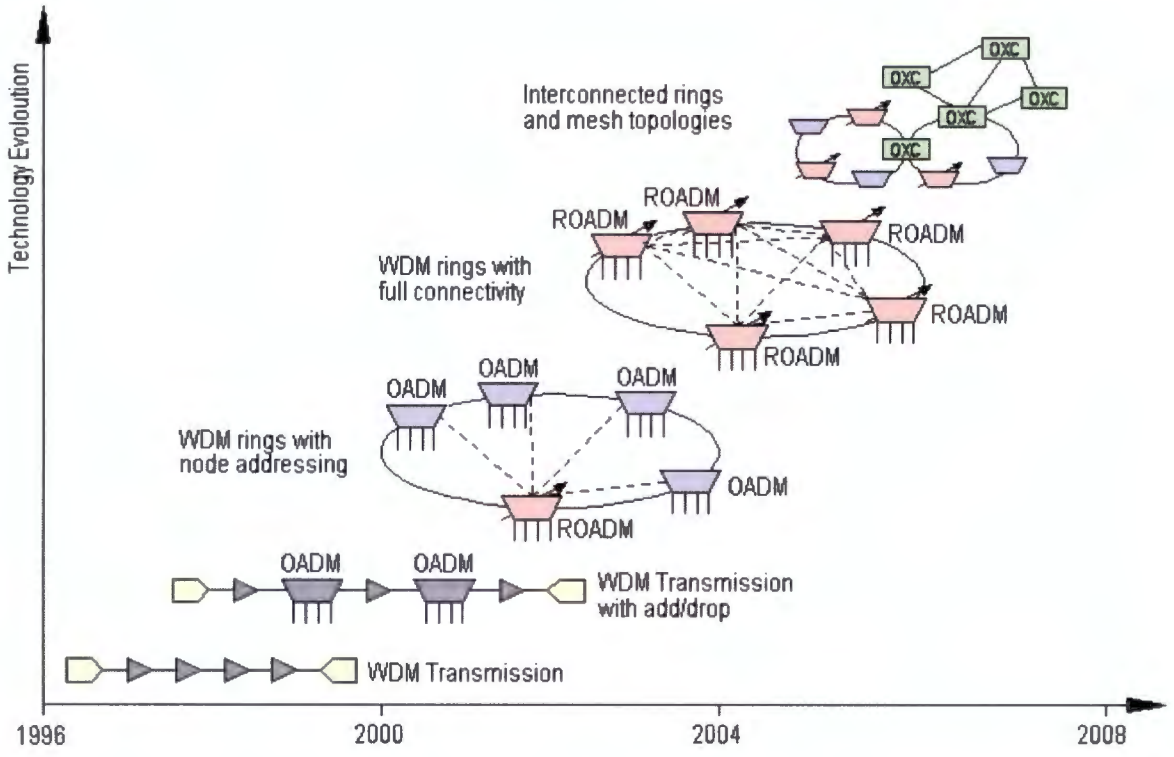


Figure 2.11: Evolution of WDM Optical Networking [31].

2.7.1 Evolution of WDM Optical Networking

2.7.1.1 WDM Point-to-Point Links

Deploying a point-to-point network is the swiftest and most cost-effective technique of transmitting data between two points in a network [32]. Deployment in point-point communication is encouraged by the accumulative demands on communication bandwidth depicted in the Figure 2.12. The wavelengths, X_A and X_B, passing through the optical links, amplify the fiber link dimensions between M and N by a factor 2 [33].

2.7.1.2 Add/Drop Multiplexer

An Add/Drop Multiplexer (ADM) is a multiplexing device that creates network interfaces between different signals. A general ADM node can be expressed with four-port models,

including these basic requirements: a wavelength is required under the road channel; multiplexed signal must transmit into the road. These requirements monitor the flow of traffic, so as to not disturb the wavelength transmission from one road channel to another [4]. Figure 2.13 shows a simple add-drop system for a network.

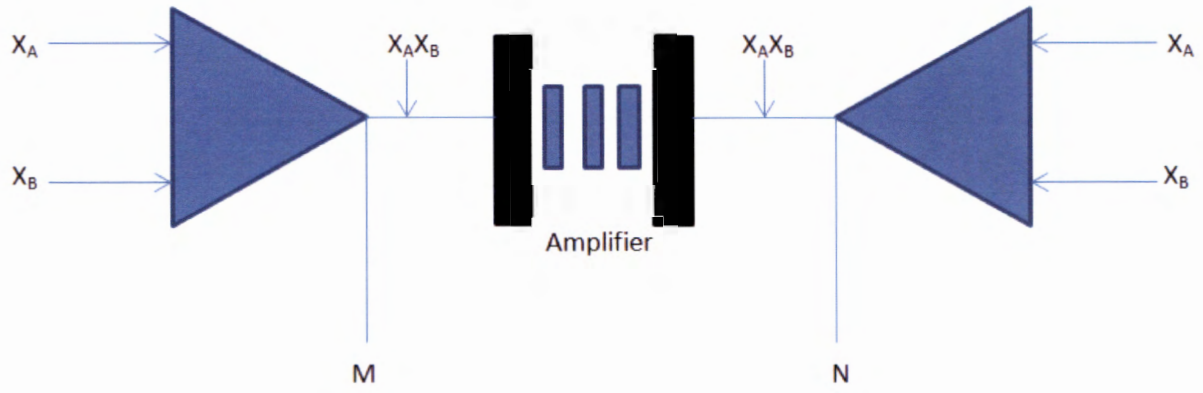


Figure 2.12: WDM point-to-point link [29, 30].

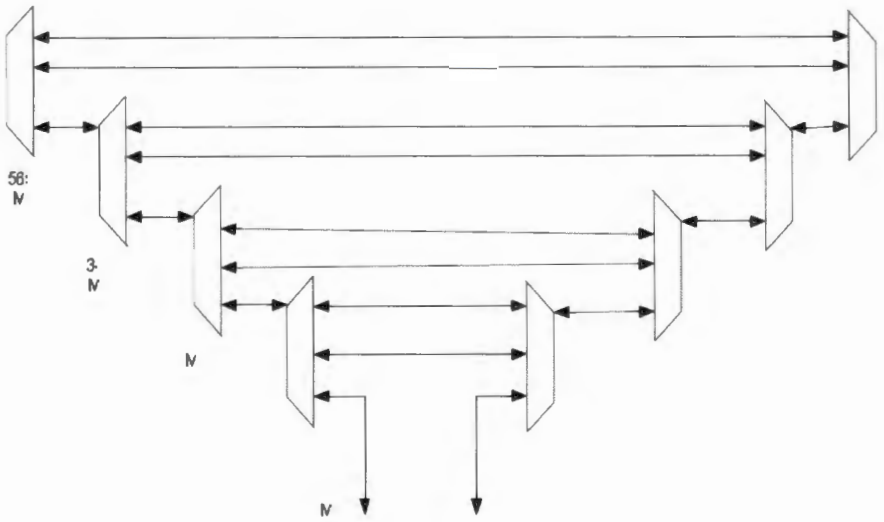


Figure 2.13: Add/Drop System [4].

2.7.1.2.1 Optical Add-Drop Multiplexing

Time domains add/drop multiplexing is graphically presented in Figure 2.14. One or more channels can be dropped and one or more channels can be inserted in the empty time slot(s). A synchronized control signal simultaneously creates a drop and through function. The performances of various ADMs are compared based on several important characteristics,

namely robustness, complexity, polarization dependence, efficiency, number of tributaries and speed limitations [34].

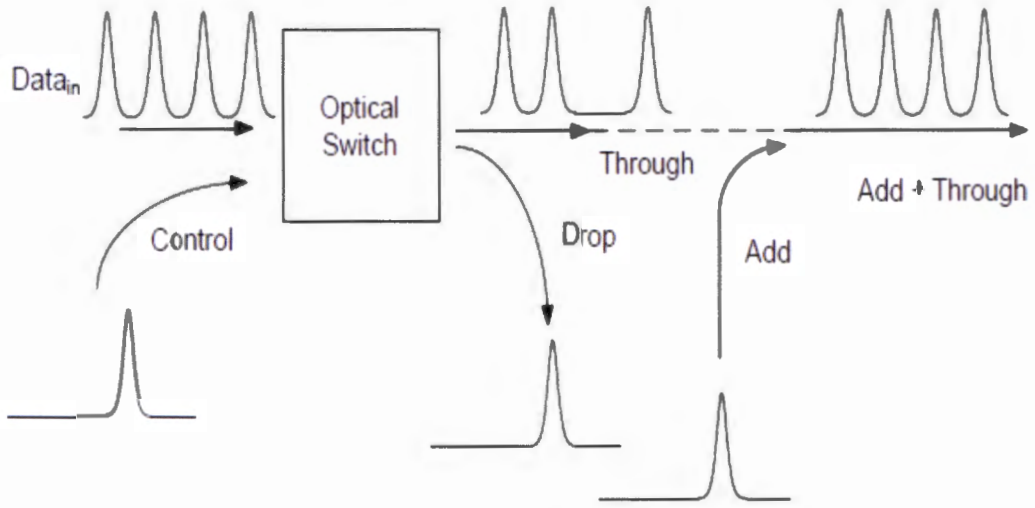


Figure 2.14: Schematic Function of a Time Domain ADM with On Gate Control [34].

An optical add/drop multiplexer performs the function of removing or inserting wavelengths in the network. Rather than combining or separating all wavelengths, the OADM can remove some while passing others on. OADMs have a key role in moving toward the goal of all-optical networks as no conversion of the signal from optical to electrical takes place [35]. A traditional OADM consists of three parts: an optical demultiplexer, an optical multiplexer and between them a method of reconfiguring the paths between the optical demultiplexer, the optical multiplexer and a set of ports for adding and dropping signals [34]. This is illustrated in Figure 2.15.

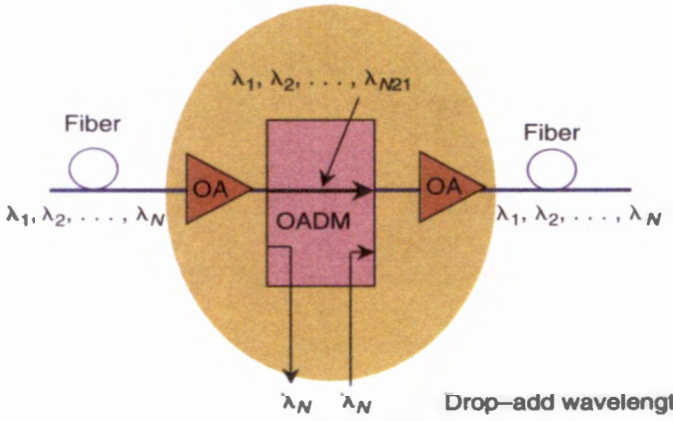


Figure 2.15: Selectively removing and adding Wavelengths [34].

2.8. Routing and Wavelength Assignment

RWA is an exclusive factor of WDM networks in which the light path is performed by choosing a physical link route between source and destination boundary nodes and preserving a specific wavelength on each of these links for the light path [36]. Therefore, for the formation of an optical connection, two requirements need to be met [37]. Firstly, it is required for one to identify an appropriate path from the source node to the destination node of the route, which is defined as the Routing problem. The second requirement, which can also be referred to as a wavelength assignment problem, revolves around the assigning of wavelengths according to the available connections. This resulting problem is known as the routing and wavelength assignment problem, charactised by the following:

(i) Discover a route from the source to the destination.

(ii) Assign a wavelength to the identified route.

A light path connection between any two nodes requires to be confirmed prior to any communication. For the establishment of the light path connection to be validated, a common wavelength needs to be assigned on all the links along the route. This requirement is referred to as the wavelength continuity constraint [36]. Figure 2.16 shows the formation of lightpaths between source-destination (*s-d*) pairs on different wavelengths in a wavelength-routed optical network. The established lightpaths between *s-d* pairs are shown in Figure 2.17 [38]. Each lightpath uses the same wavelength on all hops in the end-to-end path due to its wavelength continuity constraint. The connection requests (A-C) and (B-F) use different wavelength λ_1 and λ_2 because they use the common fiber link 6-7; this property is known as Distinct Channel Constant. The connection requests (H-G) and (D-E) use the same wavelength λ_1 that is already used by the connection request (A-C) due to a wavelength reuse characteristic. Given a set of connection requests, the establishment of lightpaths by routing and assigning a wavelength to each connection is referred to as the RWA problem [39]. The established lightpaths between *s-d* pairs are shown in Table 2.1.

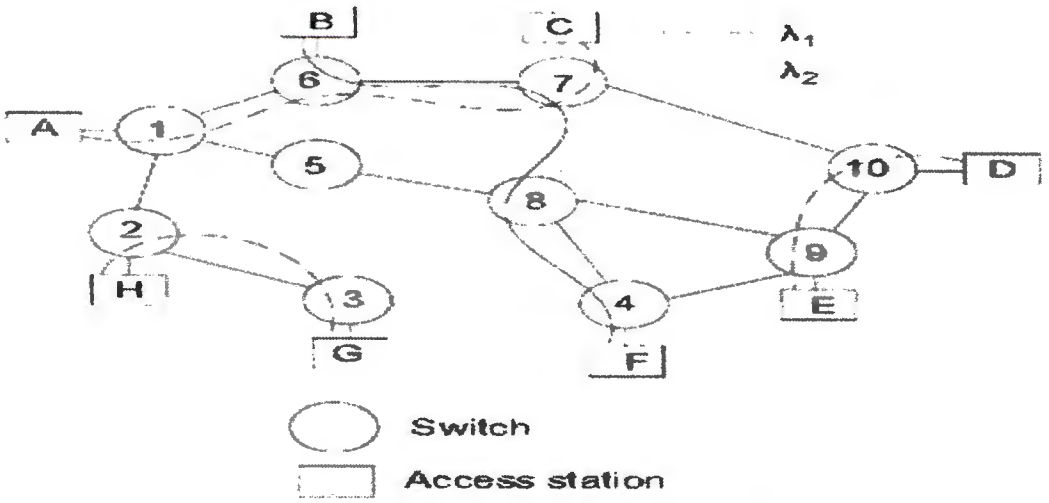


Figure 2.16: A Wavelength routed optical network [39].

Figure 2.17 gives an overview of how an RWA algorithm operates. The nature of this blocking probability can be altered by various factors such as network topology, traffic load, and number of links, algorithms employed and the accessibility of wavelength [37]. Blocking is the fundamental performance index in the policy of an all-optical network, i.e. the network connection only qualifies to be blocked when the network does not have adequate resources to maintain a connection. In this respect, resources are referred to available wavelengths in the network. This constraint is dependent on the type of algorithm used to allocate the wavelength for communicating nodes in the network. Thus algorithms resulting in minimum blocking have the best performance [36].

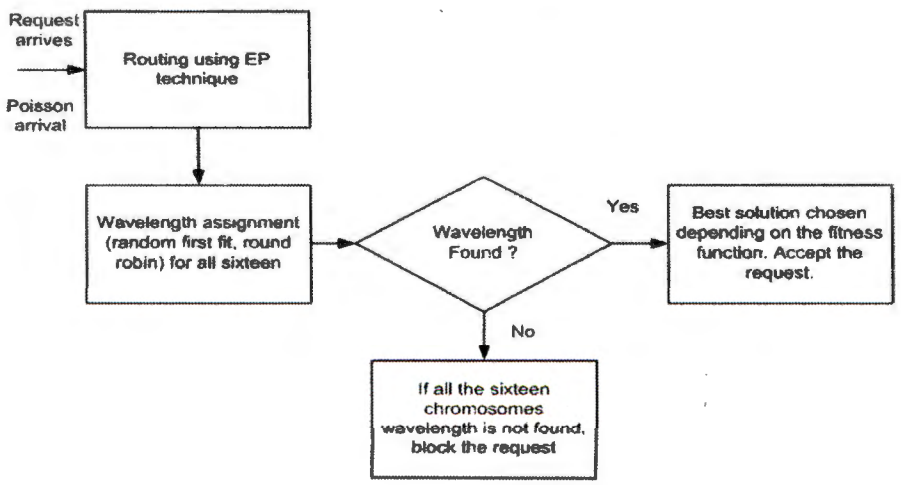


Figure 2.17: Flow chart of RWA algorithm [40].

Table 2.1: Summary of established lightpaths [38].

S-D pair	Used Wavelengths	Lightpath
A-C	λ_1	A-1-6-7-C
B-F	λ_2	B-6-7-8-4-F
H-G	λ_1	H-2-3-G
D-E	λ_1	D-10-9-E

Traffic demands in WDM optical networks can be classified into three categories namely, Permanent (or static) Lightpath Demands (PLDs), Scheduled Lightpath Demands (SLDs) and Random Lightpath Demands (RLDs). PLDs are fully known in advance and have unlimited durations. SLDs are also known in advance, but they are supposed to be active only for a limited period (for example, a few hours, days or weeks). The duration of each SLD is specified by its starting time and ending time. The SLDs for which setup and tear-down times are known in advance can take advantage of the time scheduling property. That is, unless two lightpaths overlap in time, they can be assigned the same wavelength since the paths are disjoint in time [41].

The lightpath scheduling problem in which the whole set of demands is known in advance is known as the deterministic lightpath scheduling problem [41]. Using scheduled lightpaths for traffic adaptation has different timing requirements from the other lightpath scheduling problems. Existing methods for the scheduled routing and wavelength assignment (RWA) problems assume that a lightpath should be set up either at a given time or within a given time window, which makes the lightpath scheduling inflexible for traffic adaptation [42].

For users who desire deterministic services, network resources have to be reserved in advance and guaranteed for future use. In realistic optical networks, it is likely that most of the demands would be initially of the PLD and SLD type. The reason is that the traffic load in core optical networks, such as WDM networks, is quite predictable because of its periodic nature. Such traffic patterns could be predicted from historical statistics, which repeat every day (or week) with minor variations in timing and volume. Hence, the problem of creating the set of SLDs from periodic traffic, i.e. scheduling the lightpaths, is considered. There are various periodic applications, which may be serviced more efficiently by scheduled lightpath demands. For example, SLDs become highly attractive for service providers, who offer

Optical Virtual Private Network (OVPN) or Bandwidth on Demand (BoD) services. They have to establish the set of PLDs to provide minimal network connectivity and capacity requirements, but some SLDs have to be additionally established to increase the required capacities during certain periods of a day or a week [41].

2.8.1 Lightpath Establishment

In the RWA problem, there are three main types of traffic, namely static traffic, incremental traffic and dynamic traffic [38, 43].

(i) In the static traffic, it is assumed that the entire traffic/connection requests are known in advance and the lightpaths are established to satisfy the maximum number of traffic requests. Furthermore, the traffic demand may be specified in terms of source-destination pairs. These types of problems are categorized under the static lightpath establishment (SLE) problem. As the optimal-time algorithms are ideal, polynomial-time algorithms which produce solutions close to the optimal one are preferred to solve the SLE problem.

(ii) In the incremental traffic, traffic/connection requests arrive in the system sequentially, the lightpath is established for each traffic/connection request, and the lightpath remains in the network indefinitely.

(iii) In the dynamic traffic, traffic/connection requests arrive in the system randomly based on a statistical distribution, mainly Poisson process, and a lightpath is established for each traffic/connection request which is released after some finite amount of time.The dynamic traffic demand models several situations in transport networks. Unlike the static RWA problem, any solution to the dynamic problem is computationally simple. Dynamic RWA algorithms perform more poorly than static RWA algorithms because a dynamic algorithm has no knowledge about future connection requests, whereas all the connection requests are known a priori in a static RWA algorithm.

The blocking probability (BP) using static traffic is more than that using incremental or dynamic traffic. Therefore, dynamic traffic is used in the network to minimize the BP and

maximize the network throughput. In the following subsections, we briefly discuss lightpath establishment using static and dynamic traffic.

2.8.1.1 Static Lightpath Establishment

The establishment of lightpath using static traffic is known as the Static Lightpath Establishment (SLE) problem [44]. Many studies have been undertaken to set up lightpaths in the optical network using static traffic.

2.8.1.2 Dynamic Lightpath Establishment

In dynamic provisioning, a lightpath can be established in real-time without predetermined routes and the knowledge of future lightpath provisioning events. The lightpath establishment in this case is dynamic, and the virtual topology is formed by a dynamic lightpath establishment (DLE) technique. In DLE, normally the connection is no longer required after a certain time and the lightpath is to be removed. Using this criterion, on-demand lightpath establishment is implemented in order to enable service providers to respond quickly and economically to customer demands. The DLE problem is difficult to solve and hence heuristic approaches are used [38].

2.9. Wavelength Routing Path Selection Techniques

RWA is a demanding problem [18]. There are two methods of solving the problem. One of them is by decoupling the RWA problem into two, i.e. routing problem and wavelength assignment problem, and the other method is by acknowledging the routing and wavelength assignment problem as a single problem [45]. Many routing and wavelength assignment algorithms designed to efficiently use network resources and provide satisfactory service to network users have been proposed for all-optical networks. These routing algorithms can be classified in two categories: static and adaptive routing algorithms [18].

2.9.1 Static Routing Algorithms

The static method involves knowing the entire set of connections in advance prior to any transmission in the network. A problem that may surface is setting up lightpaths for these connections in a global fashion while minimizing network resources, such as the number of wavelengths or the number of fibers in the network. Furthermore, in static routing algorithms, paths selected for transmission are precalculated for every source-destination pair. One advantage of static routing is reduction of connection provisioning time, but it cannot respond to dynamic traffic conditions in a network. The static modes are described as follows [18, 39, 46]:

(i) **Fixed path routing**: The most complex approach of finding a lightpath is known as fixed path routing (FPR). In this method, a common fixed route for a given source and destination pair is always used. In general, this pathway is calculated ahead of time using a shortest path algorithm, namely, Dijkstra's algorithm. While bearing in mind that this is a complex approach, its performance is usually not satisfactory. If resources along the fixed path are in use, future connection requests will be blocked even though other paths may exist. The SP-1 (Shortest Path, 1 Probe) algorithm is an example of a fixed path routing solution. This algorithm computes the shortest path using the number of optical routers as the cost function. A single probe is used to establish the connection using the shortest path.

The running time is the cost of Dijkstra's algorithm:

$$O(m + n\log n) \tag{2.1}$$

where,

- m is the number of edges, and
- n is the number of routers.

The running time is a constant if a predetermined path is used. This definition of SP-1 uses the hop count as the cost function. The SP-1 algorithm could be extended to use different cost functions, such as the number of EDFAs.

(ii) **Fixed alternate routing**: Fixed alternate routing (FAR) is an updated version of fixed path routing. Unlike in Fixed path routing where a common fixed path is used for a given source and destination pair, FAR considers multiple routes. The probes can be sent in series or in parallel. For each connection request, the source node attempts to find a connection on each of the paths. If all of the paths fail, then the connection is blocked. If multiple paths are available, only one of them is utilized.

The SP-p (Shortest Path, p Probes, $p > 1$) algorithm is an example of fixed alternate routing. This algorithm calculates the p shortest paths using the number of optical routers as the cost function. The running time using Yen's algorithm is:

$$O(pn(m + n\log n)) \quad (2.2)$$

where

- m is the number of edges,
- n is the number of routers, and
- p is the number of paths.

The running time is constant if the paths are precompiled. In some cases having as few as two alternate routes leads to better performance than fixed routing with full wavelength switching, thus improving blocking performance of the networks.

2.9.2 Adaptive Routing Algorithm

Adaptive routing algorithms frequently use the Dijkstra's algorithm to compute the path with the lowest cost from the source to the destination. The definition of the link cost function is important for such algorithms [18, 39, 46]. The following describes some of the adaptive modes in detail:

(i) **Adaptive routing**: The performance of the adaptive routing (AR) algorithm is better than other wavelength routing algorithms in terms of blocking probability. The major issue with both fixed path routing and fixed alternate routing is that neither algorithm takes into account the current state of the network. If the

predetermined paths are not available, the connection request is blocked even though other paths may exist. Two principle conditions that affect routing decisions are [18, 38, 39, 49]:

1. Failure: When a node or trunk fails it can no longer be used as a part of the route and,
2. Congestion: When a particular portion of the network becomes heavily congested it is desirable to route packets around the area of congestion.

Fixed path routing and fixed alternate routing are not aware of the current state of the system. For these reasons, most of the research in RWA currently focuses on adaptive algorithms [18]. Five examples of adaptive routing are LORA, PABR, IA-BF, IA-FF, and Quality of Service (QoS). Adaptive algorithms fall into two categories: traditional and physically-aware [46]. Traditional adaptive algorithms do not consider signal quality; however, physically-aware adaptive algorithms do.

(ii) **Least Congested Routing:** In Least Congested Routing (LCR), a sequence of routes is prearranged for each source-destination pair. Depending upon the arrival of a connection request the least-congested route is selected among the predetermined routes [38]. The congestion on a link is measured by the number of wavelengths available on the link. If the link has fewer available wavelengths, it is considered to be more congested. The disadvantage of LCR is higher computation complexity and its blocking probability is almost same as FAR [38, 39].

The functionality of the above routing algorithms is illustrated with a sample example network shown in Figure 2.18. It consists of 14 nodes, and 21 bi-directional optical links. In Figure 2.18, the fixed shortest route (primary route), alternate route, and adaptive route between city CA and L are shown in solid-red, dotted-green, and dashed-blue lines, respectively. In the illustration, if the links such as (CA-CA $_1$), (CA-WA), (WA-CA $_1$), (CO-NE), (TX-MD), and (WA-L) are busy (denoted in Figure 2.18 as α), the adaptive-routing algorithm can still establish a connection between cities CA and L; whereas, both the FR and the FAR algorithms are unable to establish the connection [38].

