# 國 立 交 通 大 學

資訊工程系

碩 士 論 文

# 在 NCTUns 網路模擬器上支援光學網路模擬及命令控制台

# Supporting Optical Network Simulation and Command Console on the NCTUns Network Simulator

研 究 生：虞孟正

指導教授：王協源　教授

中 華 民 國 九 十 三 年 六 月

# 在 NCTUns 網路模擬器上支援光學網路模擬及命令控制台

# Supporting Optical Network Simulation and Command Console on the NCTUns Network Simulator

研　究　生：虞孟正　　　　Student：Meng-Cheng Yu

指導教授：王協源　　　　Advisor：Shie-Yuan Wang

國 立 交 通 大 學

資 訊 工 程 系

碩 士 論 文

A Thesis

Submitted to Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science and Information Engineering

June 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年六月

# 摘要

本篇論文的主要重點,是呈現在交大網路模擬器上設計和實作全光網路的模擬環境.在本篇論文中,我們提供了兩種全光網路的環境:傳統全光網路(traditional optical networks)和光突發交換網路(optical burst switching networks). 傳統全光網路是使用迴線交換機制(circuit switching),而且在傳送資料前必需設定好光路徑(light path).光突發交換網路的資料流是突發性的,它兼顧迴線交換機制和封包交換機制(packet switching)的特色和優點,是全光網的一大研究領域.

由於目前對全光網路的研究日趨熱門,一個輔助研究的工具是必須的.模擬是具有便利性,精準度和彈性的最佳折衷辦法.數學推導方便但不具有實際應用的考量,而真實機器的實驗亦太過耗費人力金錢,而且不具備彈性.因此,我們發展這個模擬的模組,就是希望能夠題供研究者一個完整且便利的研究工具,使他們能節省時間致力於研發和設計之上.

# Abstract

This paper presents the development of the all-optical network simulation on the NCTUns (NCTU network simulator). In this paper, we provide two simulation environments for all-optical networks. One is traditional all-optical network, and the other is optical burst switching network. The traditional all-optical network is a kind of all-optical networks which is circuit switching and needs to assign light paths before transmitting data. The traffic flow of the optical burst switching networks is bursty, and the users need to reserve a wavelength of the light path temporarily for transmitting bursts. Because the current simulation tools or packages simulate the DWDM environment, traditional all-optical networks or optical burst switching networks respectively, we want to provide a simulation package which integrates these subsystems. By using our system, researchers can simulate the behaviors of the optical internet or observe the performance of the merged all-optical networks.

For traditional all-optical networks, we provide static and dynamic RWA (Route and Wavelength Assignment) schemes so that the users can develop their RWA methods or configure the light paths in their simulation cases. We also provide the virtual ring protection mechanism for optical network survivability. For optical burst switching networks, we provide the basic simulation environment and several published methods and algorithms for burst reservation and contention drop. Finally, we will show that our design and implementation are reasonable and correct.

Besides that, we also provide a useful interface called command console for controlling the devices of the simulated networks. With this interface, the users can operate and monitor the nodes in the simulated network during simulation.

In this paper, we will present the design and implementation of traditional all-optical networks, optical burst switching networks, and command console. We also do the performance evaluation and the function validation of our system.

# 致謝

本篇論文能夠如期而且順利的完成，首先要感謝我的指導老師，王協源博士。在規劃初期，王教授提供了數篇相關工作的文獻和一些設計上的想法，使我能夠很順利的將整個系統架構建立起來。在實作和測試期間，和王教授的討論中也提供許多這個系統在模擬器上可會出現的設計盲點，以及正確性、效能、擴充性的探討。

感謝口試委員林華君博士、吳曉光博士與黃寶儀博士(按姓名筆畫排序)。經由他們的指導和建議，使這篇論文能夠更加完善。

我還要感謝黃鎮遠同學與蔡宏儒同學 (依姓氏筆劃排序)。 在實作上，他們完成了不少程式碼的撰寫，並且在與我的討論中，發現初期設計的一些疏漏和不合理的地方。

最後，我要感謝我的家人。有他們的支持，我才能夠完成本篇論文。

謹此，本人在此向指導老師王協源博士和口試委員林華君博士、吳曉光博士與黃寶儀博士，和黃鎮遠同學與蔡宏儒同學 (依姓氏筆劃排序)，我的家人，以及每個提供模擬器架構和實作細節的學長和同學，表達深切的謝意。

九十三年六月　研究生 虞孟正

# Table of Contents

# List of Figures

# 1. Introduction

Simulation is an important way of researching and developing new architectures or protocols. It is much closer to the real scenario than mathematical modeling and costs much less than real hardware experiments. In fact, we do not have the required hardware and equipments to do real experiment for non-existing paradigms or architectures. In this case, simulation is the best choice to conduct new researches and designs.

Because the growth of the scale and bandwidth requirement of the internet is very high, the next generation equipments have to process data much faster and transmit much more data than before. Besides, as more and more nodes join the internet, it means that extremely high performance of routing and switching abilities must be provided for the backbone network devices.

At present, the network backbone equipments are normally ATM switches and Fast Ethernet routers. Optical transponder and optical fiber carry the optical signal and need OEO (Optical-Electronic-Optical) processing to convert the optical analog signal to digital data, and then the hardware can do computation to the converted packets such as routing, switching, and QoS. Considering to the data rate of the optical fiber and the transmission time of data on optical fiber, the OEO processing time and packet processing time in the electronic domain (segmentation, route computation, buffering, scheduling …etc) are the critical bottleneck of the performance of the routers and switches with optical fibers. This is because the electronic signal speed and silicon chip frequency can not match the light speed. If we want to take the full utilization of the data rate and bandwidth of the optical products with current optical technologies, we have to reduce or eliminate the electronic part as much as possible to reduce the loss of the mismatch of data rate. Therefore, the next generation (or at current time) backbone devices are all optical devices. For example, all-optical switches have only mirrors to directly reflect the incoming light signal to the destined outgoing port, and they do not need to do OEO processing. These switches base on circuit switching paradigm. The network of this architecture saves the OEO time and the store-and-forward time, and we call it “Traditional All Optical network”. This paradigm has some feature such as circuit switching, RWA problem (Routing and Wavelength Assignment), and protection to gain survivability.

And the next competition is: why don’t we use packet switching? The packet switching architecture has a lot of benefits: 1. Connectionless. It needs less management and control at setup. 2. Better utilization of the whole network bandwidth. The main disadvantage of the circuit switching is the low utilization of the circuit in each connection, and the traditional all optical networks have the same disadvantage. Why don’t we use packet switching on the all optical networks? This is because the packet switching devices need buffer to store packets, and lacks of buffer will cause a very serious packet drop problem. The current technologies of optical buffer are not very suitable for real application. Therefore, another new paradigm comes out and merges the advantages of circuit switching and packet switching. It is called optical burst switching. It uses control packet to do a-period-of-time reservation of a switch port for transmitting incoming bursts [4].

The purpose of this paper is building the simulation environment for traditional all-optical networks and optical burst switching networks in the NCTUns. Chapter 3 describes the related work of simulation for all-optical networks, and the design issues for our cases and environments. Chapter 4 is an introduction to the NCTUns (NCTU network simulator), which is used as the platform to develop our systems.

Chapter 5 contains the architecture overview of our system, added modules, added nodes, and modifications of our system. Chapter 6 tells the detailed implementation of our system, including the design consideration, packet format, functions of modules, and cooperation among modules. Besides, the modifications of the GUI and simulation engine will be shown.

Chapter 7 is the performance evaluation. We change the scale of our simulation cases to gain or loose the system load, and we collect the data of performance variation. After that, we analyze the data and prove that the system design is reasonable and acceptable. Chapter 8 is the functionality validation. In chapter 8 we survey papers and analyze our system behaviors, and we prove that the behaviors of our simulation system are correct and they match those of all-optical networks operating in the real world.

Chapter 9 is the design and implementation of command console, which is a useful tool for providing an interface between simulation world and users. Through this interface users can give commands to the simulated devices at runtime. Chapter 10 is about how we can expand our simulation system and the future work.

# 2.Related Work and Design Issues

## 2.1. Related Work

Optical Wavelength Division Multiplexing (WDM) Network Simulator [7], which is built on the ns2, is a simulation environment for WDM (Wavelength Divided Multiplexing) networks. It simulates the WDM environments and behaviors such as RWA and light path setup.

“A Simulation Study of Access Protocols for Optical Burst-Switched Ring Networks” [8] presents the case study of the protocols of optical burst switching networks. It implements several protocols of optical burst switching networks on WDM metro ring architecture.

“Simulation of Optical Burst Switching Protocol and Physical Layers” [9] presents a simulation environment for optical burst switching network on the OPNET network simulator.

WDMGuru of OPNET [10] provides a simulation environment for WDM environment and traditional all-optical networks. It also provides SONET standard in all-optical network simulation.

## 2.2. Our Design Issues

I. NCTUns is a simulator which simulates mainly OSI layer-2 and layer-3 protocols such as routing and switching. About simulating OSI layer-1 (the physical layer), we just simulate the bit error rate, bandwidth and propagation delay. We do not simulate signal strength, power variation, or other optical physical effects. Also, because we treat the layer-1 devices as optical fibers, we do not simulate amplifier, repeater, coupler and other layer-1 devices.

II. At present, there is no integrated package that provides simulation environments for all-optical networks. They focus only on some aspects or layers. In [7, 10], the systems simulate only DWDM environment and traditional all-optical networks. In [8, 9], they simulate only optical burst switching networks. However, we want to build an environment as real as the wide area networks operating in the real world including end-user hosts, border gateways, and switches for backbone networks. Therefore, we have to provide optical nodes such as switches and routers for both traditional all-optical networks and optical burst switching networks.

III. For DWDM environments, we have to divide optical fibers into several wavelength channels. From the view point of the design, we do not treat it as “a fiber between 2 nodes with many channels”, but “many links between 2 nodes”. This is natural, because: 1. we need to transmit traffic on many channels concurrently at the same time. If we use the first concept to design our system, the traffic will never be transmitted concurrently. It violates the event scheduling methods of the NCTUns. Therefore, we have to simulate the DWDM environment by using the second concept. 2. Each channel may have different bit error rates or bandwidths. The transponder of a wavelength channel may use different modulation from the modulations of other channels. It is trivial that we view one channel as a link object. It benefits modularity and system design. Because the NCTUns does not support more than one dedicate links between 2 adjacent nodes, we have to modify it to support such feature. We will discuss this work at 6.8.

IV. Protection for optical network survivability is rarely implemented in simulation domain. This is an interesting feature that affects performance, error rate, and loss rate of the all-optical network. Because most all-optical environments have such feature, we add this module to our design map.

V. For traditional all-optical networks, we have to focus on management considerations. For example, light path setup and protection ring setup are the main considerations. These are the main research areas of the traditional all-optical networks. We provide optimized shortest-path algorithm for static RWA scheme (it will be auto-generated when you finish setting up your simulation topology), manual protection ring setup, and dynamic RWA scheme of light path configuration.

VI. Optical burst switching network [4] is one of the most popular research area in all optical networks. We simulate some basic, published popular methods, such as segmentation [6] and JET (Just-Enough-Time) [5]. We also simulate tunable parameters such as burst length, delay factor, and packet processing time.

All design and implementation details are presented in chapter 5 and chapter 6.

# 3.About NCTUns

NCTUns (NCTU network simulator) is a network simulator developed by Network and System Laboratory, CSIE, NCTU. The differences among NCTUns and other traditional network simulators such as ns2 and OPNET are the “real” simulation methodology and module based structure [1]. Because the simulation environment is “real”, NCTUns supports normal applications. Ns2 and OPNET use a mathematical modeling program or the specified traffic generator which generates traffic flow by mathematical computation to simulate the traffic flow. We can run the general network applications such as ftp and some TCP/UDP traffic generator programs to generate traffic flow on the NCTUns. It means that you can use “ping” or “traceroute” program to manage or watch the simulated networks, or you can run a server and a client program to send or receive packets. The other network simulators do not have this feature.

The other scope of the “real” thing is the methodology of simulating the traffic in the simulation networks. The NCTUns uses OSI layer-3 and layer-4 protocol stacks of the operating system to simulate the TCP/IP protocol. In the real network communications, the packet is generated by the socket, and the socket puts the packet into the kernel. Then the packet is processed by the TCP/IP stack, and the kernel puts the packet into the driver of the network interface card. At last the driver pushes the packet into the network. What NCTUns does to a packet is almost the same way, but the differences are: 1. all of the simulated hosts, routers, and layer-3 devices use the same kernel TCP/IP stacks. This is because they are on the same computer. 2. The behaviors of layer-2 and layer-1 protocols are simulated by the simulation engines and related modules. When the packets of simulation networks enter a special network interface called “tunnel”, the tunnel interface will grab the packets up to the simulation engine. The picture below shows the work flow that a packet travels through simulated networks.

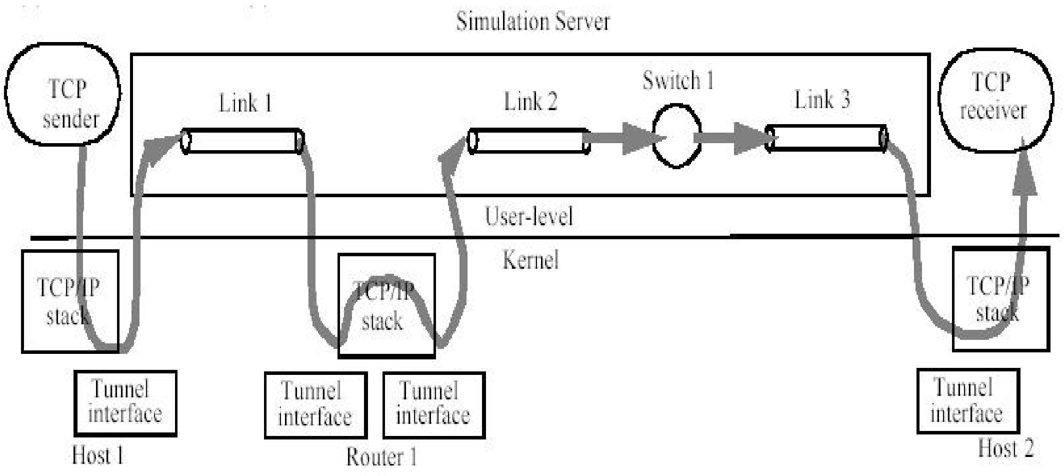


Figure 3: The illustration of simulating the traffic in the NCTUns

From the picture above, we can see that the kernel simulates layer-3 protocols of the hosts and routers, and the simulation engine simulates the protocols of layer-1 and layer-2. The developers of the NCTUns implement the layer-2 and layer-1 modules for simulating the behaviors of the network interface cards and the physical lines (or radio wave in the air). The “real” feature definitely makes the NCTUns an extremely accurate simulator, and therefore we choose it as our development platform. The other reasons that we choose this simulator as our development platform are: 1. the NCTUns uses module based architecture, and 2. NCTUns uses C++ as the development language. The existing modules save our time of developing the other types of networks and some basic network functions. We are familiar to the development language, C++, and therefore we can learn how to use the API quickly. This feature saves our time of studying, and therefore we can focus on the system design and case analysis. Because all these features make the NCTUns the best platform for developing our system, we choose it as the simulation platform.