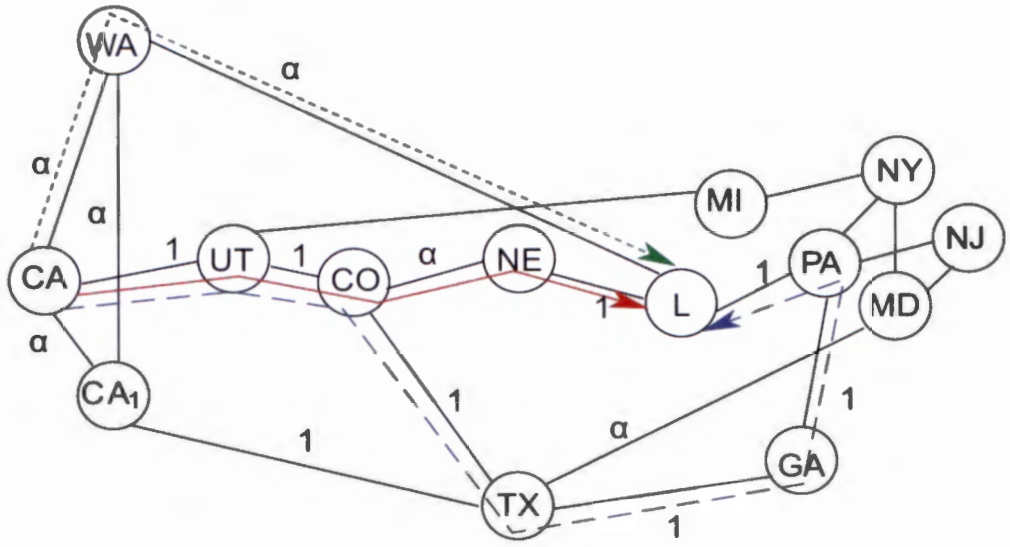


Figure 2.18: Functionality of routing algorithms [38].

2.10 Wavelength Assignment Algorithms

A wavelength employment mechanism is used to employ the best wavelength if multiple required wavelengths are accessible on the entire route between a source-destination pair. The wavelength employment may be accomplished either after a path has been discovered, or in parallel during the path employment. For optimal performance of the network it is imperative to employ the best wavelength [39]. Figure 2.19 illustrates an architectural overview of the wavelength assignment algorithm.

The wavelength assignment must adhere to the following two constraints [18, 39]:

(i) **Distinct wavelength constraint**: Two lightpaths must not be assigned to the same wavelength on the link. Therefore, all lightpaths are required to be dispersed to distinct wavelengths to avoid any interference in the optical fiber links. This constraint is fulfilled in Figure 2.20, were two lightpaths share a link (different colours represents wavelengths).

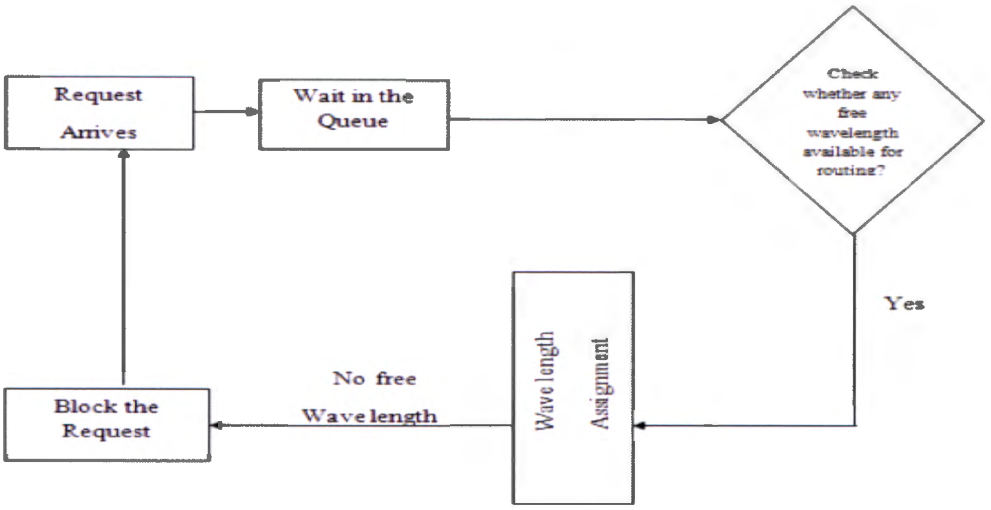


Figure 2.19: Architectural diagram for the Wavelength Assignment Algorithm [47].

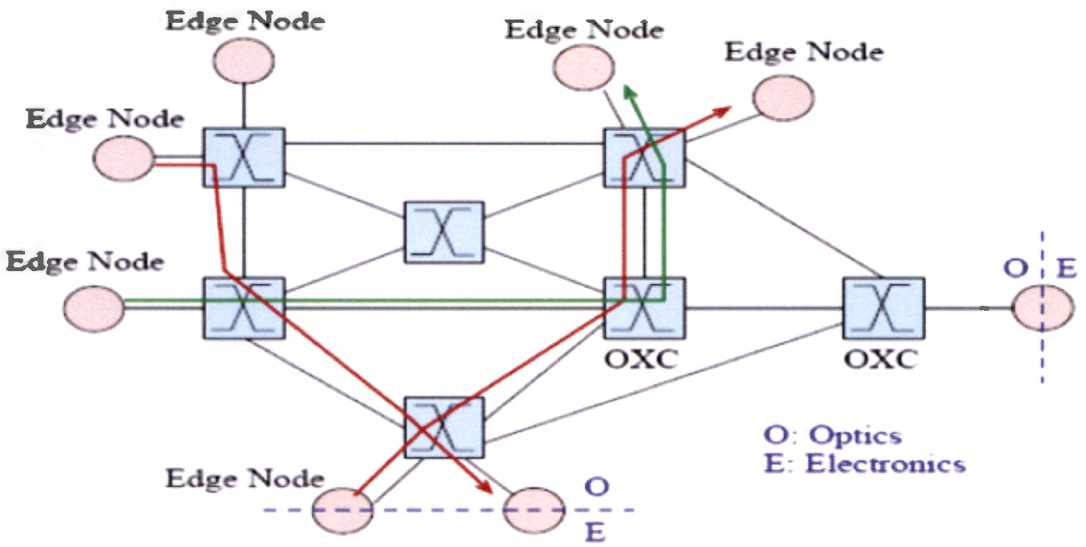


Figure 2.20: A wavelength routed through a WDM network [18].

(ii) **Wavelength continuity constraint**: If no wavelength conversion is available, then a lightpath is required to occupy the same wavelength that must be used on all fiber links along the designated route, prior to communication between any two nodes. This constraint is demonstrated in Figure 2.21, where each lightpath is denoted by a single colour (wavelength) along all the links in its path.

In a wavelength routed network the light path must be wavelength continuous; if there is no common wavelength throughout the length it results in blocking. This problem of blocking

can be overcome to an extent by the use of wavelength converters [43]. As converters are very expensive, it is not economically feasible to place converters at all nodes. Therefore, there is a trade-off between performance gain and cost. For a more practical and cost effective solution, only a few converting nodes must be used. This is known as sparse or limited wavelength conversion [43, 48].

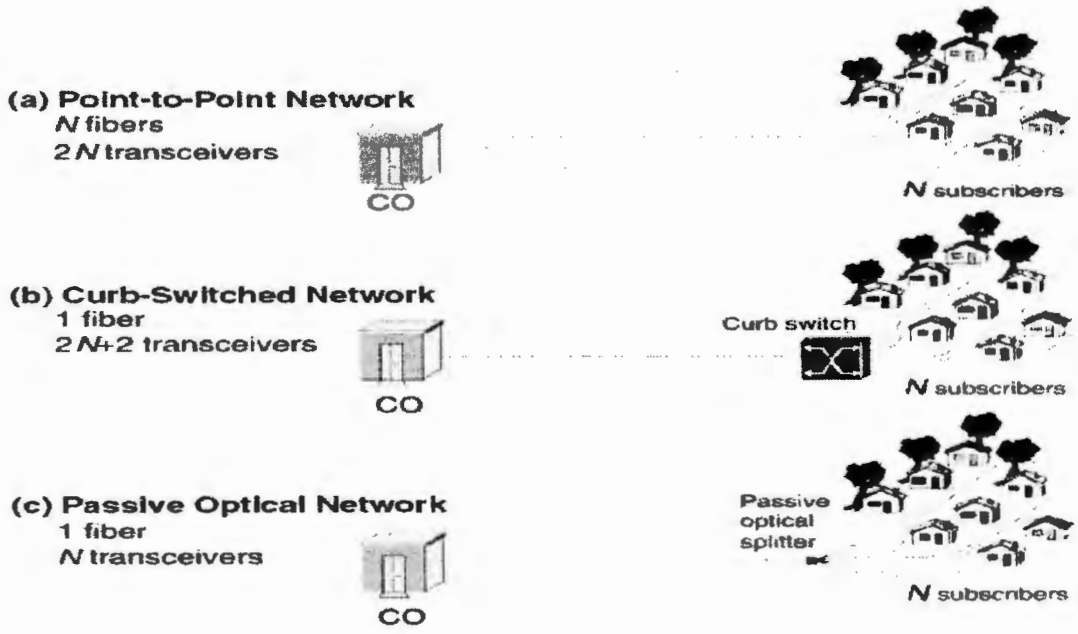


Figure 2.21: Different technologies for fiber-to-the-home (FTTH) [18].

Normally, a lightpath operates on the same wavelength across all fiber links that it traverses, in which case the lightpath is said to satisfy the WCC. Thus, two lightpaths that share a common fiber link should not be assigned the same wavelength. However, if a switching/routing node is also equipped with a wavelength converter facility, then the WCCs disappear and a lightpath may switch between different wavelengths on its route from its origin to its termination. This particular problem is referred to as the RWA problem [43].

A number of heuristics have been proposed for optimal performance of the network. Some significant heuristics such as random, most-used, and first-fit wavelength assignment algorithm are discussed in the following subsections.

(i) **Random wavelength assignment** [18, 43]: In this algorithm, a set of wavelengths that can be utilised to institute the connection is regulated. Furthermore, a wavelength is randomly employed from the set with respect to a constant probability distribution. In this method, a set of free wavelengths on a particular path is determined. Among the available free wavelengths, one is chosen randomly (usually with uniform probability) and assigned to the requested light path. When the call is completed, that particular wavelength is removed from the list of used wavelengths and is added again to the set of free wavelengths. In this manner the set of free wavelengths is updated every time a call is answered or when the holding time for the answered call is over. Even though random wavelength assignment works better than first fit assignment as it can choose any of the free wavelengths, it suffers from lack of a definite approach for wavelength assignment and that may not yield good results in some cases [49]. Algorithm 2.1 [50] shows random wavelength assignment.

Algorithm 2.1: Random Wavelength Assignment algorithm

Step1: *Initialisation of network parameters.*
Step2: *Select any source destination pair.*
Step3: *Select any root out of all possible roots for selected source destination pair.*
Step4: *Assign any wavelength out of available wavelengths.*
Step5: *Is blocking probability of path is greater than threshold value?*
Step6: *If yes, repeat step4 and select new wavelength value if no, establish network connection.*
Step7: *Repeat step 2.*

(ii) **Most-used wavelength assignment** [18, 26, 49]: In the most-used wavelength assignment, the connection request is assigned with a free wavelength that is used on the greatest number of fibers in the network. If several available wavelengths share the same maximum usage, the wavelength with a specific index is chosen. Furthermore, this algorithm considers all the accessible wavelengths that can be used to establish a connection; this means that a popular wavelength which is often used in a network is identified and employed for the connection. The wavelengths that are ideal in the most-used algorithm are more compact than those used in the first-fit algorithm. The Most-Used wavelength algorithm does not require global knowledge

of the network. This algorithm simply depends on the state of the node at that instant and chooses the wavelength from the set of free wavelengths at that output link. As it is unaware of the state of the network, this assignment strategy will not yield optimum results. Figure 2.22 shows a pseudo-code segment for the most used wavelength assignment algorithm.

```
MUW( G, T, ω, c, η )
1  do ∀ρ ∈ {1, . . . , W} →
2      A[ρ] ← True;                    { wavelength available }
3      do ∀p ∈ η(c) →
4          if ¬free(G, T, w, ρ, p) →
5              A[ρ] ← False;           { wavelength not available }
6          fi;
7      od;
8  od;
9  usage ← −∞;
10 do ∀ρ ∈ {1, . . . , W} →
11     if A[ρ] ∧ ( U[ρ] > usage ) →
12         λ ← ρ;
13         usage ← U[λ];
14     fi;
15 od;
16 U[λ] ← U[λ] + 1;
17 select-wavelength ← λ;
```

Figure 2.22: Pseudo-code for the Most Used Algorithm (MUW) [51].

In Figure 2.22, the symbols are as follows:

- $U[\lambda]$ denotes usage count for each wavelength channel,
- ρ denotes a selected wavelength,
- p denotes a link pathway in the network,
- *free* denotes a function that checks the availability of wavelength ρ on the link,
- W denotes the number of wavelength channels, and
- η denotes a function that specifies a path for a connection.

(iii) **First-Fit wavelength assignment** [36, 39]: In this algorithm, all wavelengths are indexed, and a lightpath is utilised to acknowledge the wavelengths of lower index before opting to employ wavelengths of higher index. By employing wavelengths in this manner, prevailing connections will be arrayed into more small-scaled number of aggregated wavelengths, resulting in a vast number of wavelengths accessible for extensive lightpaths. This algorithm does not require global knowledge regarding

nodes or network present status. Thus, no storage is needed for network states and no communication overhead is needed. So, the overall computational overhead is small, which results in low complexity. Its performance delivery is satisfactory in terms of blocking probability and fairness of allocation. Furthermore, it is inclined towards its small overhead and low computational complexity. The objective of this allocation scheme is to minimize wavelength fragmentation. The disadvantage of the approach is that the lower indexed wavelengths are much more used than the higher indexed wavelengths. Hence certain wavelengths are very under-utilized. Since all the nodes in the network use the lower numbered wavelengths, contention for these wavelengths increases which results in higher blocking probability [50].

In [52] a simple heuristic algorithm called First-Fit for wavelength assignment is designed to achieve the objective, subject to the WCC. Equivalently to the description above, the wavelength allocated is the first available wavelength on the first available fiber considering all physical links along the given routing path for each lightpath. The steps in this algorithm are as follows [52]:

i. Given the fixed routing path for each lightpath, from the first wavelength, check whether the wavelength that is not used on some visited fibers of each links along the path.

ii. If so, use this wavelength on the first available fiber of each link and record that it is used on these links, otherwise check the next one. A state variable – lambda - is used to check if the current wavelength is available on current fiber or not. If lambda is 0, that is available. If all the wavelengths are not available on the visited fibers of each link, try to use a new fiber for each link and check from the first wavelength again and assign the first available wavelength on the first available fiber.

iii. Not all the links in the routing path need to use a new fiber. For example, if wavelength 1 is available on some previously visited fiber of all the links except link *i*, in this case only link *i* needs to use a new fiber. Another variable - fiber_state for the fiber - is used to determine whether a new fiber needs to be used or not. If the final wavelength W is not available, try to open a new fiber for each link and the variable - fiber_state is set to 1 for the new fiber of each link.

iv. Then find the first available wavelength assigned on the first available fiber of each link, starting from the first fiber to the newly opened fiber of each link.

v. Then set the variable - fiber_state of each fiber again. If the variable - fiber_visited of a fiber is 1, the variable - fiber_state is set to 1. Otherwise, the variable - fiber_state is set to 0. So if the variable - fiber_state of the newly opened fiber is 1, this fiber needs to be used. But we do not need to use the newly opened fibers in which the variable—fiber_state is 0 and the amplifiers in these fibers will be still turned off. In this way, we can use as few fibers as possible.

The pseudocode of First-Fit algorithm is as follows [52]:

Algorithm 2.2: First-Fit Algorithm

```
1.   for each lightpath i
2.    w := -1 // w is used to check which wavelength is available for lightpath i
3.    do
4.      w := w + 1
5.      ok := 1
6.      if w = = W
7.       for each link in the selected path
8.         while fiber_state = =1 (from the first fiber)
9.                turn to next fiber
10.        Fiber_state :=1     //try to use a new fiber
11.      w := 0  //use the first wavelength
12.      for each link in the selected path
13.         while fiber_state == 1 && lambda == 0
14.            turn to next fiber
15.         if fiber_state == 0 // turn to a unused fiber
16.           ok := 0
17.   while ok = = 0 //use the current wavelength w if ok = = 1
18.   for each link in the selected path
19.      while fiber_state = =1 && lambda != 0
20.      turn to next fiber
21.        lambda := i  // the first available fiber uses wavelength w for lightpath i
22.        fiber_visited := 1
23.      for each fiber in the link
24.        if fiber_visited = = 1
```

```
25.             fiber_state := 1
26.         else
27.             fiber_state := 0
```

The functionality of the above wavelength assignment algorithms is explained with a network segment, which is shown in Figure 2.23. If we want to establish a lightpath between Node-13 to Node-10 in Figure 2.16, we observe that two wavelengths (λ_1 and λ_2) are available. If we use First-Fit algorithm, λ_1 will be assigned. Since wavelength λ_1 and λ_2 are used eight times and four times, respectively, in the network segment, λ_2 and λ_1 are used in Most-Used algorithms, respectively. In the Random scheme, any of the two wavelengths can be chosen with equal probability [39]. Table 2.2 shows performance analysis of various wavelength assignment algorithms.

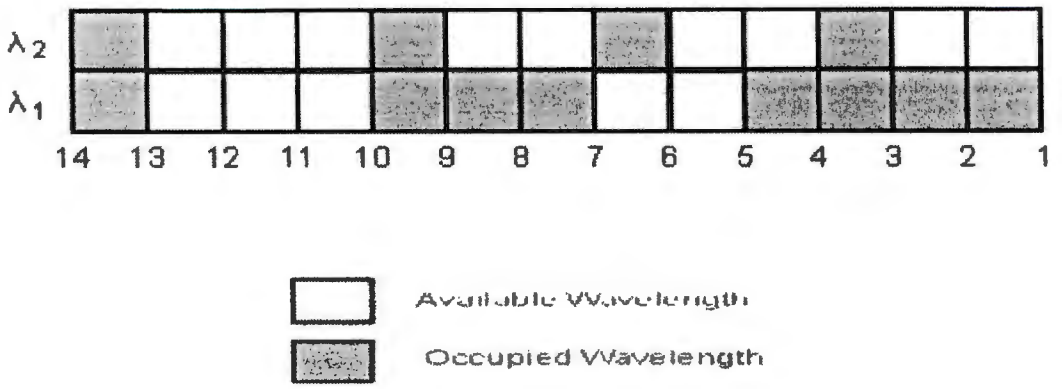


Figure 2.23: Wavelength-usage pattern for a network segment [39].

Table 2.2: Details of different wavelength assignment schemes [39].

Problem	Approach	Performance Analysis		Applicable Network
		Blocking Probability	Time Complexity	
WA+ FR	Random	More BP than FF but almost close	O (L1 W Z)	Single/multi-fiber networks
	Most Used	MU perfoms well under low load	O (L1 L2 W Z)	Single/multi-fiber networks
	First Fit	Less BP than MU	O (L1 W Z)	Single/multi-fiber networks

The notations used in Table 2.2 are explained as follows [39]:

(i) L1 denotes the length of the longest fixed route for any node pair,

(ii) L2 denotes total number of links in the network,

(iii) W denotes number of wavelengths per fiber link,

(iv) Z denotes total number of connection requests,

(v) WA+FR denotes wavelength assignment +fixed routing,

(vi) MU denotes Most Used,

(vii) BP denotes Blocking Probability,

(viii) FF denotes First Fit.

2.11 Conclusion

This chapter presented an overview on Wavelength Assignment in WDM optical networks. It then briefly discussed wavelength routing algorithms that play a vital role in wavelength assignment algorithms. Furthermore, popular heuristics in wavelength assignment algorithm namely, random wavelength assignment algorithm, Most-Used wavelength assignment and First-Fit wavelength assignment were discussed and presented in the chapter. The next chapter presents a hybrid wavelength assignment algorithm as well as the simulation setup.

CHAPTER 3

ALGORITHM DEVELOPING, MODELING AND SIMULATION

3.1. Chapter Overview

In this chapter the design of three wavelength assignment algorithms - namely, improved first-fit algorithm (IFF), improved random wavelength assignment (IR), and hybrid algorithm - is reported. Firstly, IFF algorithm is the improved version of the first-fit algorithm, and the IR algorithm is also the enhancement of the random algorithm. Secondly, the hybrid algorithm is composed of IFF and IR. Furthermore, the chapter also presents the modeling and simulation of the above mentioned wavelength assignment algorithms. The simulation tool, along with the parameters used, is discussed.

3.2. Wavelength Assignment Algorithms

In this section the improved heuristics of wavelength assignment algorithms is considered, and the hybrid algorithm is introduced.

3.2.1 Improved First-fit wavelength assignment algorithm

Generally, the first-fit algorithm is best known for indexing wavelengths from low to high, so that when a lightpath requests a wavelength for transmission, it begins with the low indexed wavelengths before employing the high indexed wavelengths. This method of indexing does not precisely show how wavelengths are arranged. The new IFF algorithm, which is an enhancement of the first-fit algorithm, introduces a new approach that will arrange wavelengths in terms of size; this approach is known as insertion sort function [53]. The insertion sort, which is a simple sorting algorithm that constructs the final sorted array one element at a time, is going to be used on the first-fit algorithm to propose an improved version of first-fit. Therefore, a lightpath would request a wavelength from a sorted range of wavelengths, and a wavelength with a lower index will be selected before a wavelength with a higher index is selected. A segment pseudo-code for insertion sort is presented as Algorithm 3.1.

Algorithm 3.1: Insertion Sort

```
void sort (Item a[], int start, int stop)
{
int w, j;
for (w = start + 1; w<= stop; w++)
for (j = w ; j >start; j--)
if (isLess(a[j], a[j-1])
Exchange (a[j-1], a[j]);
else
break;
}
```

Where,

- w denotes a selected wavelength, and
- j denotes a temporary location for a wavelength.

Now, an insertion sort function is added to the original first-fit algorithm, which now produces a new algorithm called improved first-fit algorithm (IFF). IFF, as specified is the enhancement of the first-fit algorithm. The insertion sort function assures that the wavelengths are sorted first before they can be assigned to lightpath request in the network. The IFF algorithm is presented as Algorithm 3.2.

Given L_{pi}, check if any wavelength is not used on visited fibers of each link along the path. If so, use the insertion sort function to arrange wavelengths in terms of size from low index to high index. Now, from the sorted set of wavelengths, use the first available wavelength on the first available fiber of each link and record that it is used on these links, otherwise check the next one. If lambda is 0, meaning that the wavelength is available on the current fiber, use the next fiber.

If all the wavelengths are not available on the visited fibers of each link, we try to use a new fiber for each link and check from the first wavelength again and assign the first available wavelength on the first available fiber. In actual fact, not all links in the routing path need to

use a new fiber. For instance, if wavelength 1 is available on some previously visited fiber of all the links except link *i*, in this case only link *i* needs to use a new fiber. If the final wavelength *W* is not available, we try to open a new fiber for each link and Ω_s is set to 1 for the new fiber of each link. Then we find the first available wavelength assigned on the first available fiber of each link, starting from the first fiber to the newly opened fiber of each link. Then we set the Ω_s of each fiber again. If the Ω_v is 1, then Ω_s is set to 1. Otherwise, Ω_s is set to 0. So if the Ω_s of the newly opened fiber is 1, we need to use this fiber. But we do not need to use the newly opened fibers in which Ω_s is 0 and the amplifiers in these fibers will be still turned off. In this way, we can use as few fibers as possible.

The important notation used in the algorithm is first listed. Unless otherwise stated, let:

- L_{pi} denote each lightpath i on the fixed routing path.
- w denote the wavelength used to traverse through the fiber.
- W_t denote the total number of available wavelengths.
- λ denote lambda, a state variable, which will check if the current wavelength is available on current fiber or not.
- F_l denote each fiber in the link.
- S_p denote each link in the selected path.
- Ω_s denotes the fiber_state of the fiber, which determines whether we need to use a newfiber or not.
- Ω_v denote the fiber_visited.

Algorithm 3.2: IFFAlgorithm

1. **for** L_{pi}
2. **Insertion sort** (w)
3. w := -1 // w is used to check which wavelength is available for lightpath i
4. **do**
5. w = w + 1
6. ok = 1
7. **if** w = = W_t

8. **for** S_p

9. **while** $\Omega_s = =1$ (from the first fiber)

10. turn to next fiber

11. $\Omega_s = 1$ // try to use a new fiber

12. w = 0 //use the first wavelength

13. **for** S_p

14. **while** $\Omega_s = =1$ && $\lambda = = 0$

15. turn to next fiber

16. **if** $\Omega_s = = 0$ // turn to a unused fiber

17. ok = 0

18. **while** ok = = 0 //use the current wavelength w if ok = = 1

19. **for** S_p

20. **while** $\Omega_s = =1$ && λ != 0

21. turn to next fiber

22. λ = i // the first available fiber uses wavelength w for lightpath i

23. $\Omega_v = 1$

24. **for** F_l

25. **if** $\Omega_v = = 1$

26. $\Omega_s = 1$

27. **else**

28. $\Omega_s = 0$

3.2.2 Improved Random wavelength assignment algorithm

The new algorithm, improved random wavelength assignment algorithm (IR), is an enhancement of the random wavelength assignment [50]. In IR, the wavelength is selected randomly from the set of available wavelengths. In this algorithm, a lightpath selects any source destination (*s-d*) pair for wavelength transmission. Out of the selected *s-d* pair, possible roots are then uniquely identified for network utilization. Now, random select procedure is used to randomly select any wavelength from a set of available ones, and assigns

it to the lightpath that will alternatively transmit it through the uniquely selected s-d pair. Now, if the link utilization of the path is greater than the threshold value, then the lightpath will be prompted to request another wavelength that will be selected randomly. Otherwise, a network connection will be established. The new algorithm, IR is presented in a pseudo-code form, as Algorithm 3.3.

Algorithm 3.3: Improved Random Wavelength Algorithm (IR)

Step1: **PROCEDURE** Random_Select ()

Step2: **BEGIN** Random_Select

Step3: **INITIALISE** *network parameters*

Step4: **SELECT** *any source destination pair*

Step5: **END-SELECT**

Step6: **SELECT** *any root out of all possible roots for selected source destination pair*

Step7: **END-SELECT**

Step8: random_select (w); // assign any wavelength out of available wavelengths

Step9: **IF** (*link utilization of path* > *threshold value*)

Step10: **THEN** random_select (w); // assign any wavelength out of available
// wavelengths and select new wavelength value

Step11: **ELSE** *establish network connection*

Step12: **END-IF**

Step13: **END** Random_Select

3.2.3 Hybrid wavelength assignment algorithm

The hybrid algorithm is composed of two improved heuristics of wavelength assignment. These two heuristics are IFF and IR, improved from [50, 52]. Furthermore, the hybrid algorithm captures the concepts of first-fit and random wavelength assignment. Figure 3.1 shows a flow chart of the hybrid algorithm, where a lightpath selects source-destination (s-d) pairs for wavelength transmission on the network. The hybrid algorithm then selects the IFF approach, where the wavelengths are arranged from low index to high index, as explained in 3.2.1. Now, two instances are considered in this approach namely, (i) low indexed wavelengths versus link channel, and (ii) low indexed wavelengths versus threshold value. If the low indexed wavelengths become overwhelming to the link channel, it would mean that the high index keeps pushing to the end of the queue in lightpath requests. Therefore, if the low index wavelengths are above the threshold value, then switch to IR so that both the low and high index have equal chances of being selected. If the selected wavelength is below the threshold value, then continue transmitting until the destination node; otherwise go back to the IFF approach. The hybrid algorithm is presented below in Algorithm 3.4, and Figure 3.1 below shows a flow chart for the hybrid algorithm.

Algorithm 3.4: Hybrid Wavelength Assignment Algorithm

Step1: **PROCEDURE** Random-First-Fit Algorithm ()

Step2: **BEGIN** Random-First-Fit Algorithm

Step3: **SELECT** *any source destination pair*

Step4: **END-SELECT**

Step5: **USE** *IFF Algorithm*

Step6: **IF** (*low index wavelength* > *threshold value*)

Step7: **THEN**

Step8: **USE** *IR Algorithm*

Step9: **IF** (*selected wavelength > threshold value*)

Step10: **THEN**

Step11: **USE** *IFF Algorithm*

Step12: **ELSE** *terminate*

Step13: **END-IF**

Step14: **END** Random-First-Fit Algorithm ()

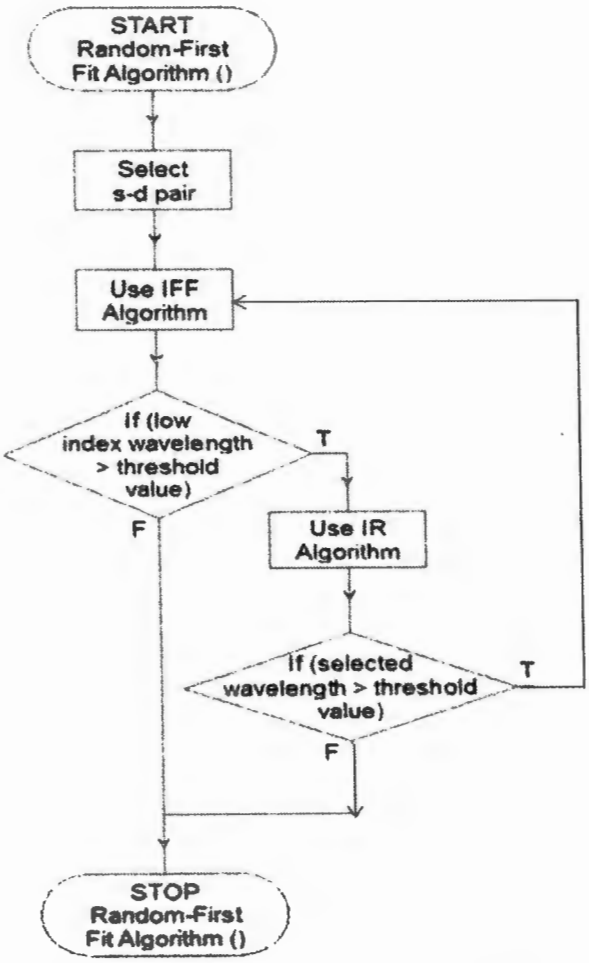


Figure 3.1: Flowchart of the hybrid algorithm [50, 52].

3.3. Simulation Software Tool

In Wavelength assignment algorithm research, several simulation software tools exist. Amongst these simulation software tools are [54]:

(i) **MATLAB**: This tool is a high-level language and interactive environment for numerical computation, visualization, and programming. It has user-friendly environment that allows a user to analyze data, develop algorithms, and create models and applications. Traditional programming languages used in this simulator are, C, C++ and Java.

(ii) **OMNeT++**: OMNeT++ is a discrete event simulation environment. It presents a component structural design for models. Components, known as modules, are programmed in C++, which is the programming language used.

(iii) **NS-2**: NS-2 is a discrete event simulator that simulates networks. This tool provides extensive support for simulation of TCP, routing, and multicast protocols over wired and wireless networks.

(iv) **OPNET**: OPNET is a network modeling simulator that analyses network communications and application. It is also a discrete event simulator and it uses programming language C++.

For the purpose of this research, OPNET Modeler 14.5 was used for modeling and simulations.

3.3.1 OPNET (Optimized Network Engineering Tools)

OPNET is a simulation tool that provides a comprehensive development environment supporting the modelling of communication networks and distributed systems [55]. The performance of modelled systems can be analysed by performing distinct simulations. The OPNET environment includes tools for all phases of a study, such as model design, simulation, data collection, and data analysis. The following are some of the advantages of OPNET [46]:

(i) OPNET provides a user-friendly graphical editor that allows the user to edit devices, configure networks, design protocols, and define packet formats.

(ii) OPNET supports several model families over the wireless network to enable communication.

(iii) OPNET has a fast discrete event simulation engine.

OPNET comprises of a hierarchical structure which is subdivided into three categories, namely, network domain, node domain, and the process domain [54, 55, 56]. To construct a network model the following OPNET workflow structure has to be followed in order to have a well-constructed error-free system. The workflow is comprised of four main steps: modelling outline, selecting statistics, running simulation and lastly, viewing and evaluating results. Figure 3.2 gives an overview of OPNET workflow.

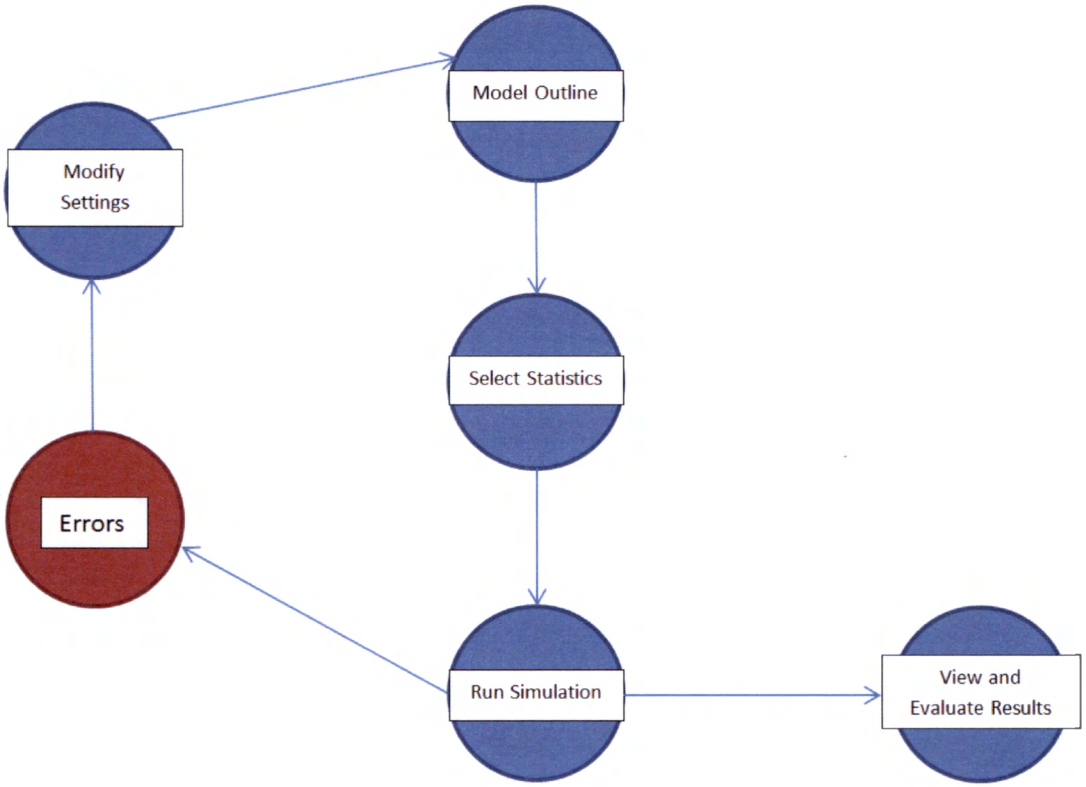


Figure 3.2: OPNET Workflow [54].

3.3.2 Performance Metrics

The evaluation of the network performance can be done using different metrics such as the throughput, the load, the delay, the jitter, the packet delivery ratio, link utilization, the routing overhead, the number of transmissions per packet and many others. However in this

dissertation, link utilization, queuing delay, and throughput are the metrics used. The summary of the performance metrics used in this thesis is summarized in Table 3.1:

Table 3.1: Summary of the performance metrics of WDM optical networks [57].

Performance Metric	Definition	Desired function
Link Utilization	It is the bandwidth that a traffic stream takes from the total link capacity.	High utilization is desirable.
Queuing Delay	The time required for bit (packet, file) to go from source nodes to destination node. Measured in seconds.	Low delay is desirable.
Throughput	It is the total number of data packets in bits transmitted from one node to the other.	High throughput is desirable.

3.3.3 Wide area IP network topology

The WDM optical model consists of four subnets in this network topology, namely, JHB_subnet (Johannesburg subnet), DBN_subnet (Durban subnet), CPT_subnet (Cape Town Subnet) and WHK_subnet (Windhoek subnet). The traffic in these subnets is routed to an IP cloud object, ISP central, via gateway routes. The four subnets and the ISP central are connected by links of PPP_DS3 model. In JHB_subnet, there are thirty-six client nodes and five servers, which are connected to three switches, which are also connected to two gateway routers and a firewall router. Furthermore, DBN_subnet, CPT_subnet and WHK_subnet, have the same network setup structure, and each consists of twelve central networks connected to a switch which is also connected to two gateway routers and a firewall router. More details on the respective subnet network setups are given in appendix A.

Network domain [54, 55, 56]: A network model outlines an overview topology of the network communication system to be simulated. This model entails a number of important components that make up a network system, and these components are labelled as subnets,

nodes, links, and geographical coordinates. The network model used for the NSFNET Network Topology is shown in Figure 3.3.

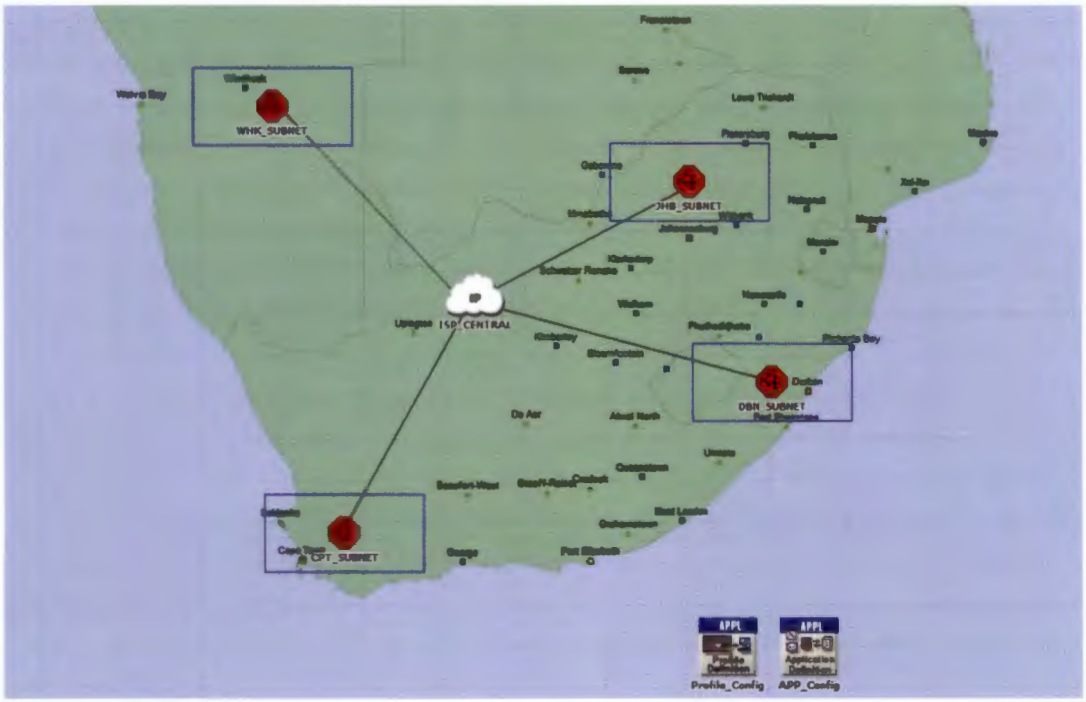


Figure 3.3: Project Area Network Setup.

Node domain [54, 55, 56]**:** A node domain specifies the internal structure of a network node. Subsequently, a node model can represent a computer, a switch, a router, or a network cloud. This node model is made up of small building blocks connected together, called modules. These modules are separated by logical functionalities and are able to communicate with each other via packet streams and statistic streams, which correspondingly connect the modules. Modules are used to transmit packets, receive packets, process data, store data, route packets, etc. Furthermore, modules include processor modules, queue modules, transceiver modules, antenna modules, and external system modules. Adding to this, a network in a node domain includes workstations, packet switches, satellite terminals and remote sensors. Figure 3.4 shows the node model used in the wide area IP network topology.

Process Domain [54, 55, 56]**:** A node model may contain several modules, each of which has a particular functionality. A module should contain a process model that actually implements the functionality or logic the module represents. A process model specifies the behaviour of the processor modules which exist in the node domain. One module is modelled and represented as a finite state machine (FSM), which expresses the behaviour that depends on

current state. Figure 3.5 gives an overview structure of the process model of the node model and the traffic model.

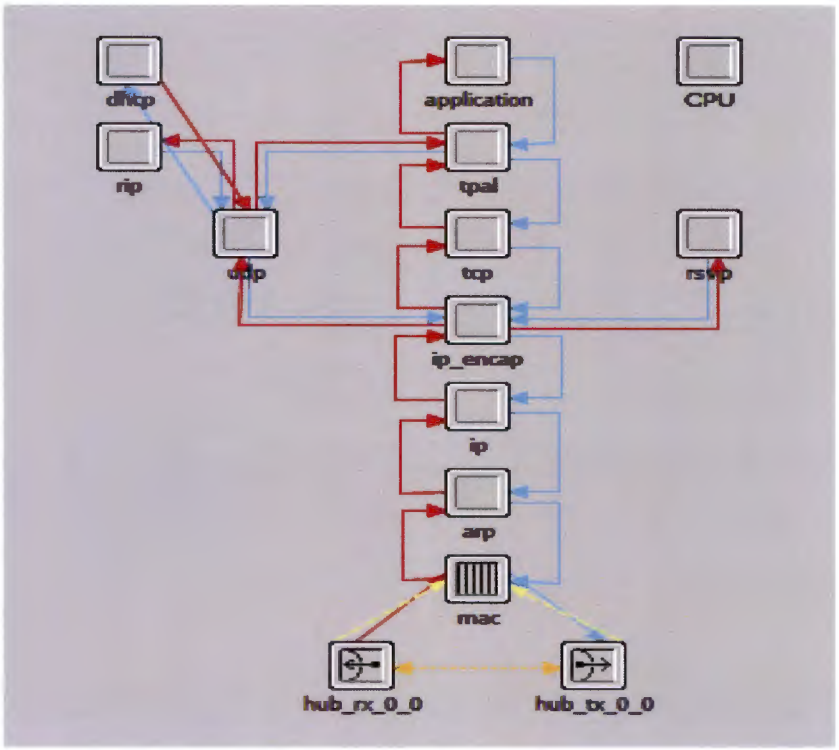


Figure 3.4: Simulation Node Model.

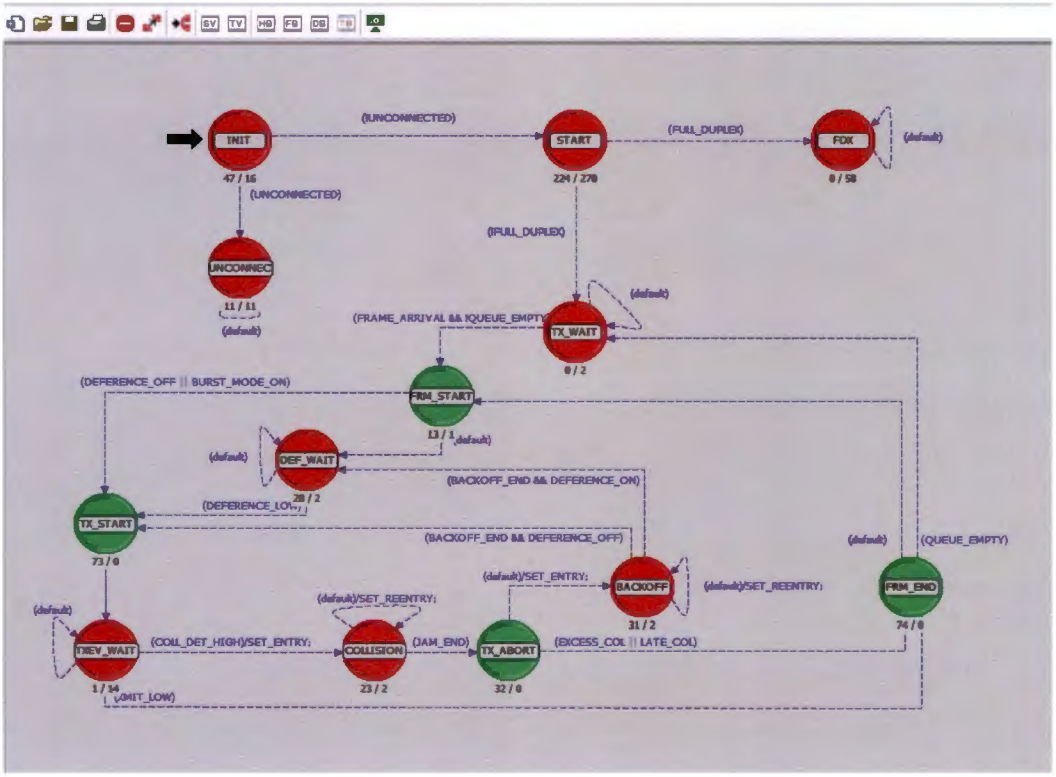


Figure 3.5: Process Model of the Project Area Network Setup.

3.4. Conclusion

In this chapter, three wavelength assignment algorithms were designed, namely, IFF, IR, and a hybrid algorithm. The IFF algorithm is made significant by an insertion sort function, which sorts wavelengths according to size, from the smallest to the largest. The IR algorithm selects a wavelength at random from the set of available wavelengths, while comparing the link utilization to the threshold value. The hybrid algorithm conceptualizes both the IFF and the IR. Furthermore, this chapter presented the modelling, design and simulation of the wavelength assignment algorithms.

The next chapter presents the analysis, evaluation and discussion of the results obtained from the modelling and simulation of the wavelength assignment algorithms and the improved algorithms.

CHAPTER 4

RESULTS AND DISCUSSION

4.1. Chapter Overview

This chapter presents the results obtained from the simulation and implementation as measured against the performance metrics. The performance metrics used for analysis are queuing delay, throughput and link utilization.

4.2. Simulation Results for Wavelength Assignment Algorithms

This section presents the results of the wavelength assignment algorithms.

4.2.1 Queuing Delay

In terms of queuing delay, the performance of the first-fit, improved first-fit, random, improved random, and the hybrid algorithm are illustrated respectively in Figure 4.1. The results in Figure 4.1 show that the hybrid algorithm performs significantly well from point 0 to 110 (sec). Moreover, looking at IFF from point 0.0000142 and 0.0000158 (average seconds), it is noticeable that the algorithm projects an increased queuing delay over time zero. This is due to the fact that all wavelengths from the set of sorted wavelengths get an opportunity to be transmitted through the network links.

As soon as running time resumes, its queuing delay decreases, this shows that the lightpaths are successfully transmitting the wavelength without any disturbance from the traffic flow. From point 110 to 300 (sec), the pattern performance of the algorithms changes considerably. The RAND gives a more stable and decreased queuing delay, while other algorithms project an increased queuing delay. This is due to the fact that in the RAND algorithm, wavelengths are selected at random which is consequential in being processed and transmitted faster, hence a low queuing delay. This sudden pattern change shows that the traffic flow in the network is congested; this is caused by low indexed wavelengths overpowering the link

channel. In this case, the high indexed wavelengths are being pushed to the end of the queue in lightpath requests, hence an increased queuing delay.

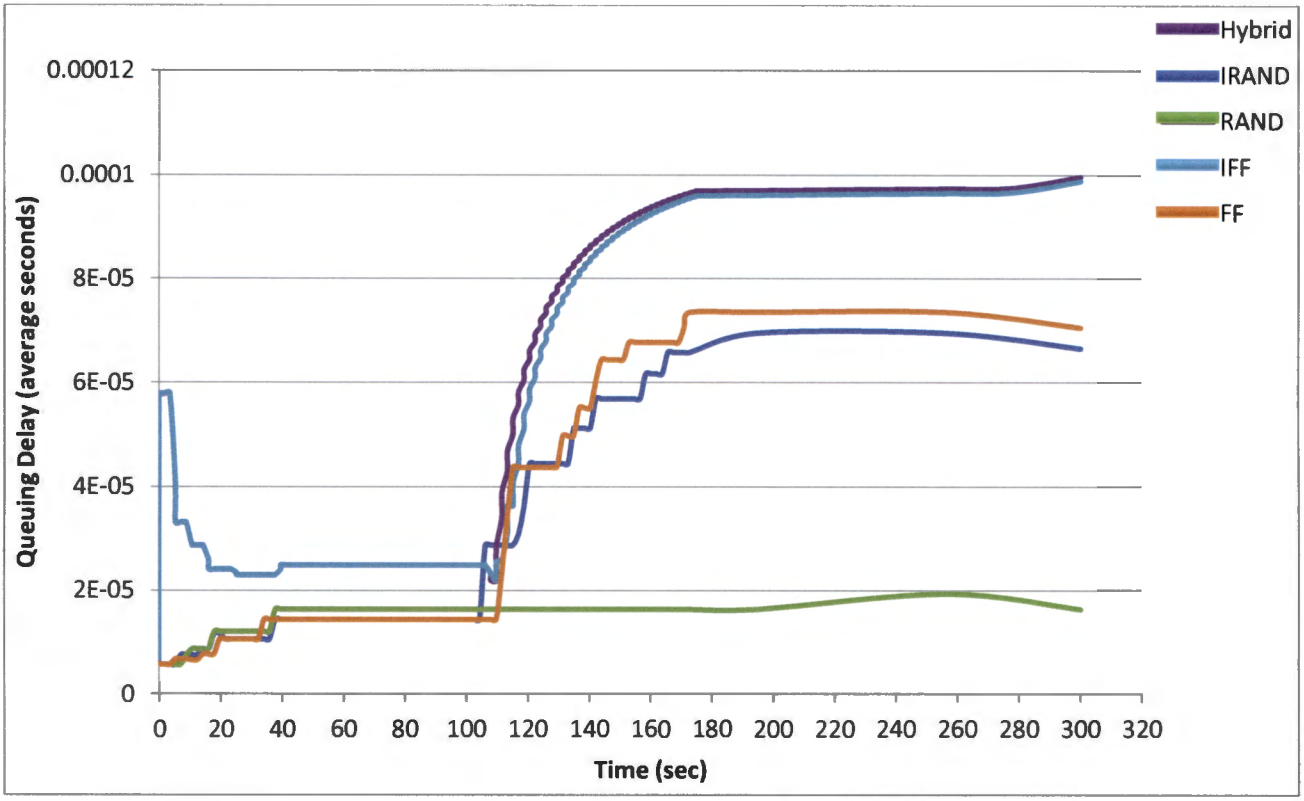


Figure 4.1: Results for Queuing Delay (average seconds).