# 4. High Level System Architecture

## 4.1. System Design

The picture below represents the high level architecture of our system

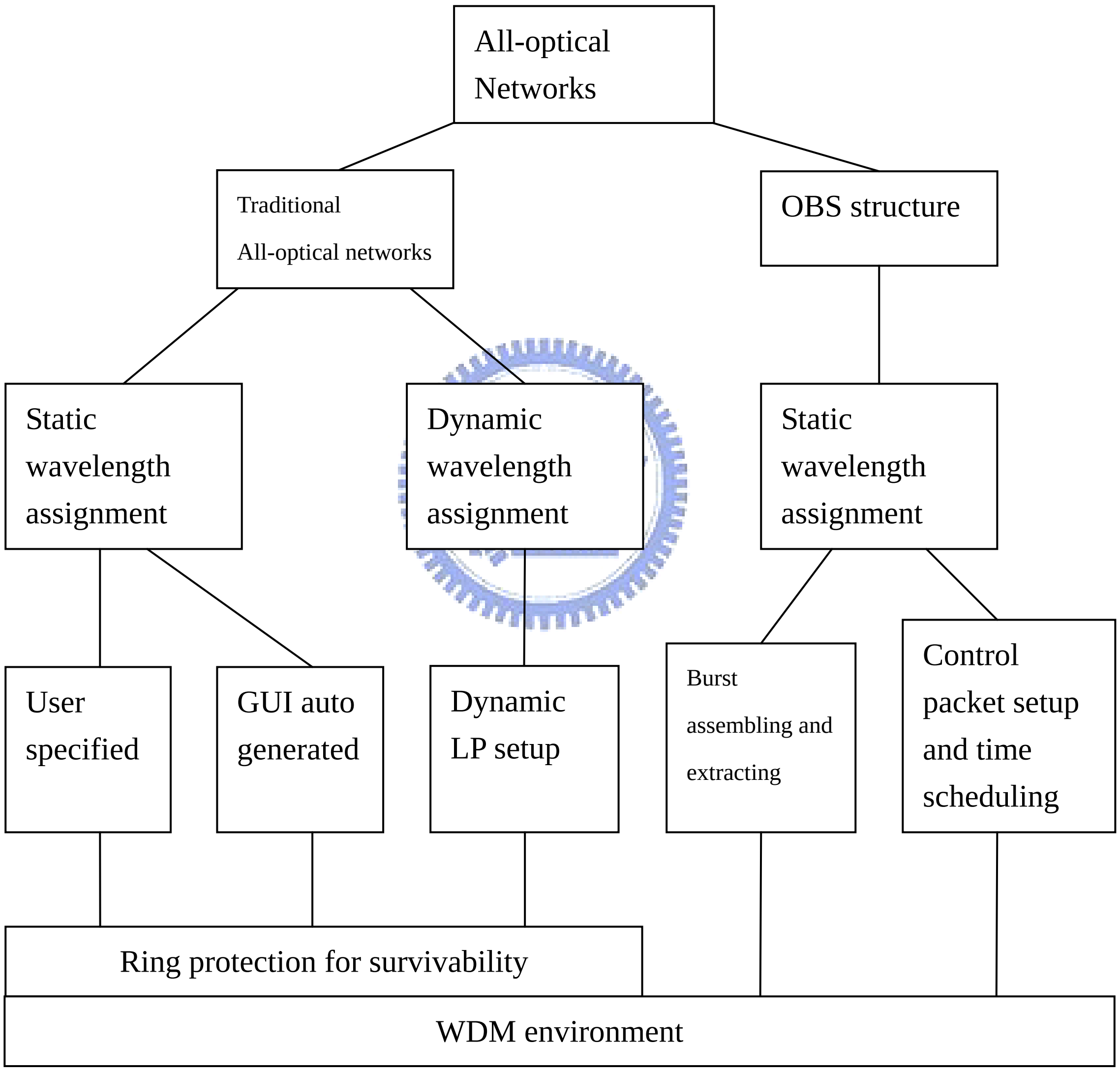

All-optical Networks
Traditional All-optical networks
OBS structure
Static wavelength assignment
Dynamic wavelength assignment
Static wavelength assignment
User specified
GUI auto generated
Dynamic LP setup
Burst assembling and extracting
Control packet setup and time scheduling
Ring protection for survivability
WDM environment

Figure 4: The graph of system architecture

- The whole system is divided into 2 subsystems, one is traditional all-optical networks, and the other is optical burst switching networks.

- The traditional all-optical networks is all-optical. It is based on WDM environment, without OEO signal transformation or the store-and-forward schemes.
- The optical burst switching networks is the new paradigm for optical internet. The traffic of optical burst switching network is bursty and connectionless [4].
- In traditional all-optical networks, we configure circuit for data transferring. The circuit is called light path. It contains the connectivity of nodes and assignment of the wavelength channels. The way to build a light path is static (at the beginning of the simulation) or dynamic (at the incoming of traffic). Because building light paths have something to do with routing, we call it RWA (Routing and Wavelength Assignment).
- In optical burst switching networks, we concern about the burst contention and the drop rate. How and when we build the light path is not important, and dynamic RWA method may cause the incorrect transmission of the bursts and control packets. Therefore we use static RWA scheme.
- We can choose the way of assembling bursts, the way of dropping contended bursts, the way of reserving the control packets, and the way of scheduling the incoming bursts.
- Besides the transmitting and multiplexing functions, our system provides optical protection mechanism for traditional all-optical network survivability. The protection mechanism we provide are SONET protection rings for ring protection and virtual rings for mesh protection [2, 3].

## 4.2. New Added Modules in the NCTUns

### 4.2.1. Optical Physical Module (named "ophy")

I. The ophy module simulates optical fibers. It simulates the propagation delay, the bit error rate, and the bandwidth. It also provides the function that you can set the optical fiber failed during a certain period of time.

II. The ophy module provides packet trace log. It also provides accumulation data log such as drop rates, collision rate, and throughput.

III. This module is used for any type of our all-optical networks.

### 4.2.2. Optical Port Module (named "op")

I. The op module reads the optical header of a packet and it sends the packet to the destined wavelength channel. From the view point of the NCTUns, the op module decides which ophy module the packets should go to.

II. When it receives a LPC (light path configuration) packet, it records the coming-in wavelength number and the going-out wavelength number, and it adds the information on the LPC packets.

III. This module is used for any type of our all-optical networks.

### 4.2.3. Management Module (named "opmanage")

I. This module is the main body of optical network protection and survivability.

II. The main functions of this module are constructing the protection ring and maintaining the survivability of the optical networks. The module will re-switch the traffic to the protection paths if the working paths are broken.

III. The opmanage module has a table of protection rings. This table contains the mapping information of working path and protection path. In general cases, working path is the default way to transmit traffic. The protection path is the backup path when the working path is failed [2, 3].

IV. When the working path is normal, the opmanage module acts as a port-layer multiplexing module. The work of this module is deciding which port the packets should go to.

V. When the opmanage module detects a working-fiber failure, the module will switch the traffic to the protection fiber.

VI. The module is used for our all-optical network systems, but the protection feature is available only in traditional all-optical network. In optical burst switching networks, it is only a module deciding which port the packets should go to.

### 4.2.4. Wavelength Assignment Modules (the one is named “wa” at switches, and the other is named “rwa” at routers)

I. The two modules handle the routing and wavelength assignment configurations. The rwa module is in the router, and the wa module is in the switch. They are used only in the traditional all-optical networks.

II. The rwa module generates the optical header. The modules below the rwa module will transmit the packets to the destined port and destined wavelength channel according to the information of optical header.

III. The wa module stores the light path configurations, and it helps the osw module to build its switching table. For example, if there is a light path "1-2-3-6-7, wave 1", the wa module in node6 will remember "I will send the packets to port 7, wavelength 1, if the packets come from port 3, wavelength 1.", and it updates the switching table in the osw module.

IV. We will discuss the cooperation of the 2 modules with dynamic RWA scheme and static RWA scheme at Chapter 5.

### 4.2.5. Optical Switching Module (named "osw")

I. It is the switching management module. The function of this module is switching the incoming packets to the right light path according to the switching table.

II. The osw module creates and maintains a switching table. The format of this table is: [in-port][in-wave]-[out-port][out-wave].

III. The wa module will tell the osw module the switching information. This is the only way that the osw module obtains the switching information. The wa module creates switching information, and the osw module stores the information and executes the traffic switching.

IV. It is used by all of our all-optical network system.

### 4.2.6. Optical Burst Switching Module (named "obsw")

I. This module is for optical burst switching networks only.

II. The main function of this module is the bursts reservation and contended bursts dropping.

III. It has a reservation table storing and managing the incoming control packets and bursts.

IV. When control packet comes, this module decides to reserve the light path for the burst or not. If the burst reservation is permitted, the obsw module stores the control packet to the reservation table. When the data bursts come, the obsw module decide to let the bursts pass or drop them according to the reservation table.

### 4.2.7. Optical Burst Wavelength Assignment Module (named “obwa”)

I. This module is only for the optical burst switching networks.

II. The module is responsible for generating control packets and aggregating data bursts.

III. When the data packets come, the obwa module queues them in the burst queue. When the burst length reaches the burst length limit or the queuing timer is expired, the obwa module generates a control packet and sends it to the next-hop switch for configuring the burst reservation. After sending the control packet, the obwa module sends the data bursts out.

IV. Of course, the obwa module always uses static RWA scheme to determine which port and which wavelength channel the packets should go to.

## 4.3. The Work Flow of Our System

### 4.3.1. Traditional All-optical Networks

I. At the beginning, users have to set the configuration details of routers and switches such as wavelength conversion, RWA scheme, and protection ring assignment.

II. When the packets come to the border router of the optical networks from other sub-networks, the rwa module checks the packets' destination. Then the rwa module decides which light path the packets should go to according to the next-hop router IP.

III. The optical header (it is equal to the MAC header in Ethernet.) will be created, and the rwa module will attach optical header to the front of the packets. After this attaching process, the packets will be sent to the opmanage module.

IV. The opmanage module sends the packets to the destined op module.

V. When the packets go down to the op module, the module will send the packets to the destined wavelength channel according to the optical header.

VI. When the packets arrive at the ophy module, the ophy module simulates the bit error, transmission time, and propagation delay. Then the ophy module sends the packets to the next node.

VII. When the packets reach the switch, it will be received by the ophy module, and the ophy module sends the packets up to the osw module. The osw module will determine which port and which wavelength channel the packets should go to. The osw module sends those packets down to the lower modules from wa module to ophy module, and the packets go to the next node.

VIII. The step III to step VII will continue until the packets arrive at the destination router port. The packets will be received by the interface module and written into kernel when they reach the destination router.

### 4.3.2. Optical Burst Switching Networks

I. At the beginning, the users set the setting details of routers and switches such as wavelength conversion, burst length, and reservation scheme.

II. When the packets come to the obwa module from another subnet, the module will assemble them to create a burst. If the timer of burst gathering is expired or the burst queue is full, the obwa module generates a control packet and sends it to the next-hop switch. After sending the control packet, the data burst is sent.

III. What the op module, ophy module, opmanage module and the osw module do to the packets in optical burst switching networks is the same as those modules do in the traditional all-optical networks.

IV. When the obsw module gets a control packet, it will check the reservation scheme and the reservation table to decide to offer the bandwidth resources for burst transmission or not.

V. The control packet travels through switches to do burst reservation for its own burst. The travel of control packets ends up at the destination router.

VI. When the data burst (packets) comes to the obsw module, the module sends the burst to the osw module or drops it according to the reservation table and the contention drop scheme.

VII. The burst data will reach the destination router if the destined path is all reserved by its control packet. The burst will be dropped in the middle of the burst transferring if reservation is denied by the obsw module in any one of the optical burst switches.

### 4.3.3. The Scheme of Protection Ring

I. The users have to assign protection rings and set some parameters manually before the simulation starts. The protection mechanism will not work without these works.

II. At normal time, the traffic goes on the working path.

III. When the working path is broken, the opmanage module senses it from the link failure signal triggered by ophy module. The opmanage module automatically switches the traffic to the protection port when it knows that the working path is failed.

IV. When the packets go to the opmanage module, the opmanage module checks whether the packets come from working path or protection path. If they come from working path, the module sends the packets to the working path. If they come from protection path or they come from working path but the working port of this switch is broken, the module sends the packets to protection path.

V. The protection ring mechanism is available only in traditional all-optical networks, but it is not available in optical burst switching networks. Because in optical burst switching networks the transmission of data burst needs to send control packet to each node on the path to the destination to reserve light path for the burst, the changing of light path caused by protection switching will fail the reservation which is already done. If one of the links on the light path is failed, the data burst has to send control packet again to reserve a new light path for transmission.

# 5. Design and Implementation

## 5.1. Packet format

The graphic below represents the format of optical header.

| | | | |
|---|---|---|---|
| sourceIP | destIP | wave | option |

The description of the frames:

I. SourceIP. It is the source port IP of the source router. Its length is 32 bits.

II. DestIP. It is the destination port IP of the next-hop router. Its length is 32 bits.

The sourceIP and destIP can be composed to form a unique ID for a light path. Because RWA process assigns a light path for each route, we need a unique key to represent each route. We choose the source router port IP and the destination router port IP to be the light path ID. From the view of our system architecture, it also means that packets which take the same route travel on the same light path.

III. Wave. It represents the wavelength channel number of the light path that the packets go on. Its length is 8 bits, and therefore we totally support 255 wavelength channels.

IV. Option. It contains the packet type and some configuration options. Its length is 8 bits. The first 4 bits represent the packet type of the packet, and the last 4 bits represent some configuration details. For example, the value 0x00 represents that the packet is data packet. None-zero value of this frame represents that the packet is for special purposes such as configuration or handshaking messages for networks. We will give a detailed explanation later.