4.2.2 Throughput

The delivery of messages through a network is an important factor that determines whether the system preserves computational time or not. From point 0 to 97 in Figure 4.2, there is data transmission from the source node to the destination node, hence a constant barrier line in all algorithms. There are various reasons to explain the non-receptive transmission in the algorithms, one of which might be that the data is too large to be transmitted causing too much bandwidth to be consumed.

From point 97 onwards, there is activity taking place in all the heuristics. The RAND algorithm seems to be performing poorly compared to the other algorithms. This may be because the RAND algorithm is selecting wavelengths at random, which causes a relatively slow transmission in the network and, tends to produce a limited number of random bits per

second. The FF is performing significantly better that the IRAND, as compared to their activity in 4.2.1 above, where the two heurists were competing with each other at a harmonic pace.

At point 180, FF experiences a sudden drop in data transmission, giving IRAND an opportunity to perform better, but as soon as point 198 is reached, FF proactivity increases. At points 200, 240 and 260, the results of the two heurists fluctuate; at some points IRAND performs better than FF and at some points it performs worse. The hybrid outperforms all the algorithms. Even when the IFF performs better than the other algorithms it does not outperform of the hybrid algorithm.

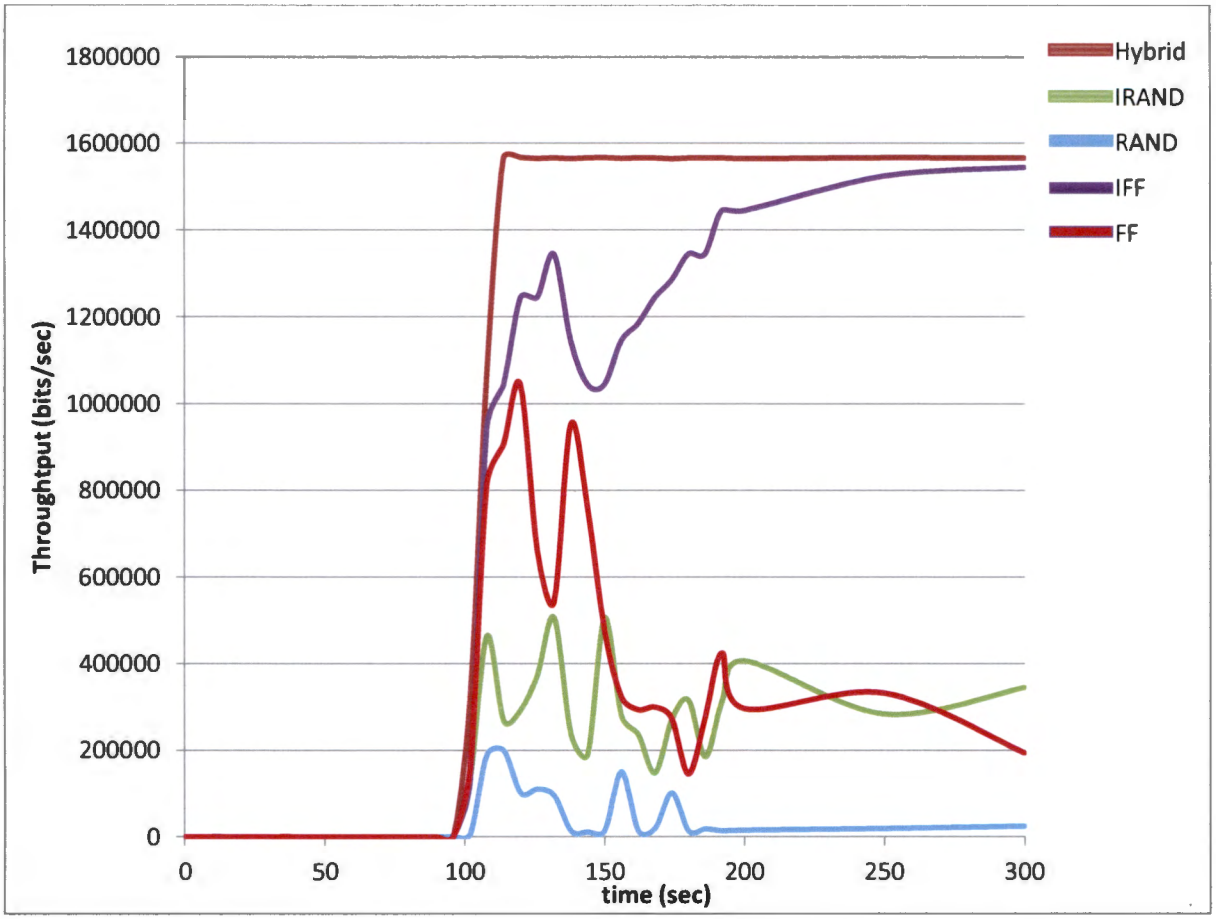


Figure 4.2: Results for Throughput (bits/sec).

4.2.3 Link Utilization

In terms of link utilization, hybrid performs outstandingly better than all the other algorithms illustrated in Figure 4.3. From the point 0 to approximately 110, there seems to be less activity in the algorithms, although point 0 to 50 shows that there is something taking place in that interval region. One of reasons for such behaviour is traffic in the link channels. Immediately after the lightpath requests have attained wavelengths for transmission, they queue up so that they may transmit the wavelength from the source node to the designated destination node. As some wavelengths are being transmitted through the optical links, the other lightpaths have to wait for the busy wavelength transmission to finish. This causes a delay as traffic waits for a particular job to finish.

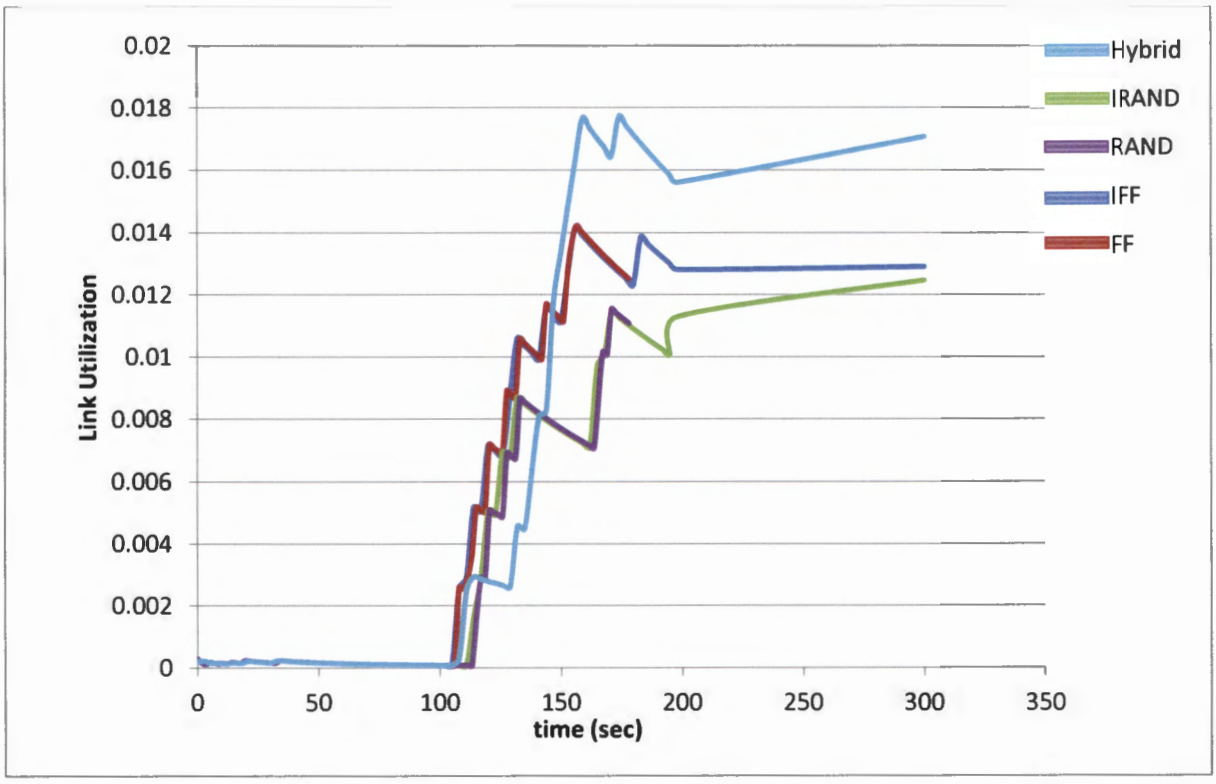


Figure 4.3: Results of Link Utilization.

From point 110 onwards, the activity of the algorithms improves increasingly. One possible reason for this is that the wavelengths are buffered in small packets, which reduces queuing delays of lightpaths. IFF and FF seem to be transmitting at the same rate, but as soon as FF

reaches point 180 it stops. Similarly, for IRAND and RAND, the transmission rate is almost similar, and at point 180, the transmission of RAND also stops. This sudden stop may be the cause of packet loss in the network due to network congestion and bit errors. Whilst the activity of these two algorithms stops, the others continue to transmit. From point 200 onwards, IFF and IRAND are converging, while the hybrid is increasingly transmitting.

4.3. Conclusion

This chapter presented the results and performance evaluation of the wavelength assignment algorithms with respect to queuing delay, throughput and link utilization. The results show that the hybrid algorithm outperforms all the other algorithms in all the performance metrics, while the IFF and IR perform better than the original algorithms. From the overall results, hybrid performs considerably better than the other algorithms, although sometimes (or at some points) IFF was at close range transmission. Furthermore, IFF also outperforms the other algorithms.

CHAPTER 5

SUMMARY, CONCLUSION AND FUTURE WORK

5.1. Summary

Optical networks employing the WDM technique are considered to be an effective and cost-efficient solution capable of providing the unrestricted capacities that are required to satisfy the enormous growth of traffic anticipated in next-generation networks. The main objective in WDM optical networks is either to maximize the number of all-optical connections, also known as lightpaths, or to minimize the blocking probability employing the limited network resources. Resolving the lightpath RWA problem efficiently is one of the most important issues in WDM optical networking. RWA is a two-way procedure, where the first method is finding an appropriate route for a traffic demand from the source to the destination node, and the second method is assigning a wavelength to the selected route. An end-to-end lightpath has to be established prior to the communication between any two nodes. The establishment of a lightpath requires that the same wavelength be assigned on all the links along the path. This restriction is referred to as WCC. Wavelength assignment is a unique feature in which wavelengths are searched before being allocated to the path selected. There are some important wavelength assignment algorithms in use, such as first-fit, random, and most-used algorithms.

The main objective of this research was to design, implement and simulate a novel wavelength assignment algorithm for WDM optical networks and compare the obtained results with the existing algorithms. This objective was achieved by evaluating three key objectives, as specified in the first chapter, as follows:

(i) Develop a novel wavelength assignment algorithm for WDM optical network.

Three wavelength assignment algorithms were designed, namely, IFF, IR, and a hybrid algorithm. IFF wavelength assignment algorithm uses an insertion sort function to sort a set of wavelengths from small index to high index. Then a wavelength of a small index is selected from the set of wavelengths. IR on the other hand carries the concept of random wavelength assignment, where a

wavelength is selected randomly from a set of available wavelengths. Furthermore, link utilization of a path is compared to the threshold value so that an appropriate path is used to transmit wavelengths across. The hybrid algorithm comprises of IFF and IR. Two cases are considered in this approach firstly, if the low indexed wavelengths become overwhelming to the link channel, it would mean that the high index keep being pushed to the end of the queue in lightpath requests. Secondly, if the low index wavelengths are below the threshold, then a switch to IR is performed so that both the low and high index have equal chances of being selected.

(ii) Implement the proposed algorithm

The improved wavelength assignment algorithms were implemented and benchmarked with existing algorithms.

(iii) Evaluate the results obtained with existing algorithms.

As a proof of concept, the results obtained from simulation show that the hybrid algorithm outperforms all the other algorithms in all the performance metrics, while the IFF and IR perform better than the original algorithms. From the overall results, hybrid performs considerably better than the other algorithms, although at some points IFF was at close range transmission. Furthermore, IFF also outperforms the other algorithms.

5.2. Concluding Remarks

In general, there are two preliminary requirements in wavelength assignment with respect to WDM optical networks, which are the discovery of routes from the source node and a destination node, and the assignment of wavelengths to the identified routes. For the fulfilment of the requirements, a wavelength assignment algorithm is required to select a wavelength for a given lightpath. The wavelength selection may be performed either after a route has been determined, or in parallel with ending a route. Since the same wavelength must be used on all links in a lightpath, it is important that wavelengths are chosen in a way which attempts to increase throughput and link utilization, and reduce queuing delay in the network.

The results in Chapters 3 and 4 present a simulation methodology for performance analysis of optical networks using the improved first-fit algorithm (IFF), and improved random wavelength assignment (IR). The simulation results revealed that the proposed wavelength assignment algorithm achieves an increased link utilization and throughput compared to the other existing wavelength assignment algorithms; although the hybrid algorithm gave poor results in the queuing delay. Consequently, it is clear that the proposed algorithm achieves good results under the indicated performance metrics.

5.3. Future Work

The main focus of this study in wavelength assignment algorithms for WDM optical networks was to evaluate the probability of the proposed algorithm under the following performance metrics: queuing delay, throughput and link utilization. Although the proposed algorithm gives satisfactory results in throughput and link utilization, it also projects an observable defect through increased queuing delay rate. To understand the cause of such a defect, a comprehensive analysis is required, which is suggested for future investigation.

References

[1] Huawei. "Huawei SoftCOM". *Reshaping the future of network architecture*, pp. 1-16, 2013.

[2] H. Nakamoto, A. Sugiyama and A. Utsumi, "Submarine Optical Communications System Providing Global Communications Network", *FUJITSU Science Technology Journal*, Vol. 45, No. 4, pp. 386–391, 2009.

[3] P. Marcell, "Resource Optimization in Optical Networks and Peer-to-Peer Traffic Identification in IP Networks", Ph.D. thesis, Budapest University of Technology and Economics, Hungary, 2009.

[4] N. Zhang, "Research on Control Routing Technology in Communication Network", *International Journal of Applied Mathematics & Information Science*, vol. 6, no. 1S, pp. 129S-133S, 2012.

[5] Y. Ito, "A New Paradigm in Optical Communications and Networks", *IEEE Communications Magazine*, vol.51, no.3, pp.24-26, 2013.

[6] N. Nag and I. Bala, "Dynamic WDM Routing solutions", *International Journal of Engineering Research & Technology (IJERT)*, Vol. 2 Issue 9, pp. 2813-2816, 2013.

[7] K. Christodoulopoulos, K. Manousakis and E. Varvarigos, "Reach Adapting Algorithms for Mixed Line Rate WDM Transport Networks", *Journal of Lightwave Technology*, vol. 29, issue 21, pp. 3350-3363, 2011.

[8] J. Wang, Z. He and J. Wu, "A wavelength protection algorithm based-on wavelength utilization ratio in DWDM Optical Networks", *International Proceedings of Computer Science & Information Tech*, vol. 27, p170-175, 2012.

[9] A. Sharma, S. Sachdeva and A. Kakkar, "Optimized WDM network with consideration of lesser blocking probability & shortest path selection", *IEEE International Conference on Signal Processing, Computing and Control (ISPCC)*, pp.1-6, 2012.

[10] S. Chaturvedi and P. Dutta, "Utilization of Optical Fibre WDM Channel in Wavelength Routed Networks using Sparse Partial Limited Wavelength Conversion", *International Journal of Engineering Trends and Technology*, Vol. 4, Issue 4, pp. 546-549, 2013.

[11] K. Aparna, S. Venkatachalam and C.A. Ahamed, "Wavelength Allocation in Dynamic Optical WDM Networks", *International Journal of Advanced Scientific and Technical Research*, vol. 6, issue 2, pp. 493-504, 2012.

[12] M. M. Salour, M. Batayneh and L. Figueroa, "Drive to miniaturization: integrated optical networks on mobile platforms", *Applied Physics A: Materials Science & Processing*, vol. 105, issue 2, pp 289-300, 2011.

[13] C. Xin, "Resource Planning for Dynamic Traffic Grooming in WDM Optical Networks", *Journal of Lightwave Technology*, vol.27, no.7, pp.817-824, 2009.

[14] G. Garg, and Er. A. Singhal,"Investigation of Various Throughput Improvement Techniques in DWDM Optical Networks", *International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE)*, vol. 2, issue 11, pp. 66-70, 2012.

[15] S. Araki, I. Nishioka, S. Ishida, Y. Iizawa and M. Nakama, "Optical network control challenges", *Summer Topical Meeting, LEOSST '09. IEEE/LEOS*, pp.139-140, 2009.

[16] A. Wang, Q. Wu, X. Zhou and J. Wang, "A New Multicast Wavelength Assignment Algorithm in Wavelength-Converted Optical Networks", *International Journal Communications, Network and System Sciences*, Vol. 2 No. 9, pp. 912-916, 2009.

[17] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xuz, Y. Zhang, X.Wen and Y. Chen, "OSA: An Optical Switching Architecture for Data Center Networks with Unprecedented Flexibility", *IEEE/ACM Transactions on Networking*, vol.PP, no.99, pp.1-14, 2013.

[18] A. Norouzi, A.H. Zaim and B. B. Ustundag, "An integrated survey in Optical Networks: Concepts, Components and Problems", *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 11, no. 1, pp. 10-26, 2011.

[19] Bo-ning Hu, Lian Hua, Rui-Mei Zhao, Hua-Wei Pang, "Computer simulation of optical fiber communication system", *International Conference on Machine Learning and Cybernetics*, vol. 5, pp. 2509-2512, 2009.

[20] Z. Zhang, and Wen-xiu Xu, "The Application of Aided Wavelet Neural Network in the Optical Fiber Communication Signal Classification", *Third International Symposium on Intelligent Information Technology Application*, vol.2, pp. 11-14, 2009.

[21] A. Kumar, "Studies on Optical Components and Radio over Fibre Systems", B. Tech thesis, National Institute of Technology, Rourkela, 2009.

[22] J. M. Senior. *Optical Fiber Communications: Principles and Practice*. London: Pearson Prentice Hall, 2009, pp. 7-10.

[23] S.A.J. Alabady, "Simulation and Evaluation of Ethernet Passive Optical Network", *Tikrit Journal of Eng. Sciences*, vol. 17, no. 3, pp. 44-58, 2010.

[24] I. Djordjevic, W. Ryan and B. Vasic, "Fundamentals of Optical Communication" in *Coding for Optical Channels*. New York: Springer Science + Business Media, 2010, pp. 25-73.

[25] R. Ramaswami, K. N. Sivarajan and G. H. Sasaki, "Components" in *Optical Networks: A Practical Perspective*. 3rd ed., Ed. Burlington: Elsevier, 2010, pp. 113-228.

[26] A. Singal, "Wavelength Assignment Algorithm in WDM Network", M.S. Thesis, University of Thapar, India, 2011.

[27] C.P. Larsen, A. Gavler, and K. Wang, "Comparison of active and passive optical access networks", *9th Conference on Telecommunications Internet and Media Techno Economics (CTTE)*, pp.1-5, 2010.

[28] A. Kumar, V. K. Banga and A. Wason, "Optimization in Optical Communication Networks: Issues and Challenges", *in Proceedings IRISET ICEMCE'2013 and ICHCES'2013*, pp. 51-56, 2013.

[29] D. Bischoff, Wavelength multiplexing: WDM and DWDM systems, pp. 1-27, 2009.

[30] M.A. Othman, M.M. Ismail, H.A. Sulaiman, M.H. Misran, and M.A.M. Said, "EDFA-WDM Optical Network Analysis", *International Journal Of Electronics And Computer Science Engineering (IJECSE),* Vol. 01, No. 4, pp.1894-1904, 2012.

[31] Kamelian. *Optical WDM Networking Sub-System Functionalities: Hybrid Solutions*, Applications Note No. 0005. Glasgow, UK: Amphotonix, 2012, pp. 1-12.

[32] Aruba Networks. *Outdoor Point-to-Point Deployment*. 2011, pp. 4-54.

[33] B. Mukherjee, "Survey of State-of-the-art" in *Optical WDM Networks*: *Principles and Practice*, 1st ed., K. M. Sivalingam and S. Subramaniam, Ed, New York: Springer-Verlag, 2010, pp. 5-24.

[34] A.N.Z. Rashed, "Optical Add Drop Multiplexer (OADM) Based on Dense Wavelength Division Multiplexing Technology in Next Generation Optical Networks", *International Journal of Multimedia and Ubiquitous Engineering*, Vol. 7, No. 1, pp. 1-14, 2012.

[35] A.N.Z. Rashed, "Transmission Performance Evaluation of Optical Add Drop Multiplexers (OADMS) In Optical Telecommunication Ring Networks", *American Journal of Engineering and Technology Research*, Vol. 12, No. 1, pp. 23-37, 2012.

[36] A. Singal and R.S. Kaler, "Performance Evaluation of Algorithms for Wavelength Assignment in Optical Ring Network", *2012 Second International Conference on Advanced Computing & Communication Technologies (ACCT)*, pp.161-166, 2012.

[37] A. Singal and R.S. Kaler, "Blocking probability of algorithms for different wavelength assignment in optical ring network", *Optik - International Journal for Light and Electron Optics*, Vol. 124, Issue 2, pp. 147-151, 2013.

[38] B. C. Chatterjee, N. Sarma and P. P. Sahu, "Review and Performance Analysis on Routing and Wavelength Assignment Approaches for Optical Networks", *IETE Technical Review*, vol. 30, issue 1, pp. 12-23, 2013.

[39] B. Rawat, A. K. Gupta and V. Yadav, "Review and Performance Analysis on Routing and Wavelength Assignment Approaches for Optical Networks", *International Journal of Scientific & Engineering Research*, Volume 4, Issue 7, pp. 247-352, 2013.

[40] U. Bhanja, S. Mahapatra and R. Roy, "A Novel Solution to the Dynamic Routing and Wavelength Assignment Problem in Transparent Optical Networks", *International Journal of Computer Networks & Communication*, vol. 2, no. 2, pp. 119-130, 2010.

[41] G. Marković, V. Aćimović-Raspopović and V. Radojičić, "A heuristic algorithm for lightpath scheduling in next-generation WDM optical networks", *Photonic Network Communications*, Vol. 23, Issue 3, pp. 272–284, 2012.

[42] J.Y. Zhang, H. Mouftah, J. Wu, and M. Savoie, "Lightpath Scheduling and Routing for Traffic Adaptation in WDM Networks", *Journal of Optical Communications and Networking*, Vol. 2, No. 10, pp. 803-819, 2010.

[43] A. Sangeetha, K. Anusudha, Shobhit Mathur and Manoj Kumar Chaluvadi, "Wavelength Assignment Problem in Optical WDM Networks", *International Journal of Recent Trends in Engineering*, vol. 1, no. 3, pp. 201-205, 2009.

[44] V. Srivastava and P. Srivastava, "Routing in Optical Networks', *VSRD Technical & non-Technical Journal*, Vol. I (3), pp. 153-158, 2010.

[45] A. Wason and R.S. Kaler, "Generic-II routing and wavelength assignment algorithm for a wavelength-routed WDM network", *Optik - International Journal for Light and Electron Optics*, Volume 122, Issue 12, pp. 1107-1112, 2011.

[46] T.A. Hahn, "Investigation of Physically Aware Routing and Wavelength Assignment (RWA) Algorithms for Next Generation Transparent Optical Networks", PhD Thesis, Montana State University, Montana, 2010.

[47] M. Arunachalam, C.R.D.Ajay Kumar and J.S.Arunbabu, "Practical Perspective on Wavelength Assignment Strategies in Optical Networks: A Survey", *International Journal of Scientific & Engineering Research*, Volume 3, Issue 9, pp. 1-5, 2012.

[48] Han-You Jeong, Seung-Woo Seo, and Yoon-Ho Choi, "Minimizing wavelength conversion cost in WDM networks with a constrained blocking probability", *15th International Conference on Optical Network Design and Modeling (ONDM)*, pp.1-6, 2011.

[49] M. Arunachalam and V. Rajamani, "Multilevel Feedback Queue Wavelength Assignment Algorithm in Survivable Optical WDM Networks", *ICTACT Journal on Communication Technology*, Volume 2, Issue 4, pp. 433-437, 2011.

[50] N. Nag and I. Bala, "Comparative Study of Wavelength Assignment Algorithms for WDM Optical Network", *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, issue 10, pp. 1247-1249, 2013.

[51] G. Wedzinga. *Photonic Slot Routing in Optical Transport Networks*. Norwell: Kluwer Academic Publisher, 2003, pp. 55-82.

[52] Y. Wu, L. Chiaraviglio, M. Mellia and F. Neri, "Power-Aware Routing and Wavelength Assignment in optical networks", *35th European Conference on Optical Communication (ECOC '09)*, pp.1-2, 2009.

[53] D.S. Malik. *Data Structure using C++*. Canada: Cengage Learning, 2009, pp. 533-598.

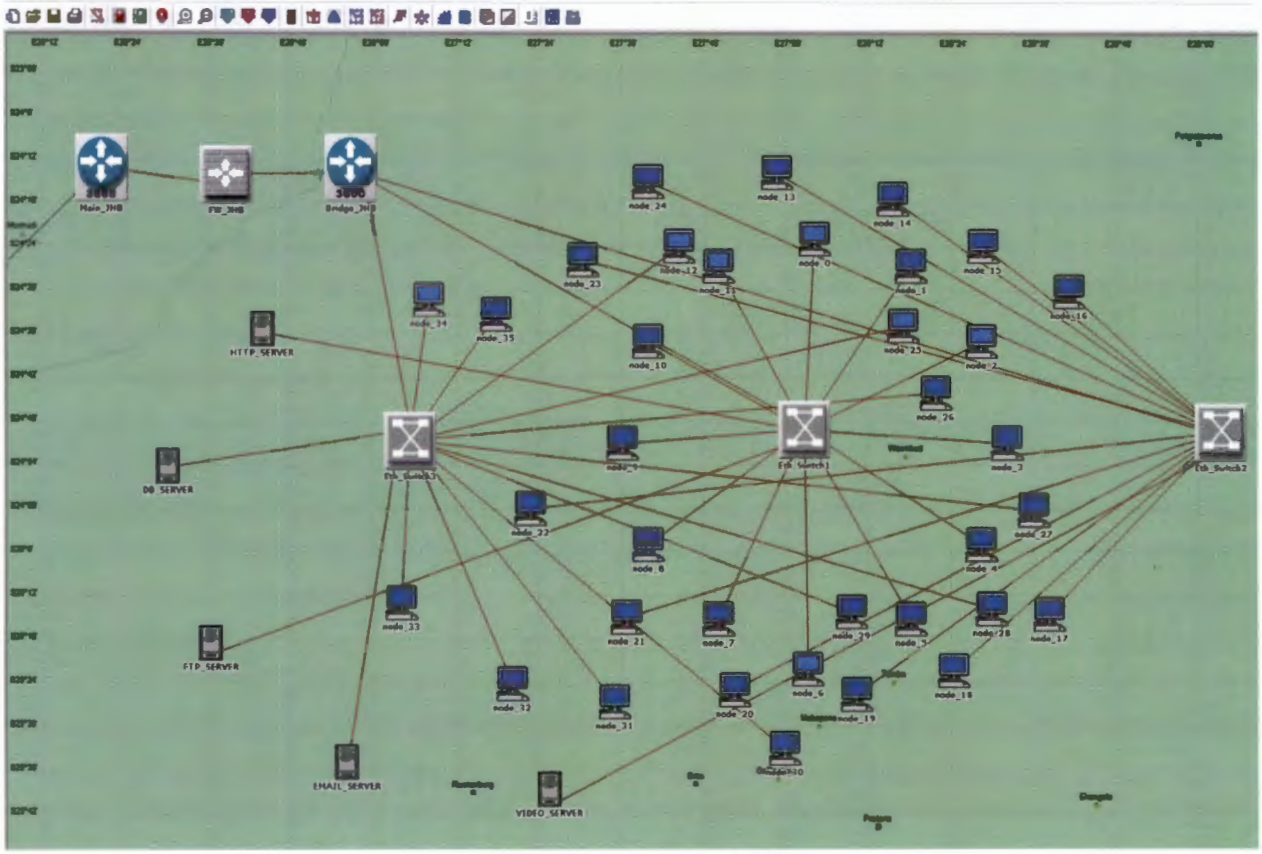
[54] Z. Lu and H. Yang. *Unlocking the Power of OPNET Modeler.* New York, USA: Cambridge University Press, 2012.

[55] S. Siraj, A.J. Gupta, and Rinku-Badgujar, "Network Simulation Tools Survey", *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 1, Issue 4, pp. 201-210, 2012.

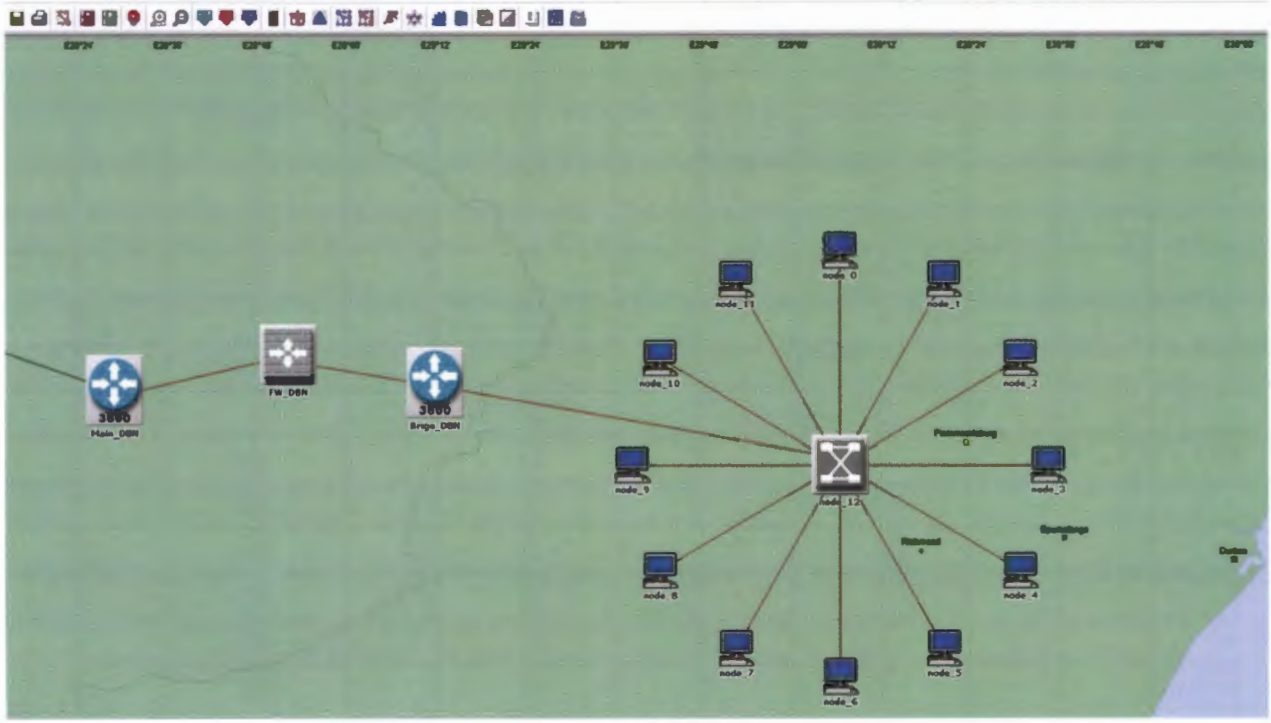
[56] M.N. Islam, "Simulation-Based Comparative Study of EIGRP and OSPF for Real-Time Applications", Msc Thesis, Blekinge Institute of Technology, Sweden, 2012.

[57] M. Mbougni, "Design, Implementation and Simulation of Wireless Mesh Networks Routing Protocols with Link Quality and Interference Awareness", Msc Thesis, North West University, South Africa, 2012.