## 5.2. Global Functions and Modules

### 5.2.1. Shortest Path RWA Object

It is a function object doing optimized route-and-wavelength assignment. When the simulation network is ready, the number of wavelength channels and the network topology will be input in this object. The object computes and optimizes the RWA. Each assigned light path will take the smallest hop counts (the "hop counts" we talk about here is not the layer-3 router hop counts, but it means the layer-2 switch hop counts), and the whole optical networks use wavelength channels as few as possible. This function will not work when we use dynamic wavelength assignment scheme. This option will be forced to be activated if the simulation networks contain optical burst switching networks. Because the users can develop various route-and-wavelength-assignment strategies for this function, it is designed as an object and it can be customized for research.

### 5.2.2. The Ophy Module

Ophy module has three main parameters. One is "ProgaDelay" which records the propagation delay of the optical fiber, another is "bw" which contains the bandwidth of the channel, and the rest is "BitErr" which represents the bit error rate of the optical link.

The ophy module has 4 main functions: sending packets and receiving packets, simulating propagation delay, bandwidth and bit error, recording the packet trace file for playing traffic flow animation, and logging the accumulation data of the packet flow.

I. Sending and receiving packets

It is the basic function of a module which simulates the physical line of the networks. After executing the member function “init()” (its job is the initialization of the module), the ophy module gets the IDs of nodes connecting to this node, and it keeps the pointer of these nodes in the variable “ConnNode_”. When the packets are ready, the ophy module calls the “get()” function of the ophy module in the connected node to grab the outgoing packets to the destined ophy.

II. Simulating the bandwidth

The ophy module has a variable named “txState” to show whether the ophy module is “sending” the packet or not. We can not let another incoming packet pass if there is a packet in transmission. The variable “txState” is a lock to lock or unlock the outgoing port. When a packet from upper module reaches the “send()” function in the ophy module, the ophy module will check the variable “txState”. If it is false, it means that the ophy module is idle. The ophy module allows the packet to pass when the sending status is idle. Once the ophy module puts a packet on transmission, it changes the value of “txState” to true (locked). Then other packets can not pass until the txState is unlocked again.

At the time the ophy module locks the variable “txState”, it also triggers a timer to simulate the transmission time. The length of the time is “packet length / bandwidth of the ophy module”. When the timer expires, the ophy module calls a function named “TxHandler()”. The function “TxHandler()” unlocks the variable “txState” and do several post-processing.

The method of simulating the bandwidth of the receiving side is similar. The packet is passed to upper module or not according to the variable named “rxState”. The packet can pass if the variable “rxState” is false, otherwise the packet can not. The function named “RxHandle()” controls the status of the variable “rxState” and it is called when the receiving timer is expired.

III. Simulating the propagation delay

Actually, the ophy module do not “send” the packet to the connected ophy module, but it calls the function “get()” of the connected ophy module to grab the packets. It is the main consideration of simulating the propagation delay. When the packet passes the function which simulates bandwidth, it goes to the function of simulating propagation delay. This function will trigger a timer, and the expiration time is the propagation delay of this ophy module. When the timer expires, the ophy module calls the get() function of the connected ophy module to grab the packet. The ophy module of the other side will get the packet after the propagation delay time.

IV. Simulating the bit error

The variable named “BitErr” represents the bit error of the ophy module. We set the bit error rate before the simulation starts. When the packet comes to the ophy module, we add packet information to the packet. The packet information contains the bit error rate recorded in the variable “BitErr”.

When the packet is received by the destination ophy module, the bit error information is extracted. The packet is dropped or not according to the bit error information.

V. Simulation of store-and-forward and non-store-and-forward scheme

We know that the all-optical networks are buffer-less and the transmission scheme is non-store-and-forward. We treat the light path which passes through several optical switches as a straight optical fiber. The inside structure of the all-optical switches are matrices of lens. The incoming optical signal will be reflected to the outgoing port by the lens in switch. From the point of view of all-optical switches, when the first bit is received by the incoming port of the switch, the first bit is sent to the outgoing port at the same time. If the switch uses store-and-forward scheme, the first bit of the packet will not be sent to the outgoing port until the last bit of the packet is received.

For simulating this scheme, we have to know what type this node is at first. Only the traditional all-optical switches and optical burst switching switches have non-store-and-forward characteristic. If the node is a router, the ophy module inside the node should follow store-and-forward scheme. Even if it is an optical burst switch, the control packet needs to be transformed to digital signals (OEO process) to configure the switch and obviously it follows the store-and-forward scheme. Therefore we have such conclusion: 1. the ophy modules in the router should follow store-and-forward scheme. 2. The ophy modules which transmit control packets should follow the store-and-forward scheme.

Now we explain how the ophy module simulates the store-and-forward and none-store-and-forward behaviors. When the ophy module receives a packet from the source another ophy module, it triggers a timer and calls the function “RxHandler()” when the timer expires. If the packet needs to follow store-and-forward scheme, the ophy module passes the packet to the upper module in the function “RxHandler()”. If the packet needs to follow none-store-and-forward scheme, the ophy module passes the packet right now. We pass the packet to the upper module not waiting the last bit of the packet is received in the non-store-and-forward scheme.

The pictures below illustrate these schemes. The doted line represents the edge between receiving and sending. The left side of the dotted line represents the time before receiving / sending. The right side of the dotted line represents the time after receiving / sending.

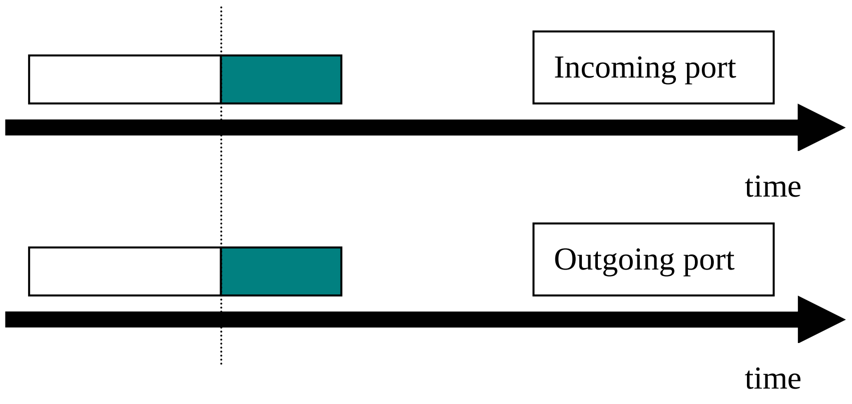


Figure 5.1: The illustration of none-store-and forward.

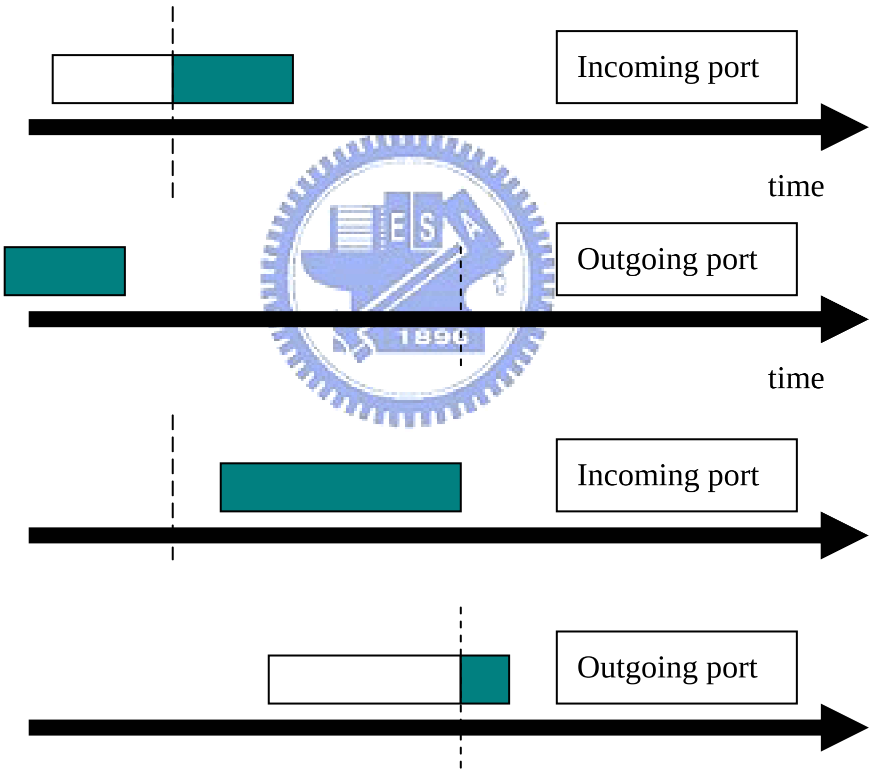


Figure 5.2: the illustration of store-and-forward

In figure 5.1, we can see that when the first bit of the packet comes into the incoming port, the first bit of the packet goes out of the outgoing port. This is the non-store-and-forward scheme. From the view point of optical devices, we treat the non-store-and-forward switches as “a coupler that switches". The incoming optical fiber and the outgoing optical fiber can be seen as one optical fiber composed by these two.

In figure 5.2, it illustrates the store-and-forward scheme. The packet will not be sent to outgoing port until the whole packet is received by the incoming port. In our simulation environment, the optical burst switch has a parameter named "packet processing time" which represents the sum of the OEO time, the routing computation time, and the switching computation time. In optical burst switching networks, this factor is very important. The switch need to transform the control packet from optical domain to electronic domain so that it can compute the burst reservation and change some control information in the packet, and therefore the control packet needs to follow store-and-forward rules. The time that store-and-forward scheme takes is a critical factor to the optical burst switching networks [5]. We have to simulate this feature to make sure that our system is correct.

VI. The packet-trace file logging and accumulation data logging

In the NCTUns, we need to log the packet-trace file in order to provide the GUI the packet streams and behaviors record so that the GUI can draw the play-back animation. The packet-trace printing function in the GUI shows the packet-trace log in detail, and we can inspect the behaviors of the networks. The other log data is performance and accumulation log. This data contains the performance and accumulation information such as number of packet drop, number of packet collision, number of incoming or outgoing packets, and throughput. The performance plotter in the GUI will use this data to draw graphs to show the variation of the data.

For packet-trace logging, we have to record the packets at 4 points: the point of starting transmission, the point of finishing transmission, the point of starting receiving, and the point of finishing receiving, to complete a packet-trace entry. The ophy module calls the function named “sslog()” (sending start log) at the function “send()”, selog() (sending end log) at function “TxHandler()”, rslog (receiving start log) at the function “recv()”, and relog() (receiving end log) at function “RxHandler()”.

For logging the performance data, the ophy module records the number of incoming packets, the number of outgoing packets, and the number of packets that all pass this ophy module. It not only records the number of packets, but also the throughput. For logging the accumulation data, the ophy module records the number of bit-error drop, and number of collision drop.

The picture below shows the finite state machine of the ophy module.

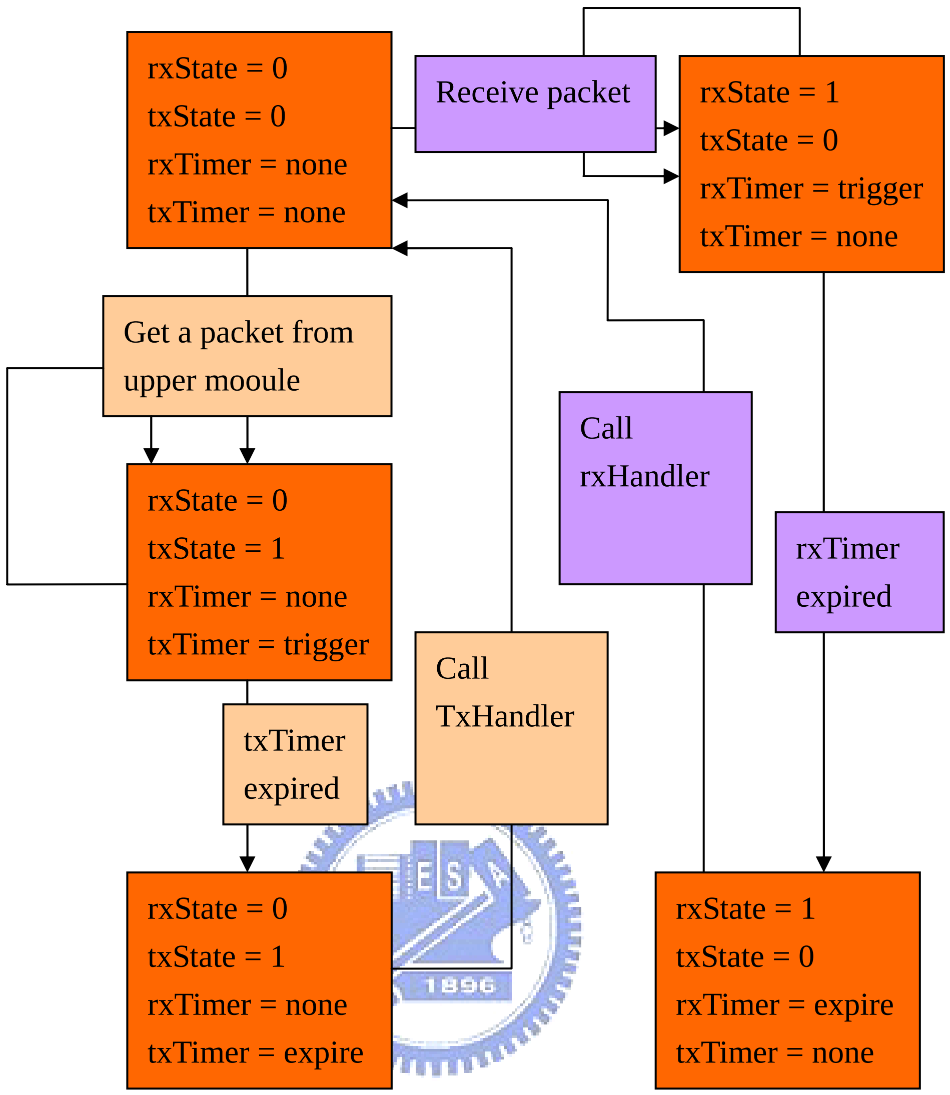


Figure 5.3: The finite state machine of the ophy module

### 5.2.3. The Op Module

The op module has 2 functions. One is that the op module marks the packets with some information. When the packets are received by the op module from the lower module, the op module will add packet information named “From” to the packet. The information represents which port the packets come from. This function is for providing information for switching. The other function is that the op module directs the packets to the right outgoing ports. When the packets are received by the op module from the upper module, the op module will check the optical header of the packets and sends the packets to the correct wavelength channels.