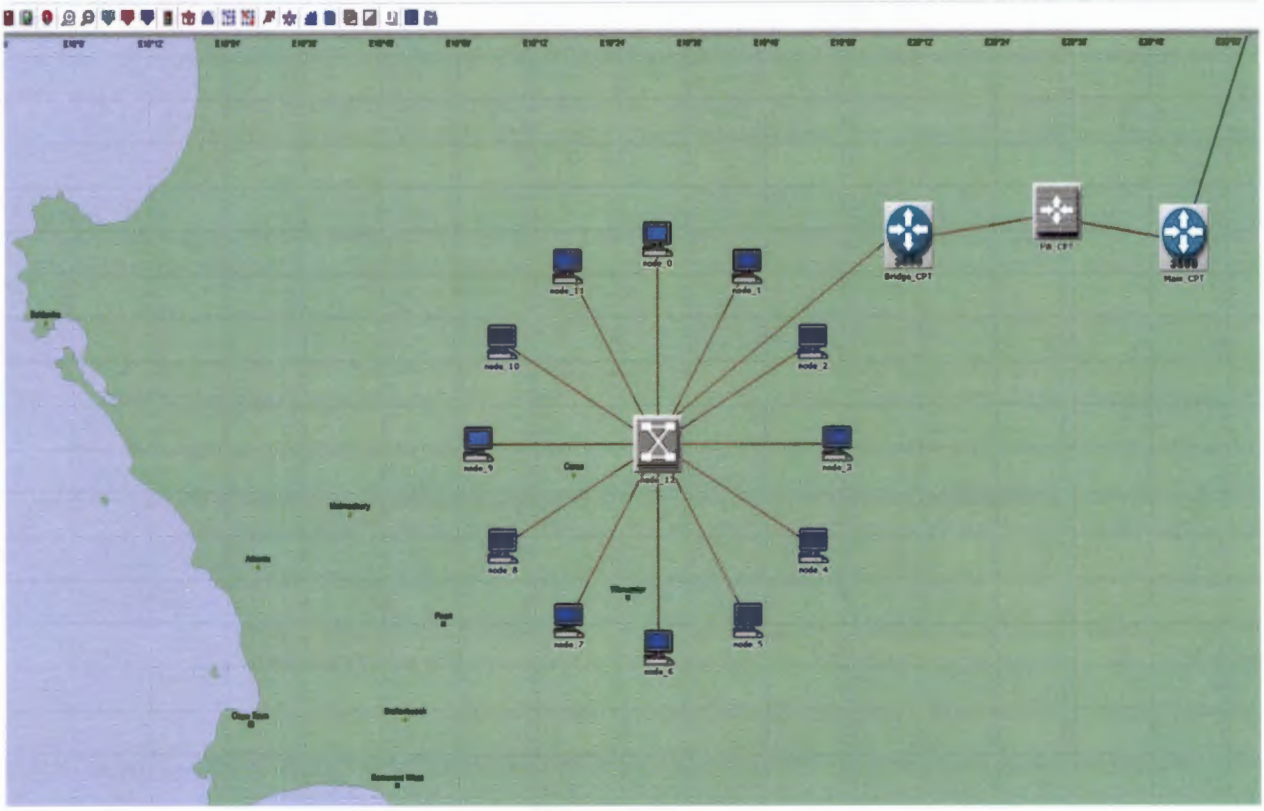
Appendix A: Simulation Setup



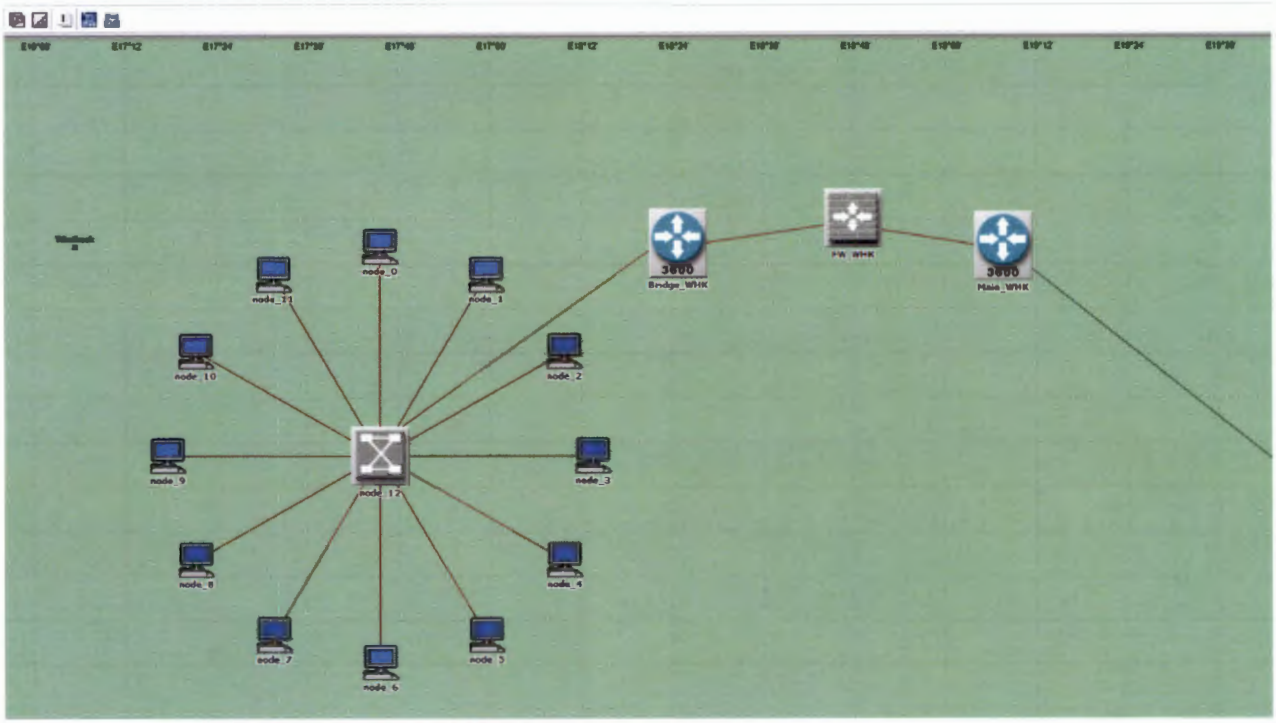
Appendix A1: JHB Subnet- Network Model



Appendix A2: DBN Subnet- Network model



Appendix A3: CPT Subnet- Network model



Appendix A4: WHK Subnet- Network model

Appendix B: C Source Code

```
/* Includes */
#include "oms_pr.h"
#include "oms_tan.h"
#include "oms_bgutil.h"
#include "ethernet_support.h"
#include "ip_addr_v4.h"
#include "ip_rte_v4.h"
#include "ip_dgram_sup.h"
#include "oms_protocol.h"
#include "oms_pipeline.h"
#include "oms_auto_addr_support.h"
#include "oms_pkt_analyzer.h"
#include "oms_vlan_support.h"
#include "ip_support.h"
#include "bridge_header.h"
#include "nato.h"
#include "oms_data_def.h"
#include "oms_rr.h"

#include <hsrp.h>

/* incoming statistics wires */
#define TRANSMITTING_INSTAT                    0
#define  RECEIVING_INSTAT                      1

#define BRIDGE_BROADCAST_ADDR    -2

/* outgoing statistics wires */
#define FRAME_WAITING_OUTSTAT                  0
```

```
#define TAG_IS_NOT_SET                  -5

/*** Ethernet constants ***/
/* maximum number of transmission attempts for a given frame        */
#define ATTEMPT_LIMIT                   16

/* truncation point for exponential backoff algorithm               */
#define BACKOFF_LIMIT                   10

/* The minimum amount of time that has to elapse between two        */
/* frame transmissions - default minimum IFG is 96 BT.              */
#define  INTERFRAME_GAP                 (96 / ethernet_state_info_ptr->bit_rate)

/* Length of time a station can transmit in         */
/* burst mode (seconds)    - default is 65536 BT    */
#define  BURST_LENGTH                   (65536 / ethernet_state_info_ptr->bit_rate)

/* constants used for controlling the frame size */
#define MIN_DATA_SIZE                   368
#define MAX_DATA_SIZE                   12000
#define MIN_FRAME_SIZE                  512
#define PREAMBLE_SIZE                   64
#define GIGABIT_MIN_FRAME_SIZE          4096

/* Length of segment (in bits) that is sent after collision         */
/* detection.
                */
#define JAM_SIZE                        32

/* Wild-carding ethernet address which represents any               */
/* destination.
                */
#define ETH_MAC_BROADCAST_ADDR          -1
```

```
/* Special value that indicates unspecified address value.          */
#define  ETH_MAC_UNSPECIFIED_ADDR        -99

/* Define a small value (= 1 psec), which will be used to           */
/* recover from double arithmetic precision losts while doing  */
/* time related    precision sensitive computations.                */
#define          PRECISION_RECOVERY                    0.000000000001

/*** Transmit completion status codes ***/
#define STATUS_ERR_LATE_COLL      0
#define STATUS_ERR_EXCESS_COLL          1
#define STATUS_OK                              2
#define STATUS_RETRANSMIT_PENDING       3

/* MACROS */
/* All Addresses that are within the range from */
/* 01 00 5E 00 00 00 to 01 00 5E 7F FF FF*, or      */
/* from 33 33 00 00 00 00 to 33 33 FF FF FF FF      */
/* are multicast addresses for Ethernet.            */
#define ethernet_mac_addr_is_multicast_addr(_mac_addr) ((((_mac_addr >> 23) ^ 131260) == 0 || (_mac_addr
>> 32) == 0x3333)?OPC_TRUE:OPC_FALSE)

/*** Global variables ***/

/* Static flag for checking if there is any routed                  */
/* background utilization in this simulation.                       */
static Boolean do_bgutil;

/*** Data structure definitions ***/

/* Enumerate the traffic type (Higher Layer or physical layer)  */
typedef enum Eth_Mac_Traffic_Type
```

```
        {
        EthC_Traffic_Throughput,
        EthC_Traffic_Load
        } EthC_Mac_Traffic_Type;


/* Define interrupt codes for generating handling interrupts     */
/* indicating changes in deference and collision status (remote */
/* interrupts from the hub) and backoff and jam periods (self   */
/* interrupts).
                */
typedef enum EthT_Mac_Intrpt_Code
        {
        EthC_Deference_Off,
        EthC_Collision_On,
        EthC_Jam_Over,
        EthC_Backoff_Over,
        EthC_Intframe_Gap_Off,
        EthC_Tx_End,
        EthC_Txn_Status_Notify
        } EthT_Mac_Intrpt_Code;


/* Define the codes related to MAC's operational mode and hub        */
/* connectivity.
        */
typedef enum EthT_Mac_Operational_Mode
        {
        EthC_Mac_Half_Duplex = 0,
        EthC_Mac_Full_Duplex,
        EthC_Mac_Connected_To_Hub,
        EthC_Mac_Not_Connected_To_Hub,
        EthC_Mac_Not_Connected
        } EthT_Mac_Operational_Mode;
```

```
/* EthC_Frame_Bursting_Not_Used signifies this     */
/* MAC does not implement Frame Bursting.  This */
/* means this MAC is operating at 10Mbps          */
/* (Ethernet) or 100Mbps (Fast Ethernet), or if   */
/* this MAC is operating at 1000Mbps (Gigabit     */
/* Ethernet), the attribute "Frame Bursting" is   */
/* set to "Disabled."                                                  */
/* EthC_Frame_Bursting_Used signifies this MAC    */
/* is operating at 1000Mbps, and that the attri-*/
/* bute "Frame Bursting" is set to "Enabled."*/
typedef enum EthT_Frame_Bursting_Mode
        {
        EthC_Frame_Bursting_Not_Used,
        EthC_Frame_Bursting_Used
        } EthT_Frame_Bursting_Mode;

/* EthC_Frame_Bursting_Off signifies this  */
/* MAC is not in the middle of a burst.           */
/* EthC_Frame_Bursting_On signifies this   */
/* MAC is in the middle of a burst.               */
typedef enum EthT_Frame_Bursting_Status
        {
        EthC_Frame_Bursting_Off,
        EthC_Frame_Bursting_On
        } EthT_Frame_Bursting_Status;


typedef enum EthT_Operational_Speed
        {
        EthC_10BaseT_Ethernet,
        EthC_Fast_Ethernet,
        EthC_Gigabit_Ethernet,
```

```
	EthC_10Gigabit_Ethernet
	} EthT_Operational_Speed;

/* Structure to store process model variables that need		*/
/* to be used only when this interface is connected.	*/
typedef struct EthT_State_Info
	{
	double		last_time_channel_became_free;	/* Last time at which the channel	*/
							/*
transitioined to	becoming free.		*/

	int		attempts;		/* Varaibles maintaining interrupt types, codes
		*/
							/*
and status of transmission of packets		*/

	int		tx_comp_status;
	int		tx_channel_objid;
	Ici*		llc_iciptr;
	int		promis;
	int		max_backoff;
	int		reentry;
	double		frag_time;
	Packet*		current_frame;
	double		frame_start_time;
	double		bit_rate;
	double		slot_time;
	double		burst_end_time;
	double		burst_start_time;
	EthT_Frame_Bursting_Mode	frame_burst_usage;	/* frame_burst_usage tells if this MAC
implements*/
							/*
frame bursting. If this MAC is using regular or	*/
							/*
fast ethernet, frame bursting will not be used.	*/
```

```
                                                                    /* If
this MAC is using gigabit ethernet, frame          */
                                                                    /*
bursting may be used, but is optional.             */

        EthT_Mac_Operational_Mode       connected_to_hub;
        double                  next_transmission; /* Time when the next transmission can take
        */
                                                                    /*
place (used only in full duplex operation).        */

        EthT_Operational_Speed          operational_speed;/* Rate (bits/sec) of operation of the MAC.
        */

        int                                             strm_to_ipx;
        int                                             strm_from_ipx;
        int                     strm_to_higher_layer;
        int                     strm_from_higher_layer;
        int                                             strm_to_lower_layer;
        int                                             strm_from_lower_layer;
        Evhandle                        end_of_gap_ev_handle;/* Handle to the self interrupt that will
                */
                                                                    /*
indicate the end of current interframe gap. Used   */
                                                                    /*
only when operating in half duplex mode over a     */
                                                                    /*
direct MAC to MAC connection.                      */

        double                  end_of_gap_time;/* Ending time of the current or last interframe gap.*/
                                                                    /*
Updated only when operating in half duplex mode    */
                                                                    /*
over a direct MAC to MAC connection.               */

        Boolean                 mac_port_of_a_gateway_node;/* Flag to indicate whether the current   */
                                                                    /*
MAC belongs to a gateway.                          */
```

```
        Sbhandle                        reassembly_buffer;/* Reassembly buffer for packet segments that
        */
                                                                                                    /*
were segmented by the multi-protocol switch     */

        OmsT_Pkt_Analyzer_Info*                 pkt_capture_info_ptr;/* Information used for sniffing
network data. */

        Boolean                 dropped_large_pkt_log_written;/* Flag used to prevent dropped large   */
                                                                                                    /*
packet log messages being written multiple times.*/

        Objid                                   neighbor_mac_id;
        Objid                   mac_comp_objid;         /* A compound variable to store all ethernet mac
        */
                                                                                                    /*
parameters pertaining to one ethernet interface         */

        Objid                   mac_attr_objid;/* The objid that corresponds to ethernet mac parameters */

        Boolean                                 mac_port_of_a_bridge_switch_node;

        /* Statistics collection variables      */
        double                  load_last_stat_update_time;
        double                  pk_thru_last_stat_update_time;
        double                                  eth_mac_global_ete_delay; /* End to end global
delay                           */
        double                                  packets_bursted;/* packets_bursted keeps track
of the number of        */
                                                                                                    /*
packets sent in a single burst.                 */

        OmsT_Bgutil_Routed_State*       load_bgutil_routed_state_ptr;
        OmsT_Bgutil_Routed_State*       thruput_bgutil_routed_state_ptr;

        /* Stat handles for statistics          */
```

```
        Stathandle      packet_load_handle;
        Stathandle      bit_load_handle;
        Stathandle      packet_sec_load_handle;
        Stathandle      bit_sec_load_handle;
        Stathandle      collision_num_handle;
        Stathandle      packet_thru_handle;
        Stathandle      bit_thru_handle;
        Stathandle      packet_sec_thru_handle;
        Stathandle      bit_sec_thru_handle;
        Stathandle      ete_handle;
        Stathandle      global_ete_handle;
        Stathandle      retrans_handle;


        Stathandle      frame_bursting_stathandle;      /*
Frame_bursting stathandles record  */
                                                        /*
when a MAC begins or ends a burst.                      */

        Stathandle      frame_burst_duration_stathandle; /* Frame_burst
        duration records  */
                                                        /*
how long a station was in a burst.                      */
        Stathandle      packets_per_burst_stathandle;   /*
Packets_per_burst records how       */
                                                        /*
many packets were sent in a single burst.*/
        } EthT_State_Info;


/*** Mnemonic macros for transitions and executives ***/

/* The value of the state variable hub_busy is set by the          */
/* connected hub to 0 or 1 based on its status.        Hub sets    */
/* this variable to 0 when it is initializing itself. The          */
```

```
/* variable is initialized to -1 by the MAC. Consequently, if    */
/* it remains as -1 then this indicates that the mac             */
/* is not connected to a hub. Based on the value of this         */
/* the "connected_to_hub" variable is updated to indicate        */
/* whether this MAC is connected to a HUB node.                  */
#define  NOT_CONNECTED_TO_HUB    (ethernet_state_info_ptr->connected_to_hub ==
EthC_Mac_Not_Connected_To_Hub)

#define DEFERENCE_ON          (eth_hdx_deference () == OPC_TRUE)

#define DEFERENCE_OFF         (eth_hdx_deference () == OPC_FALSE)

#define DEFERENCE_LOW         (eth_hdx_def_low () == OPC_TRUE)

/* A collision is detected ONLY when operating in half duplex    */
/* mode. If the MAC is connected to a hub then hub reports the   */
/* collision with a remote interrupt. In case of no hub, if the  */
/* MAC receiver starts receiving a frame when its transmitter    */
/* is transmitting a frame, then MAC detects the collision by    */
/* itself.
                */
#define COLL_DET_HIGH         (eth_hdx_coll ())

#define XMIT_LOW                    (intrpt_type == OPC_INTRPT_STAT &&   \
                                        op_intrpt_stat () == TRANSMITTING_INSTAT
&& \
                                        op_stat_local_read
(TRANSMITTING_INSTAT) == 0.0)

#define FRAME_ARRIVAL               (intrpt_type == OPC_INTRPT_STRM &&   \
                                        ((intrpt_strm == ethernet_state_info_ptr-
>strm_from_ipx) || \
                                        (intrpt_strm == ethernet_state_info_ptr-
>strm_from_higher_layer)))
```

```
#define EXCESS_COL                  (ethernet_state_info_ptr->tx_comp_status ==
STATUS_ERR_EXCESS_COLL)

#define LATE_COL                    (ethernet_state_info_ptr->tx_comp_status ==
STATUS_ERR_LATE_COLL)

#define JAM_END                         (intrpt_type == OPC_INTRPT_SELF &&          \
                                            intrpt_code == EthC_Jam_Over)

#define BACKOFF_END                 (intrpt_type == OPC_INTRPT_SELF &&          \
                                            intrpt_code == EthC_Backoff_Over)

#define QUEUE_EMPTY                 (op_q_empty () == OPC_TRUE)

#define SET_ENTRY                   (ethernet_state_info_ptr->reentry = 0)
#define SET_REENTRY                 (ethernet_state_info_ptr->reentry = 1)

#define  FULL_DUPLEX          (mac_op_mode == EthC_Mac_Full_Duplex)
#define UNCONNECTED                 (mac_op_mode == EthC_Mac_Not_Connected)

/* Test for the value of burst mode.        */
#define  BURST_MODE_ON              (frame_burst_status == EthC_Frame_Bursting_On)
#define  BURST_MODE_OFF             (frame_burst_status == EthC_Frame_Bursting_Off)

/* Check for ODB trace information display. */
#define  ETHERNET_TRACE_ACTIVE   (op_prg_odb_ltrace_active ("ethernet") == OPC_TRUE)

/*** Function prototypes ***/

static void                             ethernet_mac_stat_init ();
static void                             ethernet_mac_sv_init ();

static void                             ethernet_mac_phys_pk_accept ();
```

```
static void                     ethernet_mac_llc_pk_accept ();

static void                     ethernet_mac_error (const char *msg0, const char *msg1,
const char *msg2);

static void                     ethernet_mac_warn (const char *msg0, const char *msg1,
const char *msg2);

static void                     ethernet_mac_stat_intrpt_disable ();


static int                  eth_hdx_deference ();

static int                  eth_hub_def_on ();

static int                  eth_no_hub_def_on ();


static int                  eth_hdx_def_low ();

static int                  eth_hub_def_low ();

static int                  eth_no_hub_def_low ();


static int                  eth_hdx_coll ();

static int                  eth_hub_coll ();

static int                  eth_no_hub_coll ();


static void                     eth_mac_fdx_pkt_send (Packet* pkptr);


static void                     eth_interframe_gap_schedule ();

static void                     eth_interrupts_process (int set_deference_timer);

static void                 ethernet_mac_pk_stats_update (Packet* pkptr, double pk_size,
                                    double * last_stat_update_time_ptr,
OmsT_Bgutil_Routed_State**
                                    bgutil_routed_state_pptr, Stathandle *
packet_shandle_ptr,
                                    Stathandle * packet_sec_shandle_ptr, Stathandle
* bit_shandle_ptr,
                                    Stathandle * bit_sec_shandle_ptr,
EthC_Mac_Traffic_Type traffic_type,
                                    Boolean do_bgutil_flag);


static void                 ethernet_mac_lower_layer_streams_determine (void);
```

```
static void                     eth_mac_portno_obtain (char* portno_str);

static Boolean                  eth_mac_virtual_address_check (HsrpT_Mac_Address
dest_mac_addr);

EXTERN_C_BEGIN

static void                     ethernet_mac_pk_stats_update_endsim (void * dummy_state, int
dummy_code);

EXTERN_C_END

static void
ethernet_mac_sv_init ()
        {
        char                    proc_model_name [64];
        Boolean                 mode;
        Objid                   tx_comp_attr_objid;
        Objid                   link_objid;
        static Cmohandle eth_state_info_cmh = OPC_NIL;

        /**     Initializes state variables.                                    **/
        FIN (ethernet_mac_sv_init ());

        /*      Obtain the module's object identifier.                          */
        my_objid = op_id_self ();

        /* Determine object identifier of transmitter connecting the     */
        /* mac layer to the hub or neighbor MAC.  This id is needed to*/
        /* abort collided transmissions in the "COLLISION" state.         */
        tx_objid = op_topo_assoc (my_objid, OPC_TOPO_ASSOC_OUT, OPC_OBJTYPE_PTTX, 0);
        if (tx_objid == OPC_OBJID_INVALID)
                ethernet_mac_error ("Unable to get Objid of transmitter module.",
```

```
                "Node model is probably incorrect; make sure that",
                "ethernet_mac connects to a point to point transmitter through output stream 0.");

/* Determine the link associated with the current transmitter    */
if (op_topo_assoc_count (tx_objid, OPC_TOPO_ASSOC_OUT, OPC_OBJTYPE_LKDUP) == 0)
        {
        /* Return as there is no transceiver or links connected to this MAC.      */
        mac_op_mode = EthC_Mac_Not_Connected;
        }
else
        {
        /* Obtain the node's object identifier.
*/
        own_node_objid = op_topo_parent (my_objid);

        /* Obtain the subnet object identifier.
*/
        subnet_objid = op_topo_parent (own_node_objid);

        /* Obtain the process's process handle.
*/
        own_prohandle = op_pro_self ();

        /* Obtain the name of the process.
*/
        op_ima_obj_attr_get (my_objid, "process model", proc_model_name);

        /* Initialize own group address. Group MAC addresses represent        */
        /* link aggregation groups and serve as a single MAC address */
        /* for all the MACs in the group. If this MAC becomes a port of       */
        /* a link aggregation group then this state variable will be      */
        /* populated by the link aggregation process.                                 */
        my_group_address = ETH_MAC_UNSPECIFIED_ADDR;
```

```
/* Initialize the HSRP virtual address info pointer.                    */
hsrp_info_ptr = OPC_NIL;

/* Register this ethernet MAC process in the model wide                 */
/* registry.
                    */
own_process_record_handle = (OmsT_Pr_Handle) oms_pr_process_register
(own_node_objid, my_objid, own_prohandle, proc_model_name);
oms_pr_attr_set (own_process_record_handle,
        "protocol",         OMSC_PR_STRING,   "mac",
        "mac_type",         OMSC_PR_STRING,   "eth_hub",
        "module objid", OMSC_PR_OBJID,  my_objid,
        OPC_NIL);

/* This MAC is connected to a link. Proceed further with initialization. */
link_objid = op_topo_assoc (tx_objid, OPC_TOPO_ASSOC_OUT,
OPC_OBJTYPE_LKDUP, 0);

/* Allocate memory to the ethernet state info pointer, using categorized */
/* memory. But first define the category if it hasn't already been defined.*/
if (eth_state_info_cmh == OPC_NIL)
        {
        eth_state_info_cmh = prg_cmo_define ("Ethernet State Info");
        }

ethernet_state_info_ptr = (EthT_State_Info *) prg_cmo_alloc (eth_state_info_cmh, sizeof
(EthT_State_Info));

if (ethernet_state_info_ptr == OPC_NIL)
        {
        op_sim_end ("Error in Ethernet support code:",
                "Unable to allocate memory for Ethernet State Info.",
                OPC_NIL, OPC_NIL);
        }
```

```
		/* Get the compound attribute for the transmitter's channel	*/
		op_ima_obj_attr_get (tx_objid, "channel", &tx_comp_attr_objid);
		if (tx_comp_attr_objid == OPC_OBJID_INVALID)
			ethernet_mac_error("Unable to get Objid of transmitter compound
attribute",OPC_NIL,OPC_NIL);

		/* Obtain the address handle assigned to this node. The string	*/
		/* "MAC Addresses" rendezvous with other MACs to guarantee		*/
		/* unique addresses across all MAC types. The OMS_AA package	*/
		/* handles this feature.
		*/
		oms_aa_handle = oms_aa_address_handle_get ("MAC Addresses", "Address");

		/* Obtain the objid of ethernet mac parameters attribute */
		op_ima_obj_attr_get (my_objid, "MAC Parameters", &ethernet_state_info_ptr-
>mac_comp_objid);
		ethernet_state_info_ptr->mac_attr_objid = op_topo_child(ethernet_state_info_ptr-
>mac_comp_objid, OPC_OBJTYPE_GENERIC, 0);

		/* Determine whether the promiscuous mode is enabled.		*/
		op_ima_obj_attr_get (ethernet_state_info_ptr->mac_attr_objid, "Promiscuous Mode",
&ethernet_state_info_ptr->promis);

		/* Initialize the reassembly buffer -- to be used when this MAC	*/
		/* frame segments (rather than complete packets). This is a	*/
		/* typical condition in a multiprotocol switched environment	*/
		/* (e.g., FDDI frame segments coming to Ethernet layer).		*/
		ethernet_state_info_ptr->reassembly_buffer = op_sar_buf_create
(OPC_SAR_BUF_TYPE_REASSEMBLY,
			OPC_SAR_BUF_OPT_DEFAULT);

		/* Determine whether the operational mode of the mac is half	*/
		/* duplex or full duplex.
		*/
```

```
		op_ima_obj_attr_get (ethernet_state_info_ptr->mac_attr_objid, "Operational Mode", &mode);

		mac_op_mode = (mode == OPC_TRUE) ? EthC_Mac_Full_Duplex :
EthC_Mac_Half_Duplex;


		/* Determine the object ID of channel 0. This will be also used */
		/* to abort the transmissions when a collision is detected	*/
		/* while operating in half duplex mode.
	*/
		ethernet_state_info_ptr->tx_channel_objid = op_topo_child (tx_comp_attr_objid,
OPC_OBJTYPE_PTTXCH, 0);


		if (ethernet_state_info_ptr->tx_channel_objid == OPC_OBJID_INVALID)
			ethernet_mac_error ("Unable to get Objid of transmitter channel", OPC_NIL,
OPC_NIL);


		/* Determine the bitrate used by the transmitter's channel	*/
		if (op_ima_obj_attr_get (ethernet_state_info_ptr->tx_channel_objid, "data rate",
&ethernet_state_info_ptr->bit_rate) == OPC_COMPCODE_FAILURE)
			ethernet_mac_error ("Unable to get data rate from transmitter channel", OPC_NIL,
OPC_NIL);


		/* The slot_time and frame_bursting feature usage is determined	*/
		/* using the data rate at which this MAC is operating. Obtain	*/
		/* this data rate if MAC is connected.
	*/
		if (op_topo_assoc_count (tx_objid, OPC_TOPO_ASSOC_OUT, OPC_OBJTYPE_PTRX) !=
0)
			{
			if (ethernet_state_info_ptr->bit_rate == 10000000000.0)
				{
				/* This model is using 10 gigabit ethernet.(IEEE 802.3ae)	*/
				/* Since 10Gbps only operates in full-duplex model, there	*/
				/* is no need for protocol add-ons like carrier extension	*/
				/* or frame bursting (like in half-duplex 802.3z).		*/

				/* Slot time is the quantum for backoff interval computa-	*/
```

```
                    /* tion. It is a function of the data transmission rate,   */
                    /* which is        obtained directly from the attr. list of the      */
                    /* transmitter.
                            */
                    ethernet_state_info_ptr->slot_time = 512 / ethernet_state_info_ptr-
>bit_rate;
                    ethernet_state_info_ptr->operational_speed = EthC_10Gigabit_Ethernet;

                    /* Check the mode in which this MAC operates. If it is set      */
                    /* to half-duplex, then set it to full-duplex. Also, note */
                    /* that this MAC cannot be connected to a hub.
        */
                    if (mac_op_mode == EthC_Mac_Half_Duplex)
                            {
                            /* Change the operational mode to full-duplex.
        */
                            mac_op_mode = EthC_Mac_Full_Duplex;

                            /* Also, change it on the actual attribute setting.
        */
                            op_ima_obj_attr_set (ethernet_state_info_ptr->mac_attr_objid,
"Operational Mode", mac_op_mode);

                            /* Generate a log message to indicate this change.     */
                            ethernet_10Gbps_force_fdx_mode_log_write ();
                            }
                    }
            else if (ethernet_state_info_ptr->bit_rate == 1000000000.0)
                    {
                    /* This model is using gigabit ethernet.(IEEE 802.3z) In       */
                    /* this case, the slot time is 4096 bit times, and the         */
                    /* value of the model attribute frame_bursting is read to      */
                    /* determine if this gigabit ethernet model implements                 */
                    /* Frame Bursting.
                            */
```

```
                        /* Slot time is the quantum for backoff interval computa-      */
                        /* tion. It is a function of the data transmission rate,   */
                        /* which is        obtained directly from the attr. list of the    */
                        /* transmitter
                                */
                        ethernet_state_info_ptr->slot_time = 4096 / ethernet_state_info_ptr-
>bit_rate;
                        ethernet_state_info_ptr->operational_speed = EthC_Gigabit_Ethernet;
                        }
                    else if (ethernet_state_info_ptr->bit_rate == 100000000.0)
                        {
                        /* This model is using ethernet or fast ethernet.  In this         */
                        /* case, the slot time is 512 bit times, and frame bursting        */
                        /* is not used (Frame Bursting is only used in gigabit             */
                        /* ethernet)
                                */
                        ethernet_state_info_ptr->slot_time = 512 / ethernet_state_info_ptr-
>bit_rate;
                        ethernet_state_info_ptr->frame_burst_usage =
EthC_Frame_Bursting_Not_Used;
                        ethernet_state_info_ptr->operational_speed = EthC_Fast_Ethernet;
                        }
                    else if (ethernet_state_info_ptr->bit_rate == 10000000.0)
                        {
                        /* This model is using ethernet .  In this case, the slot */
                        /* time is 512 bit times, and frame bursting is not used           */
                        /* (Frame Bursting is only used in gigabit ethernet)               */
                        ethernet_state_info_ptr->slot_time = 512 / ethernet_state_info_ptr-
>bit_rate;
                        ethernet_state_info_ptr->frame_burst_usage =
EthC_Frame_Bursting_Not_Used;
                        ethernet_state_info_ptr->operational_speed = EthC_10BaseT_Ethernet;
                        }
                    else
                        {
                        /* This model only supports regular, fast, gigabit and 10          */
```

```
                    /* gigabit ethernet data rates. The link data rate is set  */
                    /* to a value different than any of these. Report this     */
                    /* configuration error with a simulation message and       */
                    /* notification log, and set the data rate of the          */
                    /* transmitter to 100 Mbps.
                    */
                    op_sim_message ("ERROR reported by Ethernet MAC model
(ethernet_mac_v2): Detected a non-standard",
                                    "data rate on the connected Ethernet
link. Using the data rate of 100 Mbps instead.");
                    ethernet_invalid_link_rate_log_write ();

                    /* Use 100 Mbps as the transmission rate.
        */
                    ethernet_state_info_ptr->bit_rate = 100000000.0;
                    ethernet_state_info_ptr->slot_time = 512 / ethernet_state_info_ptr-
>bit_rate;
                    ethernet_state_info_ptr->frame_burst_usage =
EthC_Frame_Bursting_Not_Used;
                    ethernet_state_info_ptr->operational_speed = EthC_Fast_Ethernet;

                    /* Update the transmitter's data rate.
                */
                    op_ima_obj_attr_set (ethernet_state_info_ptr->tx_channel_objid, "data
rate", ethernet_state_info_ptr->bit_rate);
                    }
                }

            /* Bursting should initially be off.  It will be set to on     */
            /* when it has more than one packet to send, and the its burst */
            /* timer has not expired.
            */
            frame_burst_status = EthC_Frame_Bursting_Off;

            /* Initialize the variable used to contain information whether */
            /* the surrounding node is a bridge/switch node. This is used  */
            /* to avoid  (1) writing statistics, if it is part of a bridge */
```

```
/* node, and (2) not perform encapsulation/decapsulation.                */
ethernet_state_info_ptr->mac_port_of_a_bridge_switch_node = OPC_FALSE;

/* Initialize burst_end_time to zero.  This value will be set      */
/* to a different value when and if this MAC starts bursting.      */
ethernet_state_info_ptr->burst_end_time = 0.0;

/* Create an interface control information (ICI) structure                */
/* for communication of parameters with the higher layer (LLC)            */
ethernet_state_info_ptr->llc_iciptr = op_ici_create ("eth_mac_ind");
if (ethernet_state_info_ptr->llc_iciptr == OPC_NIL)
        ethernet_mac_error ("Unable to create ICI for communication with LLC.",
OPC_NIL, OPC_NIL);

/* The last time the channel became free is initialized to a negative number larger than     */
/* the interframe gap, since we want the channel to appear to have been free for some time
*/
/* when the simulation begins.
                                                                        */
ethernet_state_info_ptr->last_time_channel_became_free = -2.0 * INTERFRAME_GAP;