### 5.2.4. The Osw Module

The osw module is the module which switches the incoming traffic in optical switches. The jobs of the module are maintaining the switching table and switching packets according to the switching table. The text below describes the two jobs of the osw module.

I. Creating and maintaining the switching table

The wa module or obsw module will send the light path information to the osw module and the osw module will create a switching table according to these information. A table entry contains source port ID, source wavelength ID, destination port ID, and destination wavelength ID.

II. Switching packets

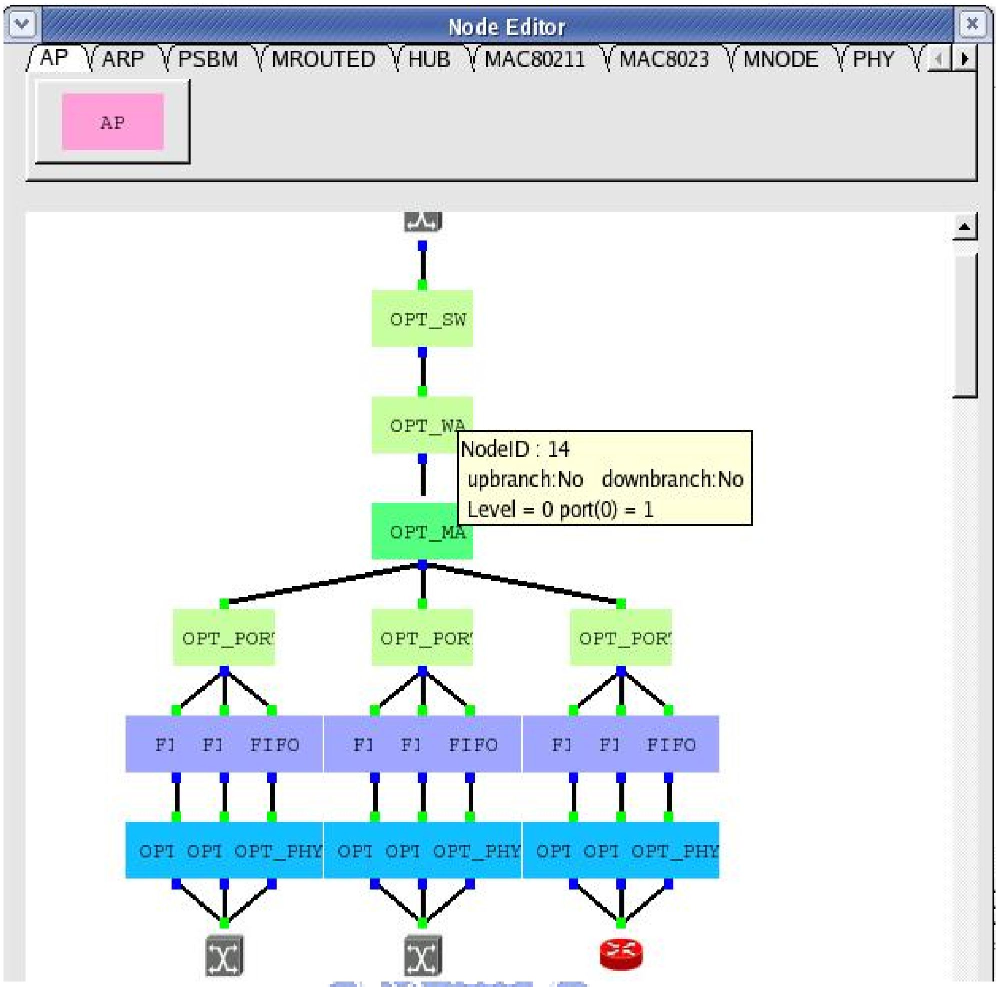
The osw module extracts the packet information named “FROM” which is added by op module. The osw module gets the source port ID and it extracts the wavelength ID from the optical header of the packet. Therefore it knows the source port and the source wavelength. Then the destination port ID and the destination wavelength ID will be obtained by checking the switching table. The osw module updates the optical header and it adds packet information named “TO” for the lower module. After these processing, the modules below the osw module know that which port and which wavelength channel the packets should go to.

## 5.3. New Added Node Types and Their Module Trees

We add 3 new nodes for our all-optical network system.

### 5.3.1. Traditional All-optical Switch

The node is only for the traditional all-optical networks. It connects only to the router node or other traditional all-optical switches. The graphic below shows the module tree of this node. The number of the module named "OPT_PORT" is according to the number of the fiber connected to this switch, and the number of module named "OPT_PHY" is according to the number of wavelength channel.



NodeID : 14
upbranch:No downbranch:No
Level = 0 port(0) = 1

Figure 5.4: The module tree of the optical traditional switch

## 5.3.2. Router Between All-optical Switches and Other Types of Subnet

This router node looks the same as the other router nodes, but the module tree of the router node is different to the other router nodes. The module tree in the router nodes changes according to the type of the connected subnets. Figure 5.5 represents the module tree of the router node which is between Ethernet and traditional all-optical networks. Figure 5.6 represents the module tree of the router node between Ethernet and optical burst switching networks.

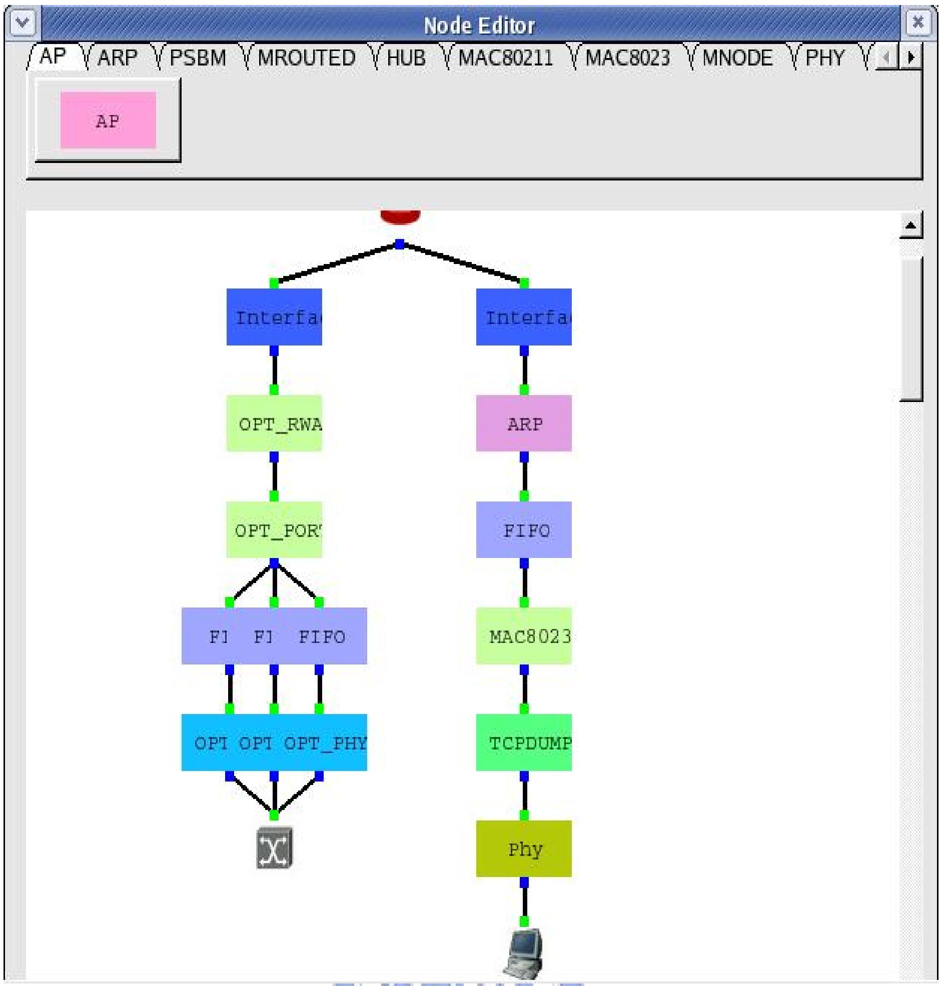


Figure 5.5: The module tree of the border router type1.

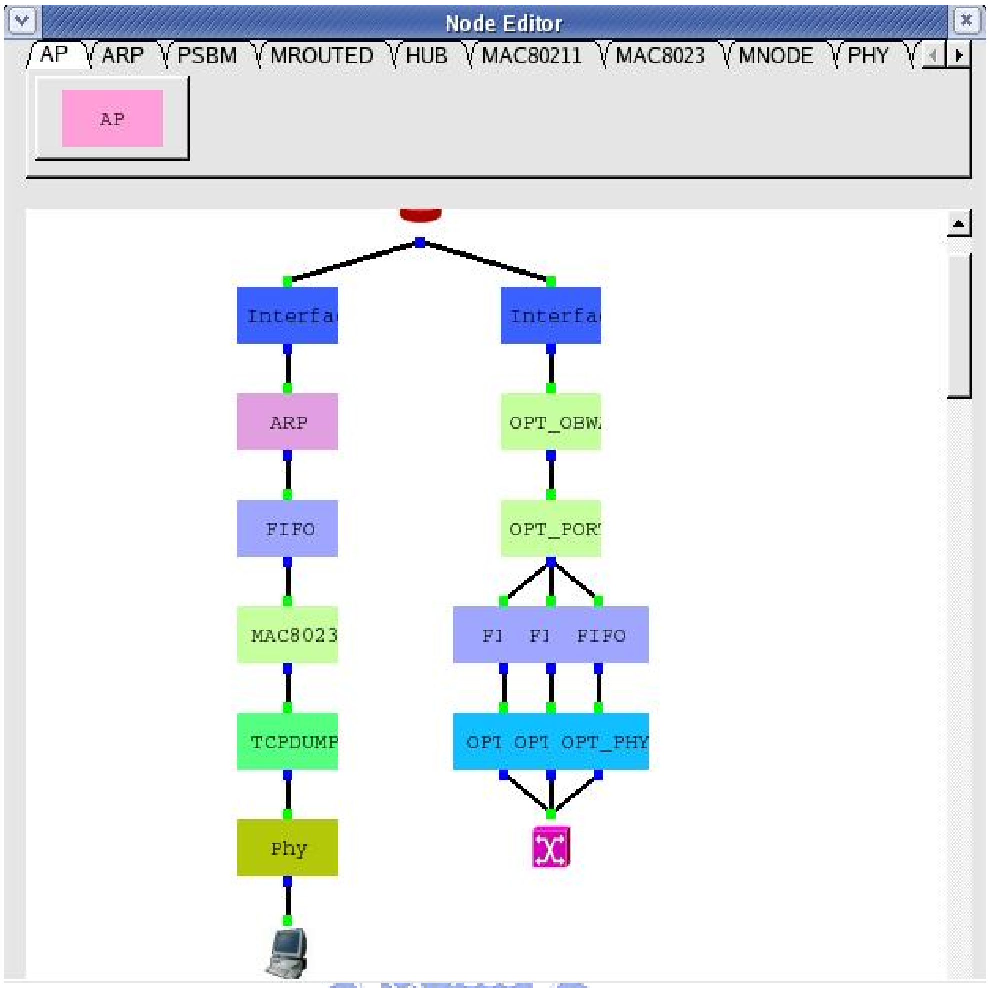


Figure 5.6: The module tree of the border router type2

### 5.3.3. Optical Burst Switch

This node is for optical burst switching networks only. It connects only to the routers node and other optical burst switches. Figure 5.7 shows the detailed module tree of the optical burst switches.

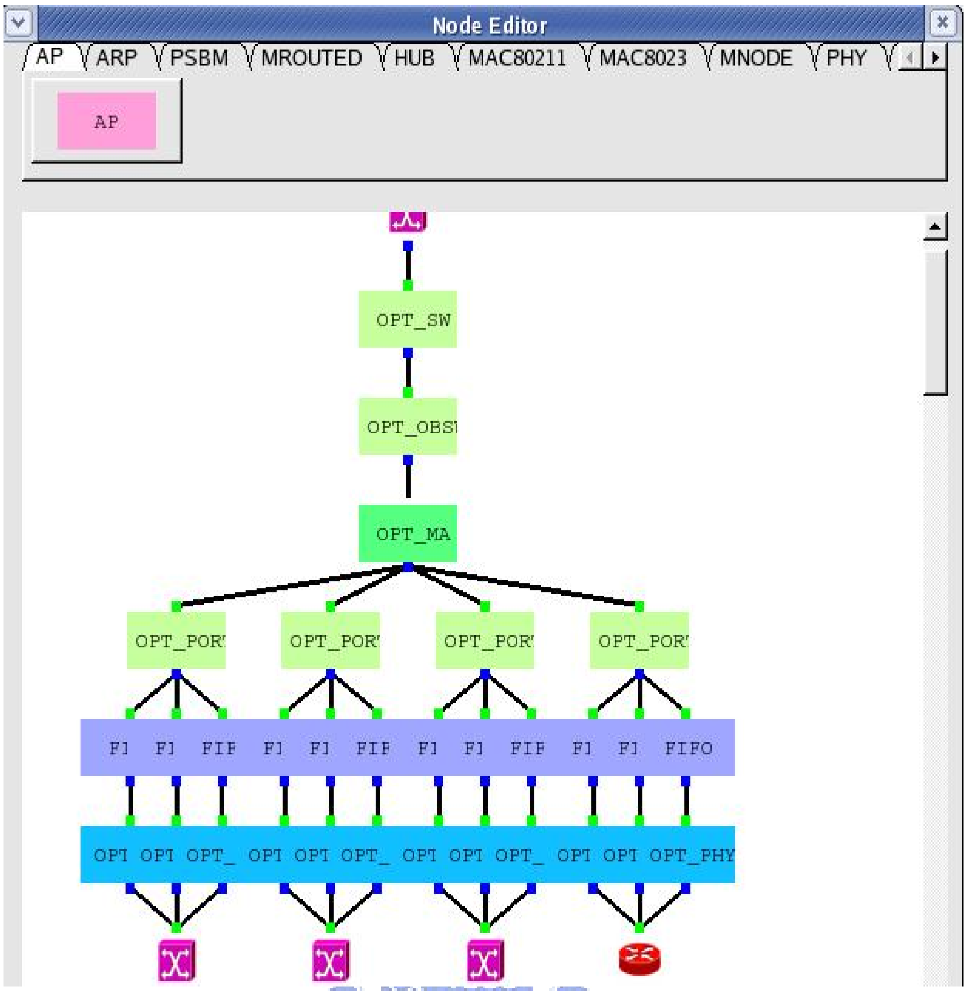


Figure 5.7: The module tree of the optical burst switching switch

## 5.4. Optical Protection Modules and Mechanism

### 5.4.1. Introduction

The bandwidth of all-optical networks is extremely high. If accidental link failures happened at the network devices, a lot of data would be lost even the devices crash for a very short time. Therefore, the protection and backup functions for optical network survivability are very important. SONET has the standard called SONET protection ring which defines the ring protection standard [2]. The standard of mesh protection and virtual ring protection are still in development. In our protection simulating cases, we provide a method to support virtual ring protection. Our scheme of the virtual ring protection also supports ring protection and mesh protection.

### 5.4.2. The Design of Management and Protection Layer

I. The main function of our protection management protocol is doing protection with virtual-ring architecture. We need to assign protection virtual rings manually and we have to follow certain rules [3].

II. Each switch has a module named “opmanage” which manages the work of protection and the ring structure. If no ring is assigned or the working port is normal, the module is only a bridge module to the upper modules and the lower modules. If we assign several rings as the protection rings, the GUI generates a file that contains information of protection rings. The opmanage module of each node reads the file and selects the needed information of the ring as a node. Assuming that we assign a ring “2-3-1-2”, node1 is in ring number 1 and so do node2 and node3. The upper neighbor of node1 is node3, and the lower neighbor of node1 is node2. Now we take a look at the opmanage module in node1. The opmanage module will select the data “3” and “2” as its upper neighbor and lower neighbor on the ring “2-3-1-2”. Because the ring is unidirectional, the opmanage module has to know its upper and lower neighbor. Only optical switches with the same type can be assigned to be a ring.

III. Now we show how the protection ring is assigned and how it works.

Figure g illustrates a network topology.

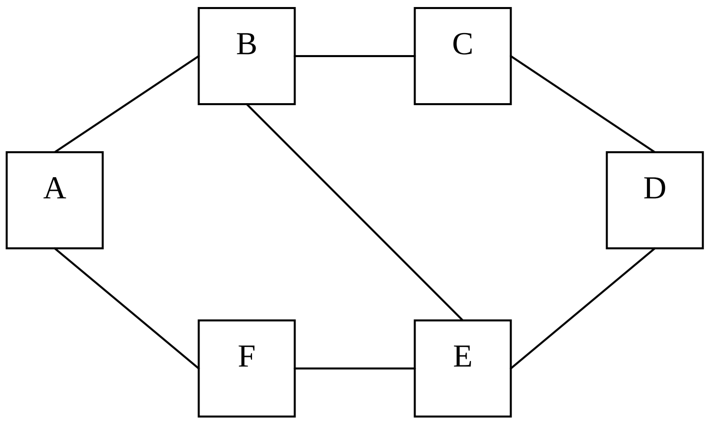


Figure 5.8: An example of protection ring

NodeA, nodeB, nodeC, nodeD, nodeE, and nodeF are assigned as a virtual ring “A-B-C-D-E-F-A”, and nodeA, nodeB, nodeE, and nodeF are assigned as a virtual ring “A-B-E-F-A”. Assuming that we assign the two rings “A-B-C-D-E-F-A” and “A-B-E-F-A“ in this case and the opmanage module creates a protection table in the opmanage module. We now take a look at the opmanage module at node B. The opmanage module at node B has these table entries: {working B→E, protection B→A, ring 1}, {working B→C, protection B→A, ring2}. These entries contain the information of the ring as a ring member. Taking entry1 for example, “{working B→E, protection B→A, ring 1}” means that it is the information of ring1, the direction of working path is B to E, and the direction of protection path is B to A. Because node B joins two virtual rings, the ring table of node B has 2 entries.

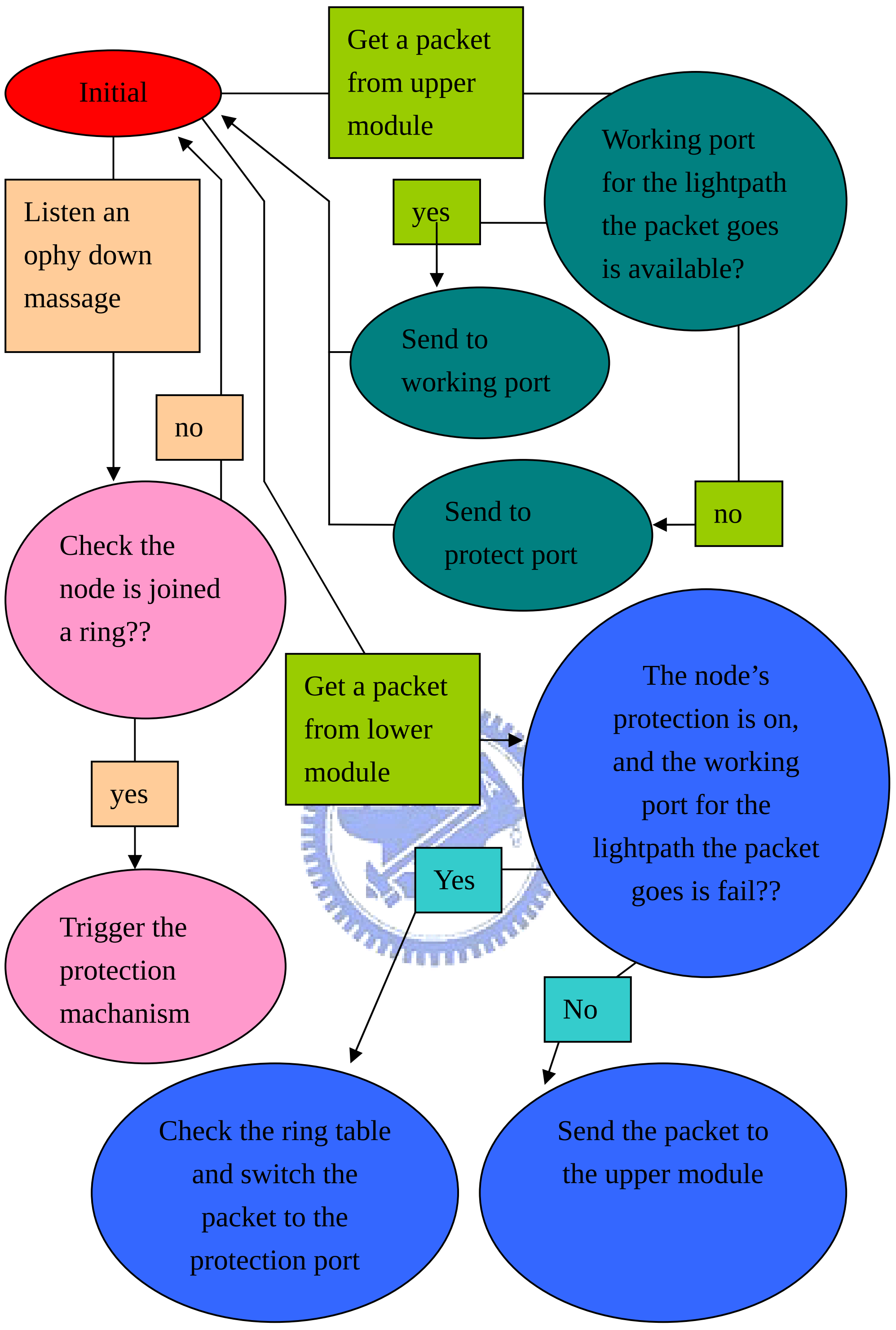
When the working port is failed, the opmanage module checks the table. If the node joins a ring, the traffic route will be changed to the protection path. Each packet has packet information that represents which ring and which path the packet goes on. Switching now is no more handled by the osw module, but handled by the opmanage module. The switching table in osw module is replaced temporarily by the protection table until the link failure is recovered, or the packets arrive at another node which is not in the ring that suffers working path failure.

IV. Light path is unidirectional, and therefore the direction of the light path must not be against the direction of the protection ring. For example, light path "D→C→B→A" is not allowed to be assigned if the ring "A-B-E-F-A" exists. The step of virtual ring setup by the GUI is clicking the ring member sequentially. For example, if we want to assign ring "1-2-3-1", we have to click the node by following the sequence "node1 → node2 → node3 → node1". Different sequence represents different ring.

V. Because there are some limitations in the topology drawer, we support only one bi-directional optical link between two adjacent nodes. Because one bi-directional optical link can be assigned to at most two rings, the direction of one ring on this link must be opposite to another. Taking figure 5.8 as an example, we can assign ring "A-B-C-D-E-F-A" and "A-F-E-B-A". The two rings "A-B-C-D-E-F-A" and "A-B-E-F-A" can't be assigned at the same time. We must follow the double cycle rules [3]. What problems will happen if we don't follow such rules? If the ring "A-B-C-D-E-F-A" and the ring "A-B-E-F-A" are assigned at the same time and the network status are normal, the node B will not know which is the right working path, B-C, B-E, or both? The protection switching may suffer the link overloading problem.

Assuming that the “B→C” and “B→E” are full-utilized and the 2 links break down. Their protection path “B→A” can’t afford the data flow that is double to its bandwidth limit. The overflowed packets will be dropped, and therefore the protection is meaningless. IF the ring “A-F-E-D-C-B-A” and the ring “A-F-E-B-A” are assigned at the same time and the link “A→B” crashes, the opmanage module will not know which protection path is valid, B-C, B-E, or both? When user set up the protection ring, they should follow these rules to avoid these problems.

The graph below represents the working flow chart of the opmanage module.



Initial
Get a packet from upper module
Listen an ophy down massage
Working port for the lightpath the packet goes is available?
yes
Send to working port
no
no
Send to protect port
Check the node is joined a ring??
yes
Trigger the protection machanism
Get a packet from lower module
The node's protection is on, and the working port for the lightpath the packet goes is fail??
Yes
No
Check the ring table and switch the packet to the protection port
Send the packet to the upper module

Figure 5.9: The illustration of the work of the protection module

VI. For supporting mesh protection, we design a smart way to select the path. The picture below ilustrates a hive-structured mesh topology. We will take the picture as our explanation.

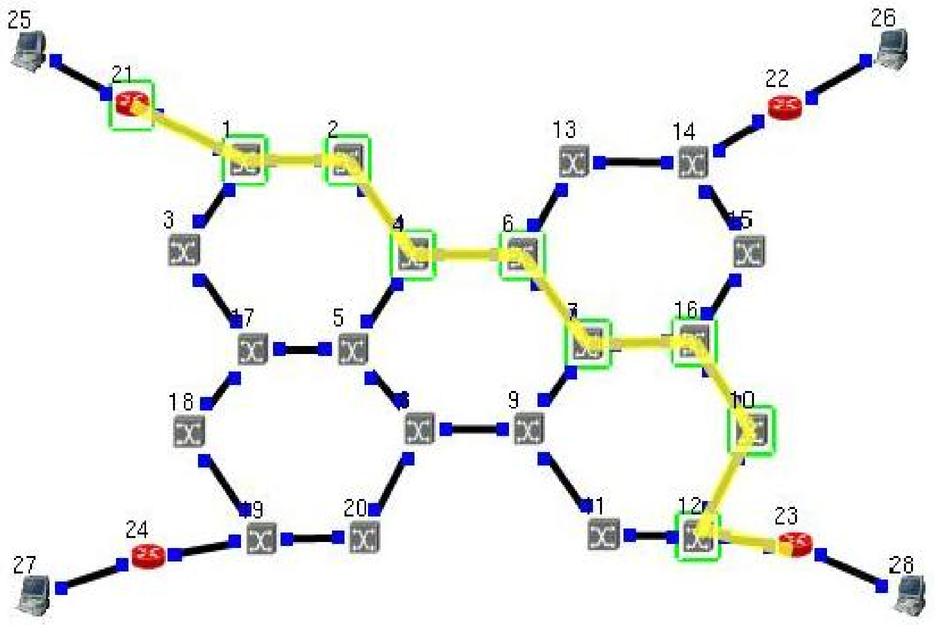


Figure 5.10: The illustration of the work of the protection module

The yellow line is the light path that we assigned manually, and the protection rings are assigned for each hive cell. For example, “1-2-4-5-17-3-1” is assigned as a ring and “4-6-7-9-8-5-4” is assigned a ring. There are total five clockwise rings and each optical link joins at most two rings whose directions are different. This assignment satisfies our rule. We assume that the link “6→7” is broken. When the packet reaches node6, the packet is marked as “from working port” and “taking the ring “4-6-7-9-8-5-4””. After the opmanage module tags the information on the packet, the packet is sent back to node4 and the packet will follow the protection direction until it arrives at node7.

When the packet reaches node7, we know that if the packet went back to node6, the packet would travel in a loop and never reach the destination. We call node7 “the last station of the protection travel of ring “4-6-7-9-8-5-4””. At node7, the packet is marked as “going on the working port” and the opmanage module chooses a working path of another ring that node7 joins. For example, we choose the ring “7-16-10-12-11-9-7”. This is because the link “7→16” is on the light path that the packet takes. Because the node is on the light path or no traffic passes the node, we can always find another ring which the node joins in that its working path overlaps with the light path. Taking the hive-structured topology as an example, node7 is the node on the light path. If node7 is not on the light path, we even don’t take the link “6→7” to the destination. The rules of our working and protection path switching are:

I. If the working path is broken, the opmanage module adds a tag “protection, ring1” on the packets, which represents that the packets will travel on the protection path of ring1.

II. When the packets arrive at the last station of the protection travel, the opmanage module changes the tag to “working, ring3”on the packets, which means that the working path of ring3 is the next working path that the packets will take. Of course, the working path of ring3 in this node overlaps with light path of the packet.

III. If the working path is normal, the traffic follows the path that the osw module decides according to the switching table.

## 5.5. Traditional All-optical Networks

In this section we descript the design and implementation of traditional all-optical networks. This subsystem is composed of edge routers, optical traditional switches, and other types of networks. Most functions of the subsystem are handled by the rwa modules and the wa modules. Figure 5.11 shows a topology of traditional all-optical networks, and we take it as the example of our description.