/* Also initialize the end of the last interframe gap to a negative number. */
ethernet_state_info_ptr->end_of_gap_time = -1.0;

/* Set up state variables used in tracking background utilization. */

ethernet_state_info_ptr->thruput_bgutil_routed_state_ptr = OPC_NIL;
ethernet_state_info_ptr->load_bgutil_routed_state_ptr = OPC_NIL;
if (oms_basetraf_set_bgutil_for_demands())
        {
        do_bgutil = OPC_TRUE;
        }

/* Initialize the last_stat_update_time to zero, which is used     */
```

```
		/* for updating background utilization statistics.				*/
		ethernet_state_info_ptr->load_last_stat_update_time = 0.0;
		ethernet_state_info_ptr->pk_thru_last_stat_update_time = 0.0;

		/* Initialize the flag that controls the number of dropped		*/
		/* packet log messages.
		*/
		ethernet_state_info_ptr->dropped_large_pkt_log_written = OPC_FALSE;

		/* Schedule a procedure which will be called for this process at the	*/
		/* end of simulation.
		*/
		op_intrpt_schedule_call (OPC_INTRPT_SCHED_CALL_ENDSIM, 0,
			ethernet_mac_pk_stats_update_endsim, OPC_NIL);
		}

	FOUT;
	}


static void
ethernet_mac_stat_init ()
	{
	/** Registers statistics and initilaizes the variables		**/
	/**	associated with them.
		**/
	FIN (ethernet_mac_stat_init ());

	/*	Register load-based statistic handles.				*/
	ethernet_state_info_ptr->packet_load_handle    = op_stat_reg ("Ethernet.Load (packets)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
	ethernet_state_info_ptr->packet_sec_load_handle = op_stat_reg ("Ethernet.Load (packets/sec)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
	ethernet_state_info_ptr->bit_load_handle       = op_stat_reg ("Ethernet.Load (bits)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
```

```
	ethernet_state_info_ptr->bit_sec_load_handle	= op_stat_reg ("Ethernet.Load (bits/sec)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);


	/*	Register throughput-based statistic handles.	*/

	ethernet_state_info_ptr->packet_thru_handle	= op_stat_reg ("Ethernet.Traffic Received (packets)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

	ethernet_state_info_ptr->packet_sec_thru_handle = op_stat_reg ("Ethernet.Traffic Received
(packets/sec)", OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

	ethernet_state_info_ptr->bit_thru_handle	= op_stat_reg ("Ethernet.Traffic Received (bits)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

	ethernet_state_info_ptr->bit_sec_thru_handle	= op_stat_reg ("Ethernet.Traffic Received (bits/sec)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);


	/*	Register collision-related statistic handles.	*/

	ethernet_state_info_ptr->collision_num_handle = op_stat_reg ("Ethernet.Collision Count",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);


	/*	Register delay statistics.
	*/

	ethernet_state_info_ptr->ete_handle	= op_stat_reg ("Ethernet.Delay (sec)",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

	ethernet_state_info_ptr->global_ete_handle = op_stat_reg ("Ethernet.Delay (sec)",
OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);


	/*	Register statistic to count number of transmission attempts	*/

	/*	per packet transmission.
	*/

	ethernet_state_info_ptr->retrans_handle = op_stat_reg ("Ethernet.Transmission Attempts",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);


	/*	Gigabit Ethernet (1000Base-X) specific statistics.	*/

	ethernet_state_info_ptr->frame_bursting_stathandle	= op_stat_reg ("Ethernet.Burst
ON/OFF",	OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

	ethernet_state_info_ptr->frame_burst_duration_stathandle = op_stat_reg ("Ethernet.Burst Duration
(sec)", OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

	ethernet_state_info_ptr->packets_per_burst_stathandle	= op_stat_reg ("Ethernet.Burst Size
(packets)", OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);


	/*	Statistic collection variable initializations.	*/
```

```
	ethernet_state_info_ptr->eth_mac_global_ete_delay = 0.0;
	ethernet_state_info_ptr->burst_start_time	= 0.0;
	ethernet_state_info_ptr->packets_bursted	= 0.0;

	FOUT;
	}

static void
ethernet_mac_lower_layer_streams_determine (void)
	{
	int			in_strm;
	int		out_strm;
	Objid			out_strm_objid;
	Objid			in_strm_objid;
	int				tx_index;
	int				rx_index;
	int				num_assoc_out;
	int				num_assoc_in;

	/** This function find outs the indices of the streams between the MAC	**/
	/** module and its transmitter and receiver.
	**/
	FIN (ethernet_mac_lower_layer_streams_determine (void));

	/* Obtain the number of outgoing streams connected to the mac module.	*/
	num_assoc_out = op_topo_assoc_count (my_objid, OPC_TOPO_ASSOC_OUT,
OPC_OBJTYPE_STRM);

	/* Obtain the number of incoming streams connected to the mac module.	*/
	num_assoc_in = op_topo_assoc_count (my_objid, OPC_TOPO_ASSOC_IN,
OPC_OBJTYPE_STRM);

	/* Loop through all the outgoing streams.
		*/
```

```
for (tx_index = 0; tx_index < num_assoc_out; ++tx_index)
        {
        /* Obtain ObjectId of the outgoing stream connected to ethernet mac        */
        out_strm_objid = op_topo_assoc (my_objid, OPC_TOPO_ASSOC_OUT,
OPC_OBJTYPE_STRM, tx_index);

        /* Fetch the outgoing stream number from the stream Objid                  */
        op_ima_obj_attr_get (out_strm_objid, "src stream", &out_strm);

        /* We will choose the stream with the stream index that maps to channel */
        /* 0 This will ensure that we transmit on a channel                */
        /* the receiver is capable of listening on                         */
        op_ima_obj_attr_get (out_strm_objid, "dest stream", &in_strm);

        /* Make sure that the stream is not going to higher layer.                 */
        if ((out_strm != ethernet_state_info_ptr->strm_to_higher_layer) && (out_strm !=
ethernet_state_info_ptr->strm_to_ipx) && (in_strm == 0))
                {
                /* Store the outstream to the transmitter.
        */
                ethernet_state_info_ptr->strm_to_lower_layer = out_strm;

                /* Terminate the loop */
                break;
                }
        }

/* Similarly determine the stream going to the MAC's receiver.                     */
for (rx_index = 0; rx_index < num_assoc_in; ++rx_index)
        {
        /* Obtain ObjectId of the ingoing stream connected to ethernet mac.        */
        in_strm_objid = op_topo_assoc (my_objid, OPC_TOPO_ASSOC_IN,
OPC_OBJTYPE_STRM, rx_index);
```

```
		/* Fetch the ingoing stream number from the stream Objid.	*/
		op_ima_obj_attr_get (in_strm_objid, "dest stream", &in_strm);

		/* We will choose the stream with the stream index that maps to channel */
		/* 0. This will ensure that we receivet on a channel	*/
		/* the transmitter is capable of transmitting on	*/

		op_ima_obj_attr_get (in_strm_objid, "src stream", &out_strm);


		/* Verify whether the input stream is from the higher layer. If not,	*/
		/* store the information.
				*/
		if ((in_strm != ethernet_state_info_ptr->strm_from_higher_layer) && (in_strm !=
ethernet_state_info_ptr->strm_from_ipx) && (out_strm == 0))
			{
			ethernet_state_info_ptr->strm_from_lower_layer = in_strm;

			/* Terminate the loop */
			break;
			}
		}

	FOUT;
	}

static void
ethernet_mac_phys_pk_accept (void)
	{
	Packet		*eth_pkptr, *llc_pkptr =
OPC_NIL;
	int		protocol_type = -1;
	Boolean		accept = OPC_FALSE;
```

```
double                                  pk_size;
double                                  ete_delay = 0.0;
const EthT_Frame_Fields*        pk_fd_ptr;
double                                  current_time;
double                                  comp_pk_size;
const OmsT_Vlan_Tag*            tag_ptr;

/* procedure called to accept a packet arriving from the phys. layer */
FIN (ethernet_mac_phys_pk_accept (void));

/* Store the current simulation time into a local variable.        */
current_time = op_sim_time ();

/* acquire the ethernet frame.
*/
eth_pkptr = op_pk_get (intrpt_strm);

if (eth_pkptr == OPC_NIL)
        {
        ethernet_mac_error ("Unable to get frame from physical layer", OPC_NIL, OPC_NIL);
        }

/* Obtain the packet size for the current ethernet packet.         */
comp_pk_size = (double) op_pk_total_size_get (eth_pkptr);

/* Extract the packet fields.                                                      */
if (op_pk_nfd_access_read_only_ptr (eth_pkptr, "fields", (const void **) &pk_fd_ptr) ==
OPC_COMPCODE_FAILURE)
        {
        ethernet_mac_error ("Unable to obtain fields from incoming frame.", OPC_NIL, OPC_NIL);
        }

/* Extract the data from the packet                                        */
```

```
if (op_pk_nfd_get (eth_pkptr, "data", &llc_pkptr) == OPC_COMPCODE_FAILURE)
        {
        ethernet_mac_error ("Received frame with NIL data field.", OPC_NIL, OPC_NIL);
        }

/* Check whether the received packet is tagged. If yes, then the tag      */
/* will contain VLAN classification information of the packet, which we   */
/* also need to pass to higher layer later within the MAC indication      */
/* ICI (unless the MAC is operating in a bridge/switch node, since, in    */
/* that case, the bridge module itself will decide the VLAN               */
/* classification of the message either from its tag it may have or       */
/* from its arrival port).
                */
if (ethernet_state_info_ptr->mac_port_of_a_bridge_switch_node == OPC_FALSE)
        {
        if (op_pk_nfd_is_set (eth_pkptr, "tag") == OPC_TRUE)
                {
                /* The packet is tagged. Get the VLAN information and insert */
                /* into the ICI.
                        */
                op_pk_nfd_access_read_only_ptr (eth_pkptr, "tag", (const void **) &tag_ptr);
                op_ici_attr_set (ethernet_state_info_ptr->llc_iciptr, "vlan_id", tag_ptr->VID);
                }
        else
                {
                op_ici_attr_set (ethernet_state_info_ptr->llc_iciptr, "vlan_id",
OMSC_VLAN_NULL_VID);
                }
        }

/* If this MAC is a part of a end-node, i.e. not operating in a bridge    */
/* or switch node, then we try to reassemble the packet if it is a SAR    */
/* segment. If this segment completes a packet then retrieve it from      */
/* the SAR buffer and continue with regular processing. Otherwise place   */
```

```
        /* the segment onto the buffer and terminate the function since the                */

        /* frame is not complete yet.
                                */

        if ((ethernet_state_info_ptr->mac_port_of_a_bridge_switch_node == OPC_FALSE) &&
(op_sar_pk_is_segment (llc_pkptr) == OPC_TRUE))

                {

                op_sar_rsmbuf_seg_insert (ethernet_state_info_ptr->reassembly_buffer, llc_pkptr);

                if (op_sar_rsmbuf_pk_count (ethernet_state_info_ptr->reassembly_buffer) >= 1)

                        {

                        llc_pkptr = op_sar_rsmbuf_pk_remove (ethernet_state_info_ptr-
>reassembly_buffer);

                        }

                else

                        {

                        op_pk_destroy (eth_pkptr);

                        FOUT;

                        }

                }


        /* Determine if the incoming packet needs to be passed to the packet analyzer.    */

        if (ethernet_state_info_ptr->pkt_capture_info_ptr->num_specifications > 0)

                {

                /* Pass a copy of the data embedded within the ethernet packet.            */

                oms_pkt_analyzer_info_capture (ethernet_state_info_ptr->pkt_capture_info_ptr, pk_fd_ptr-
>type, op_pk_copy (llc_pkptr));

                }


        /* Determine whether or not to accept the packet and send it    */

        /* up to the higher layers.
        */


        /* If the destination address is my address or my link aggregation group*/

        /* address, accept.
                                                */

        if (pk_fd_ptr->dst_addr == my_address || pk_fd_ptr->dst_addr == my_group_address)
```

```
        {
        accept = OPC_TRUE;
        }

/* Check if the destination addr is any of this nodes virtual address.          */
else if (eth_mac_virtual_address_check ((HsrpT_Mac_Address) pk_fd_ptr->dst_addr))
        {
        /* Accept this packet as it is destined to a Virtual address of this    */
        /* same node.
                                                                                */
        accept = OPC_TRUE;
        }

/* If in promiscuous mode and this packet was not sent by this */
/* MAC, accept.
                */
else if ((ethernet_state_info_ptr->promis == 1) && (pk_fd_ptr->last_sent_mac_id != my_objid))
        {
        /* If the node is a gateway, accept all packets. We suppose that this   */
        /* router is a one-armed router. All incoming traffic from the network   */
        /* will be accepted and sent back if the packet is destinated for the    */
        /* same network it is coming from.
                */
        if (ethernet_state_info_ptr->mac_port_of_a_bridge_switch_node == OPC_FALSE)
                {
                /* Don't accept BPDU packets unless the node is a bridge or a switch*/
                /* The gateway should not return BPDU packets otherwise the STA  */
                /* algorithm will disable the loop link.
        */
                if (pk_fd_ptr->type != StbC_Spanning_Tree_Protocol)
                        {
                        accept = OPC_TRUE;
                        }
                }
```

```
        else
                {
                accept = OPC_TRUE;
                }
        }

/* If the packet is a broadcast or multicast packet and the packet was not        */
/* sent by this MAC, accept.
                        */
else if (((pk_fd_ptr->dst_addr == ETH_MAC_BROADCAST_ADDR)
                || (pk_fd_ptr->dst_addr == STBC_MULTICAST_ADDR)
                || (ethernet_mac_addr_is_multicast_addr (pk_fd_ptr->dst_addr)))
                && (pk_fd_ptr->last_sent_mac_id != my_objid))
        {
        accept = OPC_TRUE;
        }

/* If this packet is not acceptable by this MAC, destroy it and exit. */
if (!accept)
        {
        /* If this packet's route is being recorded, then update the information.  */
        if (op_pk_encap_flag_is_set (llc_pkptr, OMSC_RR_ENCAP_FLAG_INDEX))
                {
                char node_name [256];
                oms_tan_hname_get (own_node_objid, node_name);
                oms_rr_info_update (llc_pkptr, node_name);
                oms_rr_info_update (llc_pkptr, "Incomplete (Packet destined to another MAC)");
                }

        /*      Destroy the ethernet packet and the encapsulated packet.        */
        op_pk_destroy (eth_pkptr);
        op_pk_destroy (llc_pkptr);
```

```
		FOUT;
		}


	/* Set the contents of the LLC-destined ICI -- set the address   */
	/* of the transmitting station.
	*/
	if (op_ici_attr_set_int64 (ethernet_state_info_ptr->llc_iciptr, "src_addr", pk_fd_ptr->src_addr) ==
OPC_COMPCODE_FAILURE)
		{
		ethernet_mac_error ("Unable to set source address in LLC ICI.", OPC_NIL, OPC_NIL);
		}


	/* Set the destination address (this mainly serves to                    */
	/* distinguish packets received under broadcast conditions.)    */
	if (op_ici_attr_set_int64 (ethernet_state_info_ptr->llc_iciptr, "dest_addr", pk_fd_ptr->dst_addr) ==
OPC_COMPCODE_FAILURE)
		{
		ethernet_mac_error("Unable to set destination address in LLC ICI.", OPC_NIL, OPC_NIL);
		}


	/* Set the protocol type field contained in the Ethernet frame.  */
	protocol_type = pk_fd_ptr->type;
	if (op_ici_attr_set (ethernet_state_info_ptr->llc_iciptr, "protocol_type", protocol_type) ==
OPC_COMPCODE_FAILURE)
		{
		ethernet_mac_error("Unable to set protocol type in LLC ICI.", OPC_NIL, OPC_NIL);
		}


	/* Accept the incoming ethernet packet. It may receive many          */
	/* kinds of packets [bridge protocol-specific (BPDU, TCN),           */
	/* network layer protocol (IP, IPX) datagrams, or generic            */
	/* application packets.] Based on the node context in which    */
	/* this MAC operates, it may or may not decapsulate these            */
	/* packets before forwarding to the higher layer.                    */
```

```
        /* Check if this MAC port is part of a bridge/switch node. If    */
        /* it is, then we do not need to perform any decapsulation or    */
        /* statistics collection for packets that are not BPDUs.         */
        /* PVSTP BPDUs are also treated like data packets and they are   */
        /* decapsulated later in the pvst_bpe process model.             */
        if ((ethernet_state_info_ptr->mac_port_of_a_bridge_switch_node == OPC_FALSE
                        ||
                (protocol_type == StbC_Spanning_Tree_Protocol && pk_fd_ptr->dst_addr !=
PVST_BPE_MCAST_ADDR )||
                protocol_type == StbC_Link_Aggregation_Protocol))
                {
                /* Either this is not operating in a bridge/switch node or       */
                /* a bridge protocol-specfic packet has been received. De-       */
                /* capsulate the packet to be forwarded to the higher layer      */
                if (llc_pkptr == OPC_NIL)
                        {
                        op_pk_nfd_get (eth_pkptr, "data", &llc_pkptr);

                        if (llc_pkptr == OPC_NIL)
                                {
                                ethernet_mac_error ("Received frame with NIL data field.", OPC_NIL,
OPC_NIL);
                                }
                        }

                /* Get the actual size of higher layer data                      */
                pk_size = (double) op_pk_total_size_get (llc_pkptr);

                /* Update the packet and bit thru statistics.    */
                ethernet_mac_pk_stats_update (llc_pkptr, pk_size, &ethernet_state_info_ptr-
>pk_thru_last_stat_update_time,
                        &ethernet_state_info_ptr->thruput_bgutil_routed_state_ptr,
                        &ethernet_state_info_ptr->packet_thru_handle, &ethernet_state_info_ptr-
>packet_sec_thru_handle,
```

```
                &ethernet_state_info_ptr->bit_thru_handle, &ethernet_state_info_ptr-
>bit_sec_thru_handle, EthC_Traffic_Throughput, do_bgutil);

        /* Record extra data-points to enable proper computation of    */
        /* the "sum/time" based statistics.                                    */
        op_stat_write (ethernet_state_info_ptr->bit_sec_thru_handle,   0.0);
        op_stat_write (ethernet_state_info_ptr->packet_sec_thru_handle, 0.0);

        /*      Compute/record ethernet end-to-end delay statistic.          */
        ete_delay = op_sim_time () - pk_fd_ptr->submit_time;
        op_stat_write (ethernet_state_info_ptr->ete_handle,      ete_delay);
        op_stat_write (ethernet_state_info_ptr->global_ete_handle, ete_delay);

        /* Destroy the ethernet frame which is no longer needed. Note */
        /* that we do destroy only is this node is not a bridge/switch    */
        /* node, as in the other case, the ethernet frame is forwarded    */
        /* to the higher layer.
                */
        op_pk_destroy (eth_pkptr);
        }
else
        {
        /* Extraction of the "data' field of the packet causes it to be    */
        /* no longer be present within the packet. Place it in the packet*/
        /* again before forwarding it to the higher layer.                    */
        op_pk_nfd_set (eth_pkptr, "data", llc_pkptr);

        /* This MAC operates as a port for a bridge/switch device      */
        /* and the contained data packet is not a bridge protocol      */
        /* specific packet. Forward it to the higher layer.                */
        /* Thus, the LLC frame is same as the ethernet frame.          */
        llc_pkptr = eth_pkptr;
```

```
		/* Update the packet and bit thru statistics.   */

		ethernet_mac_pk_stats_update (llc_pkptr, comp_pk_size, &ethernet_state_info_ptr-
>pk_thru_last_stat_update_time,

			&ethernet_state_info_ptr->thruput_bgutil_routed_state_ptr,

			&ethernet_state_info_ptr->packet_thru_handle, &ethernet_state_info_ptr-
>packet_sec_thru_handle,

			&ethernet_state_info_ptr->bit_thru_handle, &ethernet_state_info_ptr-
>bit_sec_thru_handle, EthC_Traffic_Throughput, do_bgutil);


		/* This mac port is contained in bridge/switch node. Do not     */

		/* record any statistic as this port is not the end-point */

		/* for any network traffic.
	*/

		}


	/* Forward the data contents to the higher layer together with ICI. */

	op_ici_install (ethernet_state_info_ptr->llc_iciptr);


	/* Check the value obtained from the 'type' field of the packet. */

	/* Two values defined. A value of 0x800 indicates that the packet       */

	/* should be sent to the IP network layer, and a value of 0x900         */

	/* indicates that the packet should be sent to the IPX network          */

	/* layer. Depending on the result of the check on the 'type' field      */

	/* the corresponding output stream index is checked to see if this      */

	/* module has discovered a neighbor that accepts packets of this        */

	/* 'type' (note: if we are operating in a bridge/switch node there      */

	/* is simply only one stream to higher layer).                                  */

	if ((ethernet_state_info_ptr->mac_port_of_a_bridge_switch_node == OPC_FALSE) &&
(protocol_type == OMSC_PROTOCOL_TYPE_IPX))

		{

		if (ethernet_state_info_ptr->strm_to_ipx != OPC_INT_UNDEF)

			{

			op_pk_send (llc_pkptr, ethernet_state_info_ptr->strm_to_ipx);

			}

		else
```

```
                {
                op_pk_destroy (llc_pkptr);
                }
        }
    else
        {
        op_pk_send (llc_pkptr, ethernet_state_info_ptr->strm_to_higher_layer);
        }

    FOUT;
    }


static void
ethernet_mac_llc_pk_accept ()
    {
    Packet                      *llc_pkptr, *eth_pkptr;
    Ici*                        iciptr;
    OpT_Int64                   dest_addr, src_addr;
    int                             protocol_type = -1;
    double                      pk_size;
    EthT_Frame_Fields*          pk_fd_ptr;
    double                      current_time = 0;
    char                        pk_format [255];
    OmsT_Vlan_Tag*              tag_ptr = OPC_NIL;
    int                             vid;
    OpT_Int64                   vmac_addr = -1;

    /** Accepts an arriving packet from the higher layer (LLC)    **/
    /** It may be a higher layer packet (e.g., IP datagram,       **/
    /** spanning tree bridge protocol data unit) or another               **/
    /** ethernet frame (if this mac process operates within a     **/
```

```
/** bridge/switch node.) If it is not an ethernet frame, **/
/** the packet is encapsulated in an ethernet frame.          **/
/** If this mac operates within a multi-protocol switch,      **/
/** it will have to encapsulate non-ethernet packets as              **/
/** well.
        **/

FIN (ethernet_mac_llc_pk_accept ())

/* Acquire the incoming packet from the higher layer.        */
llc_pkptr = op_pk_get (intrpt_strm);
if (llc_pkptr == OPC_NIL)
        {
        ethernet_mac_error ("No packet available from LLC.", OPC_NIL, OPC_NIL);
        }

/* Get a handle on the ICI associated with this process'     */
/* invocation.
        */
iciptr = op_intrpt_ici ();
if (iciptr == OPC_NIL)
        {
        ethernet_mac_error ("No ICI was received with data from LLC", OPC_NIL, OPC_NIL);
        }

/* Obtain protocol type of this frame if found in ICI.       */
if (op_ici_attr_exists (iciptr, "protocol_type"))
        {
        if (op_ici_attr_get (iciptr, "protocol_type", &protocol_type) ==
OPC_COMPCODE_FAILURE)
                {
                ethernet_mac_error ("Unable to read protocol type from ICI.",
                        "The ICI does have a \"protocol_type\" field.",
                        "Some other error must have occurs.");
```

```
                }
            }

        /* Check whether the ICI associated with the stream          */
        /* interrupt conveying any VLAN classification. If we are    */
        /* in a bridge/switch node, then the ICI may contain a tag   */
        /* with VLAN information. If the packet is coming from ARP   */
        /* then the ICI may contain a valid VID indicating the       */
        /* packet's VLAN classification.                                    */
if (op_ici_attr_exists (iciptr, "vlan_id") == OPC_TRUE)
            {
            /* Get the VID from the ICI.                                    */
            op_ici_attr_get (iciptr, "vlan_id", &vid);

            /* If it is a valid VID create a tag that will be the        */
            /* section of the Ethernet header and carry VLAN             */
            /* information.
            */
            if (vid != OMSC_VLAN_NULL_VID)
                {
                tag_ptr     = oms_vlan_mac_pk_tagstruct_create ();
                tag_ptr->VID = vid;
                }
            }
        else if (op_ici_attr_exists (iciptr, "vlan_tag") == OPC_TRUE)
            {
            /* Get the tag from the ICI. If the ICI doesn't contain      */
            /* any tag then the ICI field will contain OPC_NIL.          */
            op_ici_attr_get (iciptr, "vlan_tag", &tag_ptr);
            }

        /* LLC data larger than the maximum acceptable size will     */
        /* cause an error and the data to be discarded. Thus, a      */
```

```
	/* higher layer packet greater than the ethernet frame		*/
	/* size will be discarded.					*/
	pk_size = (double) op_pk_total_size_get (llc_pkptr);


	/* Update the packet and bit load statistics. */
	ethernet_mac_pk_stats_update (llc_pkptr, pk_size, &ethernet_state_info_ptr->load_last_stat_update_time,
		&ethernet_state_info_ptr->load_bgutil_routed_state_ptr,
		&ethernet_state_info_ptr->packet_load_handle, &ethernet_state_info_ptr->packet_sec_load_handle,
		&ethernet_state_info_ptr->bit_load_handle, &ethernet_state_info_ptr->bit_sec_load_handle,
EthC_Traffic_Load, do_bgutil);


	/*  Get the format of this packet to check if it's ethernet. */
	op_pk_format (llc_pkptr, pk_format);
	if (strcmp (pk_format, "ethernet_v2") != 0)
		{
		/* This MAC is either part of a bridge/switch or has	*/
		/* just received a bridge protocol-specific packet		*/
		/* or non-ethernet packets within a multi-protocol		*/
		/* switch that need to be encapsulated.				*/
		if (pk_size > MAX_DATA_SIZE)
			{
			/* Write a simulation notification log unless it	*/
			/* was written before.
	*/
			if (ethernet_state_info_ptr->dropped_large_pkt_log_written == OPC_FALSE)
				{
				ethernet_dropped_large_pkt_log_write ();
				ethernet_state_info_ptr->dropped_large_pkt_log_written = OPC_TRUE;
				}

			/* throw away the packet and return from invocation. */
			op_pk_destroy (llc_pkptr);
```

```
		FOUT;
		}

	/* Create an Ethernet frame in which to encapsulate the	*/
	/* LLC data.
	*/
	eth_pkptr = op_pk_create_fmt ("ethernet_v2");

	if (eth_pkptr == OPC_NIL)
		{
		ethernet_mac_error ("Unable to create Ethernet frame for encapsulation.",
OPC_NIL,OPC_NIL);
		}

	/* If the tag is valid, set the tag in the packet		*/
	if (tag_ptr != OPC_NIL)
		{
		op_pk_nfd_set (eth_pkptr, "tag", tag_ptr, oms_vlan_mac_pk_tagstruct_copy,
				oms_vlan_mac_pk_tagstruct_destroy, sizeof
(OmsT_Vlan_Tag));
		}

	/*	Create and initialize packet field structure that		*/
	/*	contains dest address, source address, type, fcs and	*/
	/*	last sent mac id.
	*/
	pk_fd_ptr = ethernet_mac_pk_fdstruct_create ();

	/*	Assign packet submission time (to be later used for	*/
	/*	ETE delay calculation.)
*/
	pk_fd_ptr->submit_time = op_sim_time ();

	/* Set the LLC packet in the ethernet frame.		*/
```

```
            if (op_pk_nfd_set (eth_pkptr, "data", llc_pkptr) == OPC_COMPCODE_FAILURE)
                    {
                    ethernet_mac_error ("Unable to insert LLC data into outgoing frame.", OPC_NIL,
OPC_NIL);
                    }
            }
        else
            {
            /* The MAC is operating in a port of a bridge/switch  */
            /* node and has received an ethernet frames that does */
            /* not require encapsulation (i.e., there is no need  */
            /* to create a new ethernet frame).                   */
            eth_pkptr = llc_pkptr;

            /*      Get the fields data structure from the packet.      */
            op_pk_nfd_get (eth_pkptr, "fields", &pk_fd_ptr);
            }

        /* From the ICI, obtain the intended destination address.      */
        if (op_ici_attr_get_int64 (iciptr, "dest_addr", &dest_addr) == OPC_COMPCODE_FAILURE)
            {
            ethernet_mac_error ("Unable to get destination address for outgoing frame.", OPC_NIL,
OPC_NIL);
            }

        /*      Assign the destination address of the new frame.        */
        pk_fd_ptr->dst_addr = dest_addr;

        /* Initialize vmac Address                                      */
        vmac_addr = -1;

        /* Check if the Virtual MAC address exist on this ICI          */
        op_ici_attr_get_int64 (iciptr,  "src_mac_addr", &vmac_addr);
```

```
/* Check if the Virtual MAC address is set on this ICI          */
if (vmac_addr != -1)
        {
        /* Virtual MAC address is set, use it as this packets   */
        /* source address
*/
        pk_fd_ptr->src_addr = vmac_addr;

        /* This ICI will also contain the HSRP info. Extract    */
        /* that information if not already available            */
        if ((hsrp_info_ptr == OPC_NIL) && (op_ici_attr_exists (iciptr, "hsrp_info")))
                op_ici_attr_get (iciptr, "hsrp_info", &hsrp_info_ptr);
        }
else
        {
        /* Assign this station's address to the source address field */
        if (ethernet_state_info_ptr->promis == 0)
                {
                pk_fd_ptr->src_addr = my_address;
                }
        else
                {
                /* When promiscuous mode is enabled, most of the time this   */
                /* means we are operating in a bridge/switch node, but in    */
                /* some special cases that may not be true (i.e. this mode   */
                /* is also enabled in one-armed routers). Hence, make sure   */
                /* being in a bridge/swicth before getting the information   */
                /* from ICI.
                        */
                if (ethernet_state_info_ptr->mac_port_of_a_bridge_switch_node == OPC_TRUE)
                        {
                        op_ici_attr_get_int64 (iciptr, "src_addr", &src_addr);
```

```
                pk_fd_ptr->src_addr = src_addr;
                }
            else
                {
                pk_fd_ptr->src_addr = my_address;
                }
            }
        }

    /* Set the protocol type information of the encapsulated        */
    /* packet in the ethernet frame.                                */
    pk_fd_ptr->type = protocol_type;

    /* Add the self Object ID so that the same packet received back can be discarded */
    pk_fd_ptr->last_sent_mac_id = my_objid;

    /*      Set packet field structure in the ethernet frame.       */
    op_pk_nfd_set (eth_pkptr, "fields", pk_fd_ptr, ethernet_mac_pk_fdstruct_copy,