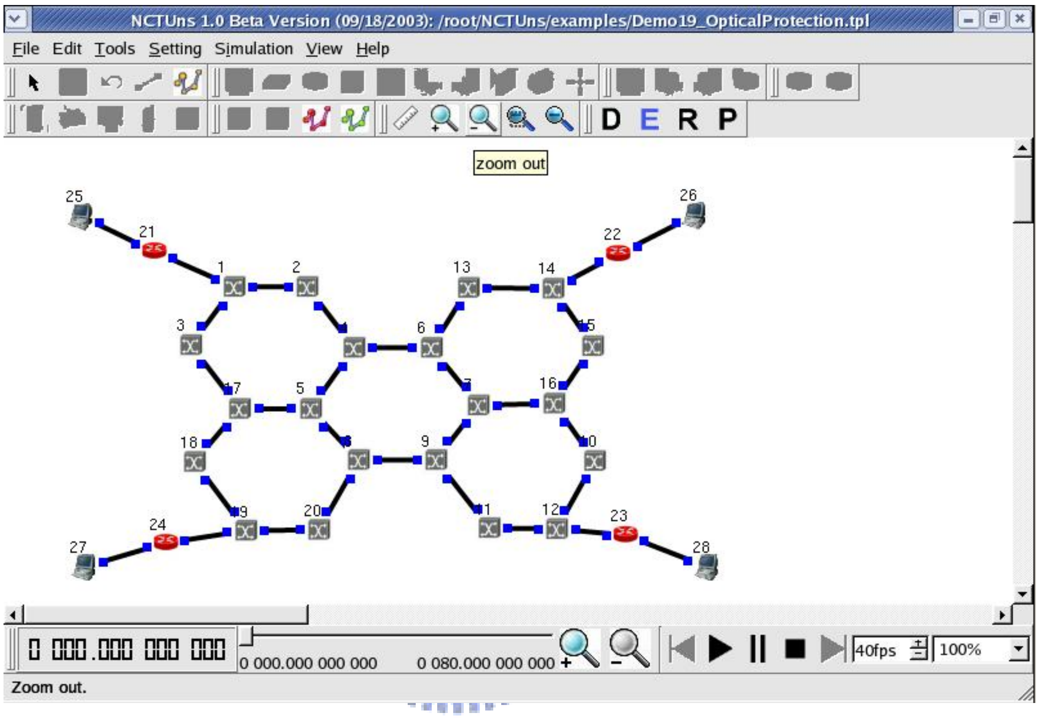

NCTUns 1.0 Beta Version (09/18/2003): /root/NCTUns/examples/Demo19_OpticalProtection.tpl
File Edit Tools Setting Simulation View Help
D E R P
zoom out
25
26
21
22
1
2
13
14
3
6
15
17
5
7
16
18
8
9
10
19
20
11
12
24
23
27
28
0 000.000 000 000
0 000.000 000 000
0 080.000 000 000
40fps
100%
Zoom out.

Figure 5.11: Traditional all-optical networks

### 5.5.1. The Functions of the Rwa Module at Router

We have discussed the function of generating optical header at chapter 4. Besides the functions of rwa module are the configuration of the RWA scheme, which we can set to static (it is done by the users or GUI auto-generated) or dynamic (it finds out a light path automatically at run time). We know that traditional all-optical networks are circuit-switching, and therefore the traffic can reach the destination only if the connection is built up. The RWA must be done before sending the data.

When the packets come to the rwa module, the rwa module finds the destined next-hop router port IP of the packets. The rwa module checks the light path table for free or connected light paths to transmit packets. If there is no such light path, the packet will be dropped.

If we use dynamic RWA scheme, the light path table is empty at start. The rwa module does not drop the packets at the first time, but stores them and creates a light path configure packet (we call it LPC packet) to broadcast to the whole optical subnet. Because we flood the LPC packets, one of the LPC packets will finally arrive at the destination router and we will find a light path for this route. The light path searching process will continue until the rwa module finds a suitable light path or there is no resource for building up a light path. The text below shows the steps of light path configuration.

Step1. The rwa module sends out a LP-request packet. The value of option in optical header is 16.

Step2. The LP-request packet is broadcasted to the whole subnet, and each clone of the LP-request packet will record value in the packet information which contains the outgoing and incoming port and wavelength channel of each node.

Step3. When the LPC packet arrives at the destination router at the first time, the rwa module reserves the light path resources for the LP-request packet and the rwa module changes the LP-request packet to the LP-reply packet. After that the rwa module sends the LP-reply packet back along the path which it came along and reserves the resource for the light path at each switch.

Step4. When the LP-reply packet goes back to the source router which sent it, the rwa module of the router sends a LP-accomplished packet to the destination router along the path that the LP-request packet went on. The LP-accomplished packet will build the light path permanently.

If the resource reservation is failed during the travel of the LP-reply packet, the source router will send a failure massage to cancel the temporary reservation and send another LP-request packet another RWA request.

The dynamic RWA methodology is three-ways handshaking, and it will never stop until it has tried all the wavelengths we have. When the light path is built, the source router will add an entry in the LP table and the destination router will add an entry to the WA table (wavelength assigned table).

### 5.5.2. The Functions of the Wa Module

The main function of wa module in dynamic RWA scheme is reserving the wavelength channels and telling the osw module the information of the light path for building switching table. The text below is the behavior details of the wa module in dynamic RWA scheme.

I. When the wa module gets a LP-request packet, the wa module just broadcast it and sets up a flag to avoid duplicated packet caused by flooding.

II. When the wa module gets a LP-reply packet, it checks the reservation table to make sure that the required wavelength channel is occupied or not. If the wavelength channel is free, the wa module temporary reserves it. If the wavelength channel is occupied, the wa module change the flag of the LP-reply packet to “failure”.

III. When the wa module gets a LP-accomplished packet from source router, it reserves the temporarily-reserved wavelength permanently. The wa module tells the osw module which port and wavelength it reserved for building the switching table.

IV. The wa module may take wavelength conversion or no wavelength conversion, and it affects the result of the RWA. If the wavelength conversion is allowed, the required wavelength at this switch and that of the whole light path can be different wavelength channels. If the wavelength conversion is not allows, the required wavelength at this switch and that of the whole light path must be the same.

Figure 5.12 represents the work flow of the rwa module. Figure 5.13 represents the work flow of the wa module.

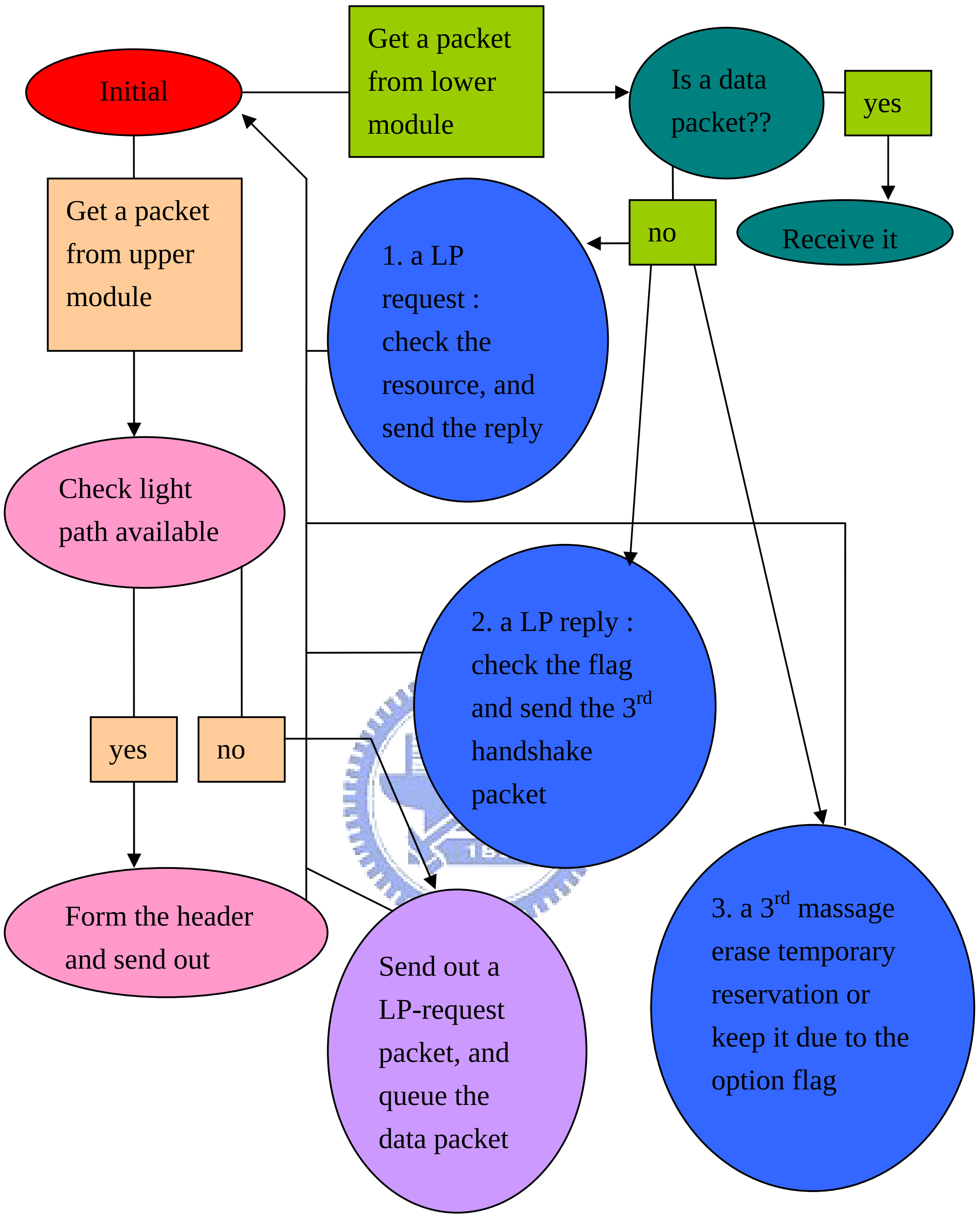

Initial
Get a packet from lower module
Is a data packet??
yes
Receive it
no
Get a packet from upper module
1. a LP request : check the resource, and send the reply
Check light path available
2. a LP reply : check the flag and send the 3rd handshake packet
yes
no
Form the header and send out
Send out a LP-request packet, and queue the data packet
3. a 3rd massage erase temporary reservation or keep it due to the option flag

Figure 5.12: The work flow of the rwa module.

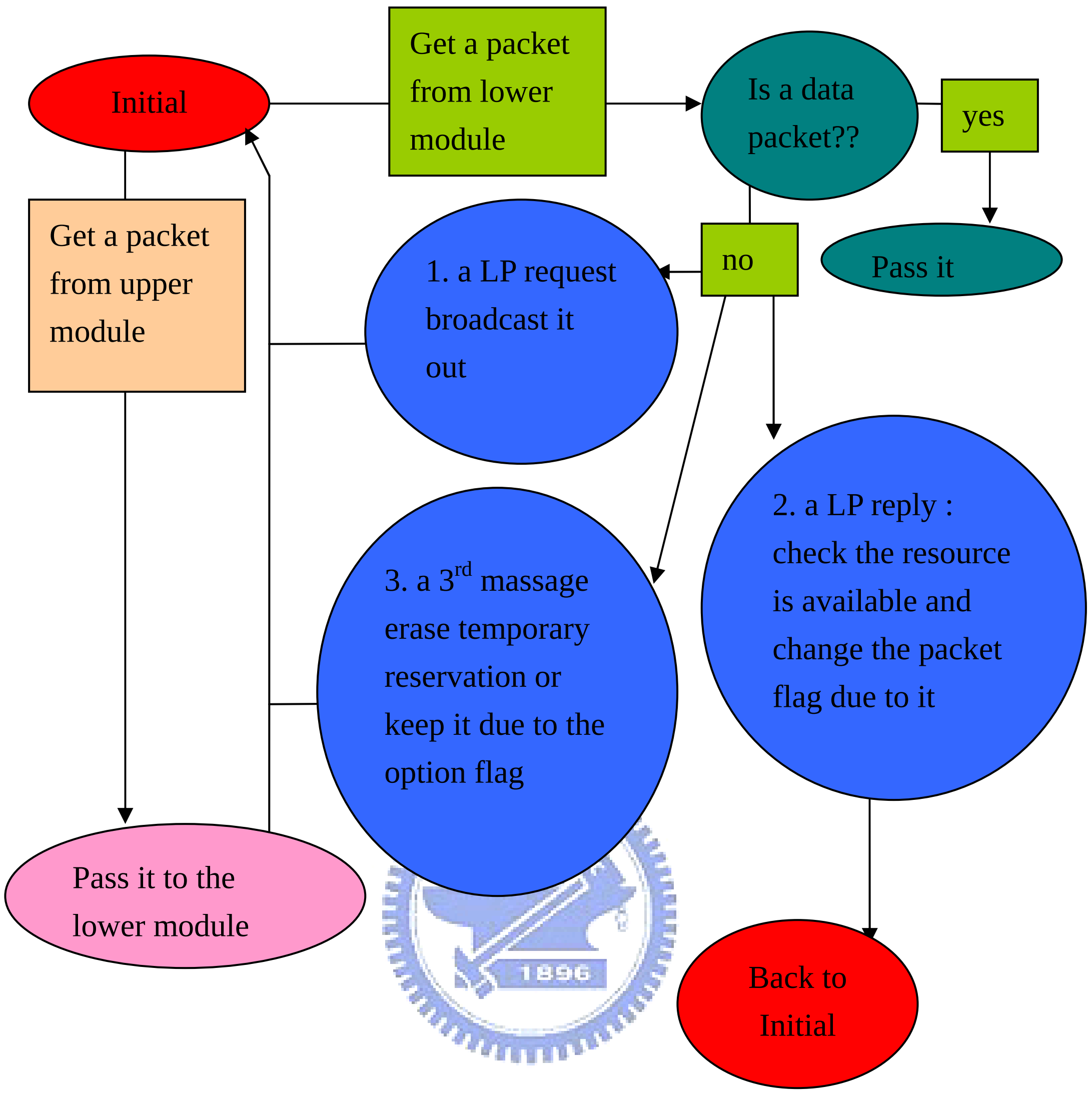

Initial
Get a packet from lower module
Is a data packet??
yes
Pass it
no
Get a packet from upper module
1. a LP request broadcast it out
2. a LP reply : check the resource is available and change the packet flag due to it
3. a 3rd massage erase temporary reservation or keep it due to the option flag
Pass it to the lower module
Back to Initial

Figure 5.13: The work flow of the wa module

### 5.5.3. Some Other Considerations about Traditional All-optical Networks

I. All-optical network is pure optical and it has no buffer, and therefore the maximum buffer length of the packet scheduling module in the optical switch node is forced to set to zero. The queue length of the default packet scheduling module of optical networks is zero at start.