        ethernet_mac_pk_fdstruct_destroy, sizeof (EthT_Frame_Fields));

    /* if necessary, apply padding to the arriving frame */
    if (op_pk_total_size_get (eth_pkptr) < MIN_FRAME_SIZE + PREAMBLE_SIZE)
        {
        if (op_pk_total_size_set (eth_pkptr, MIN_FRAME_SIZE + PREAMBLE_SIZE) ==
OPC_COMPCODE_FAILURE)
            {
            ethernet_mac_warn ("Unable to set total size of outgoing frame", OPC_NIL,
OPC_NIL);
            }
        }

    /* Obtain the original size of the packet obtained from     the    */
    /* higher layer which is designated as Load.                    */
```

```
pk_size = (double) op_pk_total_size_get (eth_pkptr);

/*      Check if it is in full duplex mode.                     */
if (mac_op_mode == EthC_Mac_Full_Duplex)
        {
        /*      Get the current simulation time.                */
        current_time = op_sim_time ();

        /*      Check if this is the first packet in the queue. */
        if (op_subq_empty (0) == OPC_TRUE)
                {
                if (current_time >= ethernet_state_info_ptr->next_transmission -
PRECISION_RECOVERY)
                        {
                        /*      Send the packet and record next transmission time.      */
                        eth_mac_fdx_pkt_send (eth_pkptr);
                        }
                else
                        {
                        /*      Enqueue the packet and schedule self intrpt for the     */
                        /*  time when the previous packet completes transmission*/
                        op_intrpt_schedule_self (ethernet_state_info_ptr->next_transmission,
EthC_Tx_End);

                        /* Insert the packet in designated subqueue awaiting    */
                        /* the completion of transmission od previous frame and         */
                        /* the interframe gap sequence.
                */
                        if (op_subq_pk_insert (0, eth_pkptr, OPC_QPOS_TAIL) !=
OPC_QINS_OK)
                                {
                                ethernet_pkt_insertion_fail_log_write (op_pk_id (eth_pkptr));
                                op_pk_destroy (eth_pkptr);;
                                }
```

```
                }
            }
        else
            {
            /* Wait in the designated subqueue till the packet's turn comes */
            if (op_subq_pk_insert (0, eth_pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
                {
                ethernet_pkt_insertion_fail_log_write (op_pk_id (eth_pkptr));
                op_pk_destroy (eth_pkptr);
                }
            }
        }
    else
        {
        /* enqueue the now assembled packet behind any other waiting packets. */
        if (op_subq_pk_insert (0, eth_pkptr, OPC_QPOS_TAIL) != OPC_QINS_OK)
            {
            ethernet_pkt_insertion_fail_log_write (op_pk_id (eth_pkptr));
            op_pk_destroy (eth_pkptr);
            }
        }

    FOUT
    }

static void
ethernet_mac_error (const char* msg0, const char *msg1, const char* msg2)
    {
    /** Print error message and exit simulation **/
    FIN (ethernet_mac_error (msg0, msg1, msg2))

    op_sim_end ("Error in Ethernet medium access control model (ethernet_mac):", msg0, msg1, msg2);
```

```
	FOUT
	}


static void
ethernet_mac_warn (const char* msg0, const char* msg1, const char* msg2)
	{
	/** Print error message and exit simulation **/
	FIN (ethernet_mac_warn (msg0, msg1, msg2))

	op_prg_odb_print_major ("Warning from Ethernet medium access control model (ethernet_mac):",
msg0, msg1, msg2, OPC_NIL);

	FOUT
	}


/* Functions that are called by the state transition		*/
/* macros. The functions return 0 or 1 based on the		*/
/* evaluation of the logical condition.				*/

static int
eth_hub_def_on ()
	{
	FIN (eth_hub_def_on ());

	if ((hub_busy == 1 && time_def_chng <= op_sim_time ()) ||
		(hub_busy == 0 && time_def_chng > op_sim_time ()))
		{
		FRET (OPC_TRUE);
		}
	else
```

```
		{
		FRET (OPC_FALSE);
		}
	}

static int
eth_no_hub_def_on ()
	{
	FIN (eth_no_hub_def_on());

	/* If the channel is currently busy, then we are necessarily deferring.  */
	if (op_stat_local_read (RECEIVING_INSTAT) > 0.0)
		{
		FRET (OPC_TRUE);
		}

	/* Deference is true if the interframe gap has not elapsed since the last time the		*/
	/* channel became free, a time which we maintain in a state variable. Note that this		*/
	/* approach respects interframe gaps ensuing not only other stations' transmissions, but	*/
	/* also our own.
	*/
	if (op_sim_time () >= ethernet_state_info_ptr->last_time_channel_became_free +
INTERFRAME_GAP)
		{
		FRET (OPC_FALSE);
		}
	else
		{
		FRET (OPC_TRUE);
		}
	}

static int
```

```
eth_hdx_deference ()
        {
        FIN (eth_hdx_deference ());

        if (NOT_CONNECTED_TO_HUB)
                {
                FRET (eth_no_hub_def_on ());
                }
        else
                {
                FRET (eth_hub_def_on ());
                }
        }


static int
eth_hub_def_low ()
        {
        /* The remote interrupt from the hub indicates that          */
        /* the deference has become OFF.                             */
        FIN (eth_hub_def_low ());

        if (intrpt_type == OPC_INTRPT_REMOTE && intrpt_code == EthC_Deference_Off &&
eth_hub_def_on () == OPC_FALSE)
                {
                FRET (OPC_TRUE);
                }
        else{
                FRET (OPC_FALSE);
                }
        }

static int
```

```
eth_no_hub_def_low ()
        {
        /* Determine if appropriate signalling has been received to indicate that deference is now low.    */
        /* This function evaluates this condition for the case of a half-duplex, direct connect situation;  */
        /* that is, active ethernet ports are connected directly, rather than via a hub.
                */
        FIN (eth_no_hub_def_low ());


        /* The self interrupt with the code                                    */
        /* EthC_Interframe_Gap_Off indicates that the                          */
        /* interframe gap has elapsed, and hence that deference        */
        /* should now be off.                                                  */
        if (intrpt_type == OPC_INTRPT_SELF && intrpt_code == EthC_Intframe_Gap_Off)
                {
                FRET (OPC_TRUE);
                }
        else
                {
                FRET (OPC_FALSE);
                }
        }



static int
eth_hdx_def_low ()
        {
        /* Evaluate whether deference has just transitioned to off. */
        FIN (eth_hdx_def_low ());

        if (NOT_CONNECTED_TO_HUB)
                {
                FRET (eth_no_hub_def_low ());
                }
```

```
        else
                {
                FRET (eth_hub_def_low ());
                }
        }


static int
eth_hub_coll ()
        {
        int             i;

        /* When connected to a hub the collisions are reported      */
        /* from hub with a remote interrupt.                        */
        FIN (eth_hub_coll ());

        i = ((intrpt_type == OPC_INTRPT_REMOTE) && (intrpt_code == EthC_Collision_On));

        FRET (i);
        }

static int
eth_no_hub_coll ()
        {
        /* If not connected to a hub then the MAC detects the */
        /* collisions by itself by listening to the carrier   */
        /* while transmitting. During the transmission if it  */
        /* receives the first bir of an incoming frame then   */
        /* this means a collision.                            */
        FIN (eth_no_hub_coll ());

        if (intrpt_type == OPC_INTRPT_STAT && op_intrpt_stat () == RECEIVING_INSTAT)
```

```
		{
		if (op_stat_local_read (RECEIVING_INSTAT) == 1.0)
			{
			FRET (VOSC_TRUE);
			}
		}

	FRET (VOSC_FALSE);
	}


static int
eth_hdx_coll ()
	{
	FIN (eth_hdx_coll ())

	if (NOT_CONNECTED_TO_HUB)
		{
		FRET (eth_no_hub_coll ());
		}
	else
		{
		FRET (eth_hub_coll ());
		}
	}



static void
eth_mac_fdx_pkt_send (Packet* pkptr)
	{
	double		current_time = 0;
	double		tx_delay = 0;
	double		pksize = 0;
```

```
	/*	Computes delay equal to the transmission and		*/
	/*	interframe gap delay and send the packet.		*/
	FIN (eth_mac_fdx_pkt_send (pkptr));

	/*	Set current time.		*/
	current_time = op_sim_time ();

	/*	Compute transmission delay.		*/
	pksize = (double) op_pk_total_size_get (pkptr);
	tx_delay = pksize / ethernet_state_info_ptr->bit_rate;

	/*	Compute the total time the next packet has to wait	*/
	/*	in the queue before it is sent.		*/
	ethernet_state_info_ptr->next_transmission = current_time + tx_delay + INTERFRAME_GAP;

	/*	Send the packet to the physical medium.		*/
	op_pk_send (pkptr, ethernet_state_info_ptr->strm_to_lower_layer);

	FOUT ;
	}


static void
eth_interrupts_process (int set_deference_timer)
	{
	int		i_stat;
	EthT_Frame_Bursting_Status*		neighbor_burst_status_ptr;

	/* Provides common interrupt processing needed in most states of the protocol.  These are	*/
	/* primarily interrupts that are asynchronous to the operation of the state machine which	*/
	/* is focused on the treatment of one outbound transmission. However, these asynchronous	*/
```

```
/* interrupts (such as processing new arrivals) must be handled whenever they occur.        */
/* The return value indicates whether the interrupt was actually processed.
*/
/* The 'set_deference_timer' argument indicates if the caller is requesting that the inter-  */
/* frame gap timer be set by this procedure. The caller will know if it is appropriate to    */
/* do this, based on the state in which the state machine is currently located.              */
FIN (eth_interrupts_process (set_deference_timer))

/* Divide treatment along the lines of interrupt type. */
intrpt_type = op_intrpt_type ();

/* Stream interrupts are either arrivals from the higher layer, or from the physical layer. */
intrpt_strm = OPC_INT_UNDEF;
if (intrpt_type == OPC_INTRPT_STRM)
        {
        /* Determine the stream on which the arrival occurred. */
        intrpt_strm = op_intrpt_strm ();

        /* If the event was an arrival from the physical layer, accept the packet and decapsulate it. */
        if ((intrpt_strm == ethernet_state_info_ptr->strm_from_higher_layer) || (intrpt_strm ==
ethernet_state_info_ptr->strm_from_ipx))
                {
                ethernet_mac_llc_pk_accept ();
        }

        /* Otherwise, the event must be an arrival from the higher layer; accept the packet and
enqueue it.     */
        else
                {
                ethernet_mac_phys_pk_accept ();
                }
        }

else if (intrpt_type == OPC_INTRPT_STAT)
```

```
{
/* Determine on which statistic input a change has occurred. */
i_stat = op_intrpt_stat ();

/* The 'receiving' input statistic reflects activity on the physical media; if it has moved to  */
/* a low value, this may have implications for the value of the 'deference' signal which
*/
/* regulates access to the media.
                                                                                                */
if (i_stat == RECEIVING_INSTAT && op_stat_local_read (RECEIVING_INSTAT) == 0.0)
        {
        /* Remember the last time that the channel went through this transition. */
        ethernet_state_info_ptr->last_time_channel_became_free = op_sim_time ();

        /* Under certain circumstances, the caller requests that this procedure react to     */
        /* the channel going free by activating an interframe gap timer; this is aprpopriate*/
        /* if we are currently deferring to a passing frame, for example, and that frame    */
        /* has just ended.
                                                                                                */
        if (set_deference_timer == OPC_TRUE)
                {
                /* If we are in a direct connection, half-duplex configuration, then this
signal    */
                /* means that we should start timing the interframe gap; (in a hub-
attachment        */
                /* configuration, the deference process is modeled within the hub and
signalling is */
                /* perfomed via remote interrupts instead.
                                                                                                */
                if (NOT_CONNECTED_TO_HUB)
                {
                        /* In the special case of Gigabit ethernet, we also require that our
'neighbor'         */
                        /* not be bursting in order to start timing the interframe gap (i.e., if
it is       */
                        /* bursting, it will control the media for multiple consecutive
transmissions).    */
```

```
                                        if ((int) ethernet_state_info_ptr->bit_rate == 1000000000)
                                                {
                                                /* Obtain the frame bursting status of the sending      */
                                                neighbor_burst_status_ptr =
(EthT_Frame_Bursting_Status *) op_ima_obj_svar_get (ethernet_state_info_ptr->neighbor_mac_id,
"frame_burst_status");
                                                if (*neighbor_burst_status_ptr ==
EthC_Frame_Bursting_Off)
                                                        {
                                                        /*      Since the neighbor is not bursting, we
can safely       */
                                                        /*      schedule a self-interrupt to indicate
deference-off    */
                                                        /*      after an INTERFRAME GAP has
expired.                        */
                                                        eth_interframe_gap_schedule ();
                                                        }
                                                }

                                        else
                                                {
                                                /*      Schedule a self-interrupt to indicate deference-off
        */
                                                /*      after an INTERFRAME GAP has expired.
                */
                                                eth_interframe_gap_schedule ();
                                                }
                                        }
                                }
                        }
                }

        else
                {
                /* An interrupt has occurred which is not of type stream or statistic      */
                /* Each state is expected to perform its own treatment of these cases.     */
```

```
		/* However, since it will be a remote or self-interrupt, load the state	*/
		/* variable for the interrupt code; this avoids redundant calls to the	*/
		/* KP for obtaining this code throughout the process model.			*/
		intrpt_code = (EthT_Mac_Intrpt_Code)op_intrpt_code ();
		}

	FOUT;
	}


static void
eth_interframe_gap_schedule ()
	{
	double					current_time;

	/* Cause an interrupt to occur after an interframe gap has elapsed. The beginning of the	*/
	/* gap is provided in the state variable 'last_time_channel_became_free', which must be		*/
	/* set to a new time whenever a transmission of our own completes or aborts; and whenever	*/
	/* the channel is detected to go low. This procedure enforces that there can be only one	*/
	/* outstanding interrupt for interframe gap timing at any given moment.
		*/
	FIN (eth_interframe_gap_schedule ());

	/* Get current time for various calculations. */
	current_time = op_sim_time ();

	/* Normally, this procedure should not be called to schedule overlapping interframe gap events.
	*/
	/* However, just in case, protect against this happening by canceling the existing one if it	*/
	/* is still valid. It is not necessary to do this test if we have exceeded the time for which	*/
	/* this timer was scheduled, the last time this procedure was called. This is guaranteed not to
	*/
	/* happen on the first call to this procedure due to the initialization of 'last_end_of_gap_time'	*/
	if (ethernet_state_info_ptr->end_of_gap_time >= current_time)
```

```
		{
		if (op_ev_valid (ethernet_state_info_ptr->end_of_gap_ev_handle))
			{
			/* If the current interrupt is scheduled already for the exact time at which we are
	*/
			/* planning to schedule the end of intreframe gap interrupt then there is no need to
	*/
			/* continue and redo the same thing.
								*/
			if (ethernet_state_info_ptr->end_of_gap_time == ethernet_state_info_ptr-
>last_time_channel_became_free + INTERFRAME_GAP)
				FOUT;

			op_ev_cancel (ethernet_state_info_ptr->end_of_gap_ev_handle);
			}
		}

	/* The interframe gap ends INTERFRAME_GAP seconds after the channel becomes free.
			*/
	/* Normally, this procedure should not be called if that time would be in the past.
		*/
	/* However, if this happens, assume a self interrupt was missed (e.g., due to a small
	*/
	/* floating point discrepancy) and that it is still needed. In that case, schedule it for now.  */
	/* Though this condition is not expected, this behavior ensures against deadlocks when the self-*/
	/* interrupt is required to get out of state.
						*/
	ethernet_state_info_ptr->end_of_gap_time = ethernet_state_info_ptr->last_time_channel_became_free
+ INTERFRAME_GAP;
	if (ethernet_state_info_ptr->end_of_gap_time < current_time)
		{
		ethernet_state_info_ptr->end_of_gap_time = current_time;
		}

	/* With the desired time end time calculated, we can schedule the interrupt to occur with its	*/
	/* unique code. Also, retain the handle for the event, so it can be canceled (see above.)	*/
```

```
	ethernet_state_info_ptr->end_of_gap_ev_handle = op_intrpt_schedule_self (ethernet_state_info_ptr->end_of_gap_time, EthC_Intframe_Gap_Off);


	FOUT

	}


static void

ethernet_mac_stat_intrpt_disable ()

	{

	Objid			statwire_id;

	Boolean			edge_trig_value = OPC_FALSE;

	int				i;

	int				num_stat_wire = 0;


	/**	Disables all statistic intrpts if operation mode is specified as full duplex.	**/

	FIN (ethernet_mac_stat_intrpt_disable ());


	/*	Count the number of connected statwires. */

	num_stat_wire = op_topo_assoc_count (my_objid, OPC_TOPO_ASSOC_IN, OPC_OBJTYPE_STATWIRE);


	/*	Disable the triggers for each of them.	*/

	for (i = 0; i < num_stat_wire; i++)

		{

		statwire_id = op_topo_assoc (my_objid, OPC_TOPO_ASSOC_IN, OPC_OBJTYPE_STATWIRE, i);


		op_ima_obj_attr_set (statwire_id, "rising edge trigger", edge_trig_value);

		op_ima_obj_attr_set (statwire_id, "falling edge trigger", edge_trig_value);

		}


	FOUT;

	}

static void
```

```
ethernet_mac_pk_stats_update (Packet* pkptr, double pk_size, double * last_stat_update_time_ptr,
        OmsT_Bgutil_Routed_State** bgutil_routed_state_pptr, Stathandle * packet_shandle_ptr,
        Stathandle * packet_sec_shandle_ptr, Stathandle * bit_shandle_ptr, Stathandle * bit_sec_shandle_ptr,
        EthC_Mac_Traffic_Type traffic_type, Boolean do_bgutil_flag)
        {
        int                             i;
        int                             stat_updates_in_this_interval;
        double                  update_time;
        double                  bkg_utilization = 0.0;
        double                  dist_utilization = 0.0;
        double                  routed_packets_per_sec;
        double                  routed_pk_size_std_dev;
        double                  update_interval;
        double                  max_bkg_bits_per_update;
        double                  max_bkg_pks_per_update;
        double                  bkg_bits_per_update;
        double                  bkg_pks_per_update;
        double                  curr_sim_time;
    Packet*     bgutil_pkptr;

        /**     Record the ethernet stats for packet and bit throughput.                **/
        /**     These statistics take into account background traffic, if any.          **/
        /** If the packet pointer is non-nil, we examine the packet to                  **/
        /** determine if there is new background utilization traffic specified      **/
        /** in the packet.
                                **/
        FIN (ethernet_mac_pk_stats_update (pkptr, pksize, ...));

        /* Store current simulation time.    */
        curr_sim_time = op_sim_time ();

        if ((do_bgutil_flag) && (pkptr != OPC_NIL)&&
```

```
            (op_pk_encap_flag_is_set(pkptr, OMSC_BGUTIL_ENCAP_FLAG_INDEX) ==
OPC_TRUE))

            {

            /* Access the encapsulated bgutil packet.                */

            op_pk_encap_pk_access (pkptr, "bgutil_tracer", &bgutil_pkptr);


            /* Add ethernet overhead to background traffic. */

            oms_bgutil_segmentation_info_update (bgutil_pkptr, OPC_NIL, OmsC_Tracer_Ethernet,

                        OMSC_BGUTIL_OVERHEAD_ETHERNET,
OMSC_BGUTIL_DO_NOT_SEGMENT, 0.0,

                        oms_bgutil_tracer_segment_func, OPC_FALSE, OPC_NIL);


            if (*bgutil_routed_state_pptr == OPC_NIL)

                        {

                        /* this is the first tracer packet - allocate bgutil routed state */

                        *bgutil_routed_state_pptr =

                                    oms_bgutil_routed_state_create (UNITS_IN_BPS, DO_NOT_SCALE);

                        }

            }


      /* We are only interested in background utilization if this mac is in full */

      /* duplex mode.  We ignore background utilization in half duplex mode.                */

      if ((FULL_DUPLEX) && (*bgutil_routed_state_pptr != OPC_NIL))

                  {

                  /* Check to see if we have a packet, */

                  if (pkptr != OPC_NIL)

                              {

                              /* This packet might contain a background utilization tracer packet.
*/

                              /* Obtain the background utilization till the current time.
                              */

                              /* The value returned denotes the background utilization which is valid until      */

                              /* right before the current time.  This function is efficient if there is no  */

                              /* routed background utilization.
                                                */
```

```
                bkg_utilization = oms_bgutil_routed_utilization_get (*bgutil_routed_state_pptr,
                        pkptr, curr_sim_time, ethernet_state_info_ptr->bit_rate,
&routed_packets_per_sec,
                        &routed_pk_size_std_dev, OmsC_Bgutil_Get_Prev);
                }
        else
                {
                /* We do not have a packet to examine, but we still need to purge      */
                /* any utilization that has expired and determine the current          */
                /* utilization.                                                        */
                bkg_utilization = oms_bgutil_routed_table_purge_stale
(curr_sim_time,ethernet_state_info_ptr->bit_rate,
                        &(*bgutil_routed_state_pptr)->table, &routed_packets_per_sec,
&routed_pk_size_std_dev);
                }
                }

    if (!FULL_DUPLEX || bkg_utilization == 0.0)
                {
                /* There is only explicitly modeled traffic. Record    */
                /* statistics w.r.t. only explicit traffic.            */
                /*      Calculate Statistics. In the model code (below), we    */
                /*      will record the "<units>/sec" statistic in <units>     */
                /*      where <units> can be "bits" or "packets". OPNET's      */
                /*      statistics "capture mode" feature will be used to      */
                /*      record it in <units>/sec.                              */
                op_stat_write (*packet_shandle_ptr,     1.0);
                op_stat_write (*packet_shandle_ptr,     0.0);
                op_stat_write (*packet_sec_shandle_ptr, 1.0);
                op_stat_write (*packet_sec_shandle_ptr, 0.0);
                op_stat_write (*bit_shandle_ptr,     (double) pk_size);
                op_stat_write (*bit_shandle_ptr,     0.0);
                op_stat_write (*bit_sec_shandle_ptr, (double) pk_size);
```

```
        op_stat_write (*bit_sec_shandle_ptr, 0.0);

        /* Maintain time at which these statistics were last updated.    */
        *last_stat_update_time_ptr = curr_sim_time;
        }
else
        {
        /* Compute  statistics for the time upto                         */
        /* the current time. Find the time elapsed since          */
        /* the last statistic update.                                    */
        update_interval = curr_sim_time - *last_stat_update_time_ptr;

        /* Determine how many stat updates we should have  */
        /* during this interval.                                         */
        stat_updates_in_this_interval = oms_bgutil_num_updates_get (update_interval);

        /* We will start our updates at the time of last stat update. */
        update_time = *last_stat_update_time_ptr;

        /* bit_rate is the tx rate of the transmitter we are attached to. */
        max_bkg_bits_per_update = ethernet_state_info_ptr->bit_rate * update_interval /
                stat_updates_in_this_interval;

        max_bkg_pks_per_update = routed_packets_per_sec * update_interval /
                (stat_updates_in_this_interval * bkg_utilization);

        /* Now we will loop and do several stat updates which represent */
        /* the routed background utilization traffic.                    */
        for (i = 0; i < stat_updates_in_this_interval; i++)
                {
                /* Obtain a uniformly distributed utilization while preserving the mean. */
                dist_utilization = oms_bgutil_dist_uniform_utilization (bkg_utilization,
OMSC_BGUTIL_DIST_FACTOR);
```

```
            /* Convert the utilization to the total number of bits and packets for this update. */

            bkg_bits_per_update = max_bkg_bits_per_update * dist_utilization;
            bkg_pks_per_update = max_bkg_pks_per_update * dist_utilization;

            /* Write the stat values for this update. */
            op_stat_write_t (*packet_shandle_ptr,    bkg_pks_per_update, update_time);
            op_stat_write_t (*packet_sec_shandle_ptr, bkg_pks_per_update, update_time);
            op_stat_write_t (*bit_shandle_ptr,     bkg_bits_per_update, update_time);
            op_stat_write_t (*bit_sec_shandle_ptr, bkg_bits_per_update, update_time);

            /* Move forward to the next stat update time. */
            update_time += (update_interval / stat_updates_in_this_interval);
            }

        /* Update the statistic based on the current packet.     */
        op_stat_write (*packet_shandle_ptr,     1.0);
        op_stat_write (*packet_shandle_ptr,     0.0);
        op_stat_write (*packet_sec_shandle_ptr, 1.0);
        op_stat_write (*packet_sec_shandle_ptr, 0.0);
        op_stat_write (*bit_shandle_ptr,     (double) pk_size);
        op_stat_write (*bit_shandle_ptr,     0.0);
        op_stat_write (*bit_sec_shandle_ptr, (double) pk_size);
        op_stat_write (*bit_sec_shandle_ptr, 0.0);

        /* Maintain time at which this statistic was last updated.          */
        *last_stat_update_time_ptr = curr_sim_time;
        }

FOUT;
}
```

```
static void

ethernet_mac_pk_stats_update_endsim (void* PRG_ARG_UNUSED (dummy_state), int
PRG_ARG_UNUSED (dummy_code))

        {

        /** This procedure is scheduled for the end of simulation for the        **/

        /** purpose of providing a background utilization stat update if         **/

        /** needed.
                                **/

        FIN (ethernet_mac_pk_stats_update_endsim (dummy_state, dummy_code));


        /* Update first the load stats, then the traffic received stats.    */

        /* A nil packet pointer means that we do not have a new packet                */

        /* to examine for background utilization information.                         */


        /* Update the packet and bit load statistics. */

        ethernet_mac_pk_stats_update (OPC_NIL, 0, &ethernet_state_info_ptr->load_last_stat_update_time,

                &ethernet_state_info_ptr->load_bgutil_routed_state_ptr,

                &ethernet_state_info_ptr->packet_load_handle, &ethernet_state_info_ptr-
>packet_sec_load_handle,

                &ethernet_state_info_ptr->bit_load_handle, &ethernet_state_info_ptr->bit_sec_load_handle,
EthC_Traffic_Load, do_bgutil);


        /* Update the packet and bit throughput statistics. */

        ethernet_mac_pk_stats_update (OPC_NIL, 0, &ethernet_state_info_ptr-
>pk_thru_last_stat_update_time,

                &ethernet_state_info_ptr->thruput_bgutil_routed_state_ptr,

                &ethernet_state_info_ptr->packet_thru_handle, &ethernet_state_info_ptr-
>packet_sec_thru_handle,

                &ethernet_state_info_ptr->bit_thru_handle, &ethernet_state_info_ptr->bit_sec_thru_handle,
EthC_Traffic_Throughput, do_bgutil);


        /* Close any files open for packet analysis.   */

        oms_pkt_analyzer_file_close (ethernet_state_info_ptr->pkt_capture_info_ptr);


  FOUT;

        }
```

```
static void
eth_mac_portno_obtain (char* portno_str)
        {
        int                     count = 0;
        int                     index;
        char            temp_str [32];
        char            mod_name [128];

        /**     Returns the port number in a string format (P0)     **/
        /** The port number is obtained from the name of     **/
        /** ethernet_mac module.                                    **/
        FIN (eth_mac_obtain_portno (char* portno_str));

        /* Obtain the name of the current ethernet module.    */
        op_ima_obj_attr_get (my_objid, "name" , &mod_name);

        /* Copy the initial string for the Port in the name      */
        strcpy (portno_str, "P");

/* The name specified is only mac here.    */
if (strlen (mod_name) == 3)
   strcat (portno_str, "0");
else
   {
   for (index = 3; index < strlen (mod_name); ++index)
      {
                        /* Check whether the digit character is a digit          */
      if (mod_name [index] > 47 && mod_name [index] < 58)
        temp_str [count++] = mod_name [index];
      }
   }
```

```
        (hsrp_info_ptr->num_groups <= 0))
        FRET (OPC_FALSE);

    /* Loop through the virtual addresses information and check if          */
    /* any address match the destination mac address                        */
    for (index = 0; index < hsrp_info_ptr->num_groups; index++)
        {
        /* Initialize Virtual address
        */
        virtual_addr_info_ptr = OPC_NIL;

        /* Get the next virtual address info ptr                            */
        virtual_addr_info_ptr = (HsrpT_Virtual_Addr_Info*) hsrp_info_ptr->virtual_addrs_array_ptr
[index];

        /* Check if this is a matching valid virtual address                */
        if ((virtual_addr_info_ptr != OPC_NIL) &&
            (virtual_addr_info_ptr->virtual_mac_address == dest_mac_addr) &&
            (virtual_addr_info_ptr->addr_active == OPC_TRUE))
            FRET (OPC_TRUE);
        }

    FRET (OPC_FALSE);
    }
```