II. In dynamic RWA scheme, the light path is not built permanently. It will be torn down when the timer expired or the light path is rarely-used. Of course, the connection will be built again when the rwa module has packets to send. There is parameter to decide the maximum life time of a light path.

III. The RWA can be done by users or by the shortest path RWA tools. It can also be done dynamically by the rwa module and the wa module.

IV. The optical system type of the same subnet must be the same. For example, the number of wavelength channels, network types, and switch types of nodes in the same subnet must be the same.

V. The host node can not connect to the optical switches directly. Only routers and the same type of optical switches can be connected to each other.

## 5.6. Optical Burst Switching Networks

Optical burst switching is a new paradigm for all-optical networks [4]. It provides new network architecture, and it combines the advantages of circuit switching (traditional all-optical networks) and packet switching. In this new architecture it is easy to handle the management and the quality of service which are the advantages of circuit switching. It also reaches the high bandwidth and resources utilization and this feature is the advantage of packet switching. By the way, the reservation schemes and the algorithms of choosing delay factors are various and it is hard to choose the appropriate strategies. Therefore this becomes a very important research area in all-optical networks. Our optical burst switching networks has two main modules. The obwa module is in routers, and the obsw module is in optical burst switches. The detail functions of the two modules will be described below.

### 5.6.1. The Obwa Module

- The main functions of this module are assembling burst, generating control packet, and assigning wavelength channel.
- When packets come into this module, the packets will be queued at the burst queue. The module will not send the packets at once, but it will gather more packets to compose a burst.
- We have to create a mechanism to send the bursts. We define 2 conditions to send out bursts. The burst is the aggregation of the packets in the burst queue. One condition is that the queue length reaches the limit that is set by users. The other condition is that the queuing timer is expired. When the first packet of the burst comes, the obwa module triggers a timer. When the timer expires, no matter how small this burst is, the obwa module sends the burst out. We use the timer method for making sure that the traffic will not die in the networks if the traffic is sparse, or making sure that TCP ACK and SYNC packets will always be sent out on time. The length method and the timer method can be used alternatively to gain

performance and utilization [insert ref].

- When the burst is ready to be sent out, the first thing is creating a control pack of this burst. We know that the transmission admission and priority of a burst in the optical burst switching networks are based on the configure result between switches and the control packet of this burst, and therefore we generate the control packet. A control packet contains such information:

  I. Burst ID represents the burst.

  II. Burst transmission time presents how long this burst transmits.

  III. Burst arriving time shows when the burst will arrive.

  IV. Wavelength contains what channel the data burst takes.

  V. The start router of the burst

  VI. The end router of the burst

  The information is extracted by the optical burst switches and the switches can find that there is contention among bursts or not. After that the optical burst switches can decide to reserve the resources for this burst or not.

- After sending the control packet, the burst is pushed to lower module. There is a delay factor between the control packet and the burst. Because the longer the delay time is and the probability that the burst gets the resources gets higher, the delay factor is important. For JET scheme [5], if the delay time is too short, the burst may arrive at switches earlier than control packet and the burst will be dropped.

- The obwa module also has to decide which wavelength channel the traffic takes. The obwa module will call the static RWA function to give a light path to it. Only static wavelength assignment is available in optical burst switching networks.

- This paragraph describes the design considerations of the obwa module.

I. The burst can be a period of time of transmission or aggregation of packets. We "gather" the burst, but we do not merge the packets together and encapsulate a new header to construct a burst. We only send these packets continuously and each packet has a tag of burst ID. Assuming each packet transmits for 100 micro seconds and we define 10 packets to be a burst. If the burst arrive at the 2000th micro second, the burst mean "the packets arrive at 2000~2100 micro second" from the view point of time. The burst means "the packets ID 1~10 or a burst data" from the view point of packet. The Old conception only treats bursts as the combination of packets, and the contention problem makes the all packets dropped. The time concept reduces the drop unit from "contented bursts" to "contented packets (time)", which is the concept of burst segmentation [6].

II. The control packet has its own wavelength channel in the design of the devices. This is because that the control packet needs to take OEO processing and it has to take different transponder to transmit it. The receiver of the control channel transforms the optical signal, and the transponder of the control channel transforms the electronic signal to optical signal. Those of the data burst channel are all-optical and they handle only optical signals. So we design our module that control packets are sent on their own wave length channels. The transmission of the control packet follows store-and-forward scheme, which we discussed at 6.2.2, sectionV.

III. We have to set the queue length of the FIFO module below the obwa module in the router to zero. In traditional all-optical networks, the queue length of the FIFO modules below the rwa module can be nonzero. This is because the data burst needs accurate timing to transmit and it can not be delayed at the fifo queue. If the FIFO queue length is not zero, the burst may be queued and delayed at the FIFO module for a certain time. The burst will miss the period of time during which the reservation is valid. For example, the control packet reserves the 2000th to 2100th micro second for its burst and the FIFO queue length is not zero. The burst may be queued in the FIFO and delayed for a while, and the arriving time to the switch of the burst may not be the same to the time the control packet reserved. The most of the burst may be dropped.

IV. The delay factor between control packet and burst is important. This is because we have to confirm that the control packet will arrive at the end router before the burst. We know the transmission time pf the control packet contains OEO time, store-and-forward time, and packet processing time. Even if the control packet goes earlier, it will reach the end router later if the delay factor is too small. Therefore we provide some parameters that can help finding the appropriate delay factor. They are packet process time, OEO time, and store-and forward time. These 3 parameters are set on the obsw module and it uses a global way to share these parameters with the obwa module. Propagation delay and hop counts are provided by the ophy module and the static RWA object.

V. The RWA scheme in obwa module is rather different to that in the rwa module. In rwa module, the RWA object gives the route a light path and assigns the incoming / outgoing port and incoming / outgoing wavelength. The packet follows the path to the destination router. However, in obwa module, the RWA object only provides the incoming and outgoing port, because the resources (wavelengths) need to be reserved before using them and the wavelengths are assigned temporarily. The text below describes the method of the wavelength assignment. When the wavelength channel A is busy during the required reservation period, the module tries to assign wavelength channel B. if wavelength channel B is not available during that time, it tries to assign wavelength channel C. the checking loop continues until that the module finds that all the data burst wavelength channels are occupied.

VI. Each light path has its own burst queue. For example, there are 3 routers in a simulation network. There are 2 assigned light paths "router1→ router2" and "router1→router3". Each light path has its own burst queue and the router1 has 2 burst queues. Because the packets may take different light path, we can not merge all the incoming packets into only one burst queue.

### 5.6.2. The Obsw Module

- The obsw module is in the optical burst switches. Its main functions are deciding the burst reservation and handling burst transfer controlling.
- The control packet tells the obsw module when and how the burst comes. The information in the control packet contains when the burst comes, what light path the burst takes, and how long the burst will transmit. The

module will decide to reserve the light path for the burst or not according to the information that control packet provides.

- The obsw module has a table called reservation table. This table contains the data of the wavelength status of each optical port. When a control packet comes, the obsw module reads the control packet to know the needed resources that the burst asks for, and the obsw module checks the reservation table if there is contention or not. If there are free wavelength channels at that burst period, the obsw module reserves it and adds the current reservation to the table. If the required wavelength channel is occupied, it rejects the reservation request (original method) or offers the non-contented time of this wavelength channel (burst segmentation). After the control packet being processed, the obsw module sends the control packet to the next node.
- When the obsw module receives an incoming burst packet, it checks the arriving time and the burst ID of the packet. The obsw module searches the reservation table for the entry of the burst. If there is no such burst entry, it means that the burst is not allowed to use the resource to transmit. The obsw module drops the packet. If the obsw module finds an entry contains the burst ID but the packet incoming time does not fall on the reservation period, it means that the packet comes too late or too soon and the resource may be used by other bursts. Only the burst packet that has the mapped reservation entry and arrives at accurate time can be transmitted to the next node. The job when the obsw module receives outgoing packets from the osw module is the same.
- This paragraph lists the design considerations for obsw module.

  I. We need to simulate a special processing for control packets. The control packet spends more time on OEO processing, store-and-forward transmitting, and computation in the optical burst switches. The OEO processing time and store-and-forward features are handled by the ophy module, and therefore the only factor that the obsw module has to simulate is the computation time in electronic domain. The obsw module has a parameter to represent the factor and has a timer to simulate the computation of each control packet.

II. Because the reservation and burst dropping algorithms are various and this field is one of the most popular research areas in optical burst switching networks, we implement several simple methods and reserve some frames of header and empty functions for the researchers to develop their own algorithms. For example, we only implement burst segmentation for dropping algorithms. We implement only random and first-come-first-serve methods for reservation algorithms. In the future we can add priority features and some prediction patterns for supporting QoS or developing flexible and low-drop-rate algorithms.

III. The jobs of RWA in the obsw module are a little different to traditional all-optical networks. The wavelength channel is assigned dynamically in the router, and the control packet asks for the same wavelength channel which is given in the router. This is because the default value of wavelength conversion is “No” in the obsw module. If the required wavelength channel is not free, the contention happens and the late-arriving or low-priority packets are dropped. If the value of wavelength conversion is “Yes”, the obsw module can assign another wavelength channel to the burst and the data channel of this burst has to be changed to the current assignment.

### 5.6.3. The Work Flow of the Optical Burst Switching Networks

- At the start of simulation, the obwa module gets the value of control packet processing time from the obsw module, and this factor can be used as one of the burst assembling considerations.
- When the obwa module receives an incoming packet from the upper module, it means that the packet comes from the interface module and needs to be sent out. The obwa module calls the function "rt_gateway()" to find the next-hop router interface and it puts the packet into the burst queue of this route.
- When the burst length is long enough or the burst gathering timer expires, the bust is formed and it is ready to be sent. The obwa module generates a control packet of this burst and the obwa module sends it out. The RWA object here gives the obwa module the destined outgoing port and the hop count of the light path. By feeding the control packet processing time, OEO time, and the hop counts, the delay factor can be computed. The obwa module sends out the burst later for a period of time that the delay factor represents.
- When the control packet reaches the obsw module in the optical burst switch, the obsw module reads the information in the control packet and checks the resources reservation table. If the information matches the conditions and rules of the reservation, the obsw module schedules the burst of this control packet into its reservation table. If the required resources are not available, the control packet will be dropped. The

reservation rules can be user-defined or default.

- After the delay the data burst is sent out from the obwa module. When the packets of the burst arrive at the obsw module, the obsw module checks the reservation table and figures out that the packets are allowed to transmit or not. If the answer is yes, the obsw module passes them to the osw module to do the switching. Otherwise the obsw module drops them.
- The burst will reaches the destination if all of the reservations are successful and the burst is not dropped by the optical burst switches.

The picture below shows the optical burst switching networks on the NCTUns:

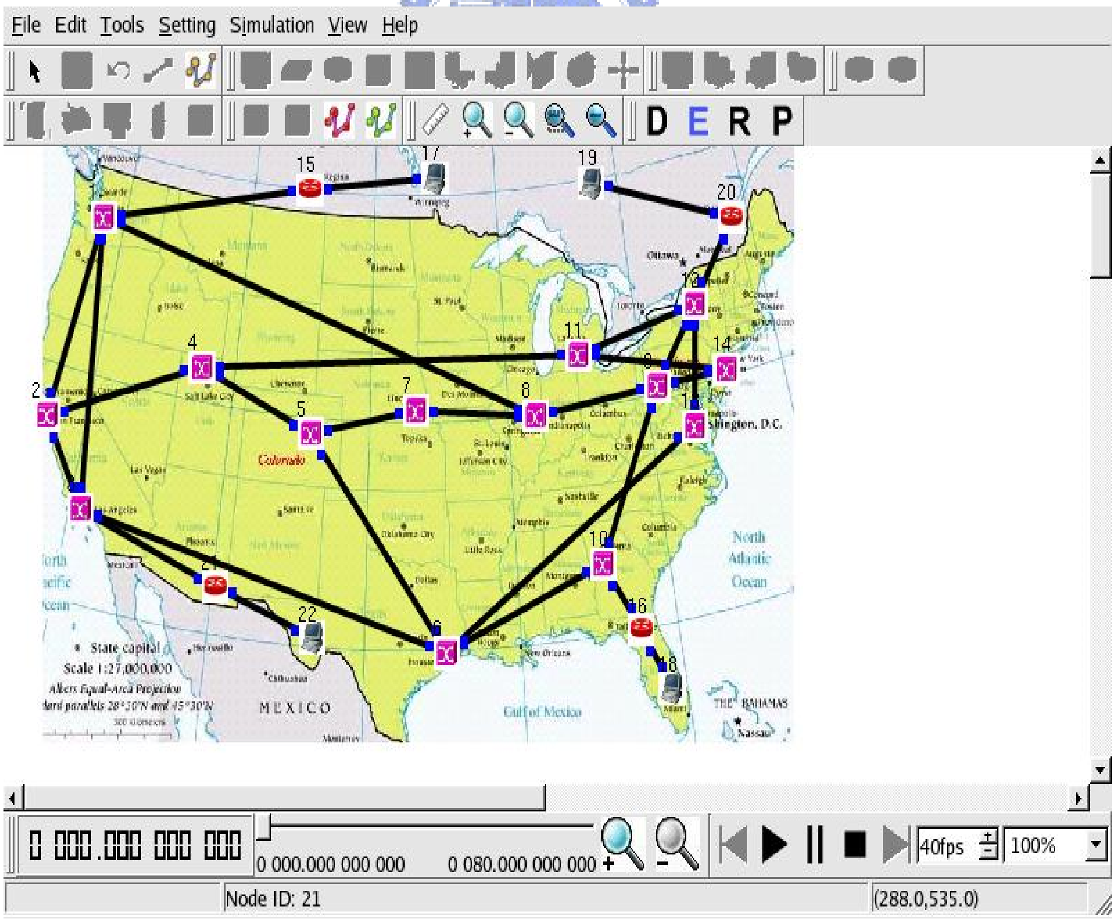


Figure 5.14: The example case of optical burst switching network

## 5.7. The Design and Integration of the GUI for All-optical Networks

The GUI of the NCTUns is an interface between simulation engine and users. It helps users building their simulation topology and setting the parameter values. It also helps users monitoring and controlling the simulation, and it analyzes and illustrates the performance and behavior data after simulation. Because we add the all-optical network simulation package, we have to add some functions of the GUI to support our system.

- We add the 2 new node types of traditional all-optical switch and optical burst switch to the node tool bar. The color of traditional all-optical switch is gray and the color of optical burst switch is pink.
- We define new module trees of our all-optical networks. After drawing the network topology, the GUI will transform the topology to the set of modules. Therefore we add new module trees for generating the tcl content for the optical switches and the routers that connect to optical networks.
- We provide the new mdf (module definition file) containing our new modules. The mdf contains is the file that provides the GUI the details of module information. We add our new modules to the mdf and the GUI can identify our new modules.
- We add a new option on the pop-up menu, and we can decide the number of wavelength channel. Users can decide how many wavelengths the system has by their own.
- We add protection ring and light path setup utility on the tool bar. After pressing the light path setup bottom, we choose a router as the head of the light path, several optical switches as the body of the light path, and a

router as the end of the light path. That is how we set up a light path. Of course these selected nodes must be connected and the optical switches must be the same type. After pressing the protection ring setup bottom, we choose several optical switches to form a protection ring. Of course the selected optical switches must be connected, the same type, and their selecting sequences must be circular.

- The animation playback now can display the format of the packet trace file of the optical networks, and it can render the traffic flow animation of the optical networks. We also add a useful tool to specify the display of wavelength channels, and then we can watch the traffic flow on the specified wavelength channel. If the GUI shows all the traffic flow on all wavelength channels, the screen will be a mass and the simulation result is not easy to be identified.

## 5.8. Modifications of the NCTUns

The most challenge to develop our system on the NCTUns is the syntax limitation of the tcl file. For providing multi-port and multi-wavelength channel simulation environment, we can find that the module trees of the new-added nodes have multi-level branches. However, the old tcl parser does not support such structure. The old module tree has only one level branch. The engine itself can support multi-level branches of the module trees in a node, but the simulation data of our system can not be translated from tcl file to the engine data structure unless the tcl parser supports it. The old tcl parser can identify only one level branch and it will ignore other levels of branches. The tcl file of our system also may cuase parsing error if we use the old tcl parser.

This is because the old tcl parser does not support net-structured syntax like this:

```
Module test
    Define port 1
        Module test1
            Define port 1
                …
            EndDefine
        …
    EndDefine
```

We modify the tcl parser and several data structures, variables, objects for supporting multi-branches architecture.

- We enhance the tcl parser, and now it can identify the net declaring of "Define port". The parser can parse the tcl file of our system to the multi-branches data structure of the simulation engine.
- We modify the basic module object named "NslObject". One of its constructor parameter is the port number that represents under which port this module lies. This parameter is now expanded to a "traversing list". For example, a switch module of the Ethernet switch has only one level branch, and the port number of the phy module under branch 4 is 4. If we just use the old object structure containing only the nearest level of branch as the port number, some identification of the objects and the relationships among objects will be confused. So we have to change this parameter of the object for helping us obtaining the whole traversing list. For example,

if the ophy module is in the port2 of the switch and wavelength channel4 of port2, the port traversing list is (2, 3) and the tcl parser passes these parameters to construct a module object.

- We change the way of the sorting, searching, and registering of variables. Variable registry is for inter-module communication in a node. Each module object is independent, and it has no idea what the other modules do and what information the other modules have. If we want to share a variable among many modules, we have to register this variable. For example, the obwa module wants to know the bandwidth and the propagation delay of the wavelength channel, but the information is stored at ophy module. Therefore we register “BW” and “prop_delay” at the ophy module to share these variables among modules in the node. This is how the module communicates with other modules in a node. The searching key of the registered variables in the old engine is port number and variable name. Because we change the type of port number, we have to change the way using the port number as a search key. The whole port traversing list has to be compared.
- Also we have to care about the capability of the old syntax and old module. A lot of modules use old syntax and structure, and therefore our modified engine has to support them. The multi-level branches syntax and structure are the generalized expansion of single branch syntax. Obviously the syntax is compatible to old module and the port ID is a kind of traversal list that contains only one member. The only things we have to care about are the APIs, functions and macros. We have to fix the name and the parameters to match our new APIs. We check all the modules and substitute new functions for old functions.

- Besides the new syntax and data structure, the new storage format is needed. The packet trace file for animation playing and the logging function of the packet trace file has to be modified for supporting all-optical networks. In old version of the NCTUns, the packet trace file format contains only 802.3 the Ethernet and 802.11 the Wireless LAN. All-optical networks have additional information to present in the packet trace file. For example, optical networks has multiple wavelength channel as wireless LAN, but it has the wired LAN features that each channel can send data in parallel. Optical network has light path configuration packet for traditional all-optical networks and control packet for optical burst switching networks. Therefore we add the new defined macro of these optical packet types, and new field for wave length channel in the packet trace file. We also modify the PrintPtr program so that it can parse and print the new format of the packet trace files of optical networks.

# 6. Performance Evaluation

We will discuss the performance of our system in this chapter. We list several results and analyze the performance data. We show that our system design is reasonable and efficient according to the analysis result.

## 6.1. System Information of Our Experiment Platform

CPU: Pentium Celeron 2.4GHz

Memory: 512MB DDR RAM with clock rate 400 MHz

OS: Linux, Fedora Core 1 with kernel 2.4.22

The simulation needs only one computer, and therefore the bandwidth of the network interface card is not in the consideration. The factors that affect the simulation speed are the CPU clock rate and memory size and clock rate. We use Linux as our operating system.

The Figure 6.1 represents the topology of our simulation case.

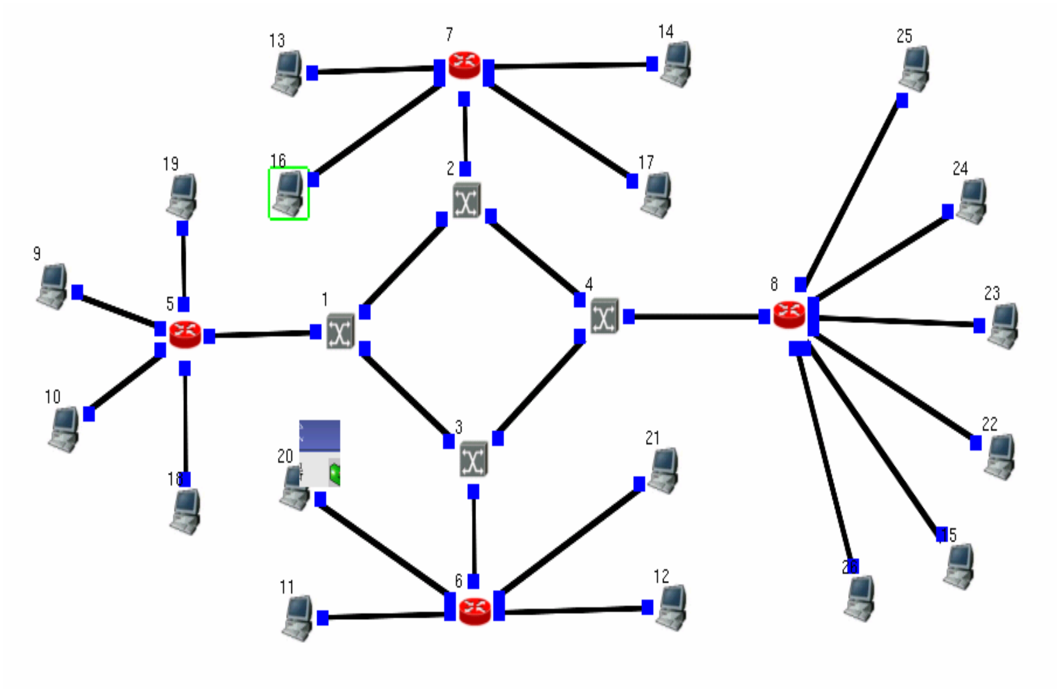


Figure 6.1: The case of performance evaluation

The red icon represents routers, and the gray small box with ‘X’ on it represents the traditional all-optical switch. The icon which looks like a computer represents the end host.

There are three TCP connections in the “3 TCP connections” case. They are connection from node21 to node25, connection from node19 to node24, and connection from node13 to node23.

There are four TCP connections in the “4 TCP connections” case. They are connection from node21 to node25, connection from node19 to node24, connection from node13 to node23, and connection from node12 to node22. There are five TCP connections in the “5 TCP connections” case. They are connection from node21 to node25, connection from node19 to node24, connection from node13 to node23, connection from node12 to node22, and connection from node9 to node15.

There are six TCP connections in the "6 TCP connections" case. They are connection from node21 to node25, connection from node19 to node24, connection from node13 to node23, connection from node12 to node22, connection from node9 to node15, and connection from node16 to node26.

The factors that changes in our simulation cases are:

- The number of wavelength channel
- The quantity of traffic flow
- The bandwidth of each wavelength channel
- The operation status of packet trace log
- The network type

Because NCTUns is an event-triggered network simulator, more events in a simulation case needs more system resources and time to complete simulation. We can predict that the simulation takes more time to finish and it needs more memory space and CPU cycles if the factors we listed above become larger. Also we want to do some experiments and we analyze the growth of time and space that simulation takes. We will prove that the performance data is reasonable. In this simulation case we use the static RWA scheme and no wavelength conversion scheme. The packet trace log function needs a lot of IO time and a lot of CPU cycles, and therefore we only do some comparison to make sure the additional time needed is in reasonable range. The virtual tick is set to 10 nano second per tick, and the length of simulation time is 10 second (virtual time). The bandwidth of each link between host and router of Ethernet is 100 Mbps, and the data rate of greedy TCP connection will be about 70 to 80 Mbps.

The tables below shows the experiments result.

| 6 greedy TCP, channel bandwidth is 1000 Mbps, trace log off, all opt traditional | | | |
|---|---|---|---|
| 3 wavelengths | 8240K RAM take | 245 sec to finish | 76.5 Mbps / each |
| 4 wavelengths | 8240K RAM take | 243 sec to finish | 77.0 Mbps /each |
| 5 wavelengths | 8241K RAM take | 242 sec to finish | 73.0 Mbps /each |
| 6 wavelengths | 8241K RAM take | 247 sec to finish | 75.4 Mbps /each |

Figure 6.2: The changing factor is wavelength number.

| 6 greedy TCP, 3 wavelength channels, trace log off, all opt traditional | | | |
|---|---|---|---|
| 100 Mbps / wave | 8021K RAM take | 151 sec to finish | 41.7 Mbps / each |
| 200 Mbps / wave | 8239K RAM take | 249 sec to finish | 78.1 Mbps /each |
| 500 Mbps / wave | 8240K RAM take | 247 sec to finish | 77.2 Mbps /each |
| 1000 Mbps /wave | 8240K RAM take | 242 sec to finish | 74.4 Mbps /each |

Figure 6.3: The changing factor is the bandwidth of optical wavelength channel.

| 3 wavelengths, channel bandwidth is 1000 Mbps, trace log off, all opt traditional | | | |
|---|---|---|---|
| 3 TCP connection | 8020K RAM take | 98 sec to finish | 74.5 Mbps / each |
| 4 TCP connection | 8088K RAM take | 143 sec to finish | 71.3 Mbps /each |
| 5 TCP connection | 8163K RAM take | 189 sec to finish | 78.0 Mbps /each |
| 6 TCP connection | 8240K RAM take | 240 sec to finish | 76.0 Mbps /each |

Figure 6.4: The changing factor is the traffic quantity.

| 3 wavelengths with 1000Mbps each, all opt traditional, 6 TCP connections | | | |
|---|---|---|---|
| Trace log off | 8241K RAM take | 241 sec to finish | 77.6 Mbps / each |
| Trace log on | 26525K RAM take | 671 sec to finish | 78.0 Mbps / each |

Figure 6.5: The changing factor is the status of packet trace log.

| 3 wavelengths with 1000Mbps each, all opt traditional, 6 TCP connections, log off | | | |
|---|---|---|---|
| Simulate 10 sec | 8239K RAM take | 241 sec to finish | 77.6 Mbps / each |
| Simulate 12 sec | 8241 K RAM take | 289 sec to finish | 76.0 Mbps / each |
| Simulate 14 sec | 8240 K RAM take | 327 sec to finish | 79.9 Mbps / each |
| Simulate 16 sec | 8240 K RAM take | 380 sec to finish | 76.2 Mbps / each |

Figure 6.6: The relationship between the real time needed and the simulated time

## 6.2. Analysis

From the view point of the NCTUns, the requirement of the memory space depends on the number of events in the simulation engine. Taking a look at figure 6.2, the number of wavelength channels does not cost much memory size. This is because the module object size is not big as the space needed when the events comes in, and the growth of wavelength channel number has nothing to do with the growth of event quantity in engine.

Taking a look at figure 6.3, the time needed column in the row “100 Mbps each wave” is much smaller than those in other rows. This is because the 6 TCP connections are given only about half bandwidth. The events are half and the needed time is in direct ratio to the event number. The number of events in the engine in a certain time is not large (it is according to the propagation delay), and therefore the column of memory space is not so affected as the column of executing time. The same reason explains the data in the figure 6.4. The data in figure 6.6 shows that the time needed is linear to the simulation time. The growth is linear and it is the same as that of other simulation cases in the NCTUns. The data in figure 6.5 shows that the packet trace log needs a lot of resources. The packet trace log generates log events and some stack overhead, and it costs a lot of IO time and CPU cycle. The memory space and time needed are much less if we turn the packet trace log off.

We have proved that our system design and implementation does not decrease the original performance of NCTUns in traditional all-optical networks.

The cases in the optical burst switching networks are similar. The only difference is the traffic behavior is not the same as that in the traditional all-optical networks. However it is the problem of functionality, and it is related to the network behavior. Therefore it is not in the consideration of the performance evaluation. Besides, the performance analysis result is the same as that in traditional all-optical networks.

I. The executing time needed is linear to the traffic quantity (average speed * connections).
II. The time needed is linear to the simulation length
III. The number of wavelengths is independent to the memory space and executing time.
IV. The system does not affect the original performance result of the other cases in the NCTUns.

## 6.3. Scalability Test

Looking at Figure 6.1, we want to test that if the numbers of nodes becomes large, the performance can be still the same or it is affected heavily. We add two routers and eight nodes at each optical switch. We add total 40 nodes to expand this simulation case. We find that the simulation time is still the same (with difference about one to three seconds), and therefore we know that the number of nodes does not affect the simulation time. The memory space used by simulation engine is increased to 9000K byte around, and we can explain that the increased memory space is taken by the added nodes in this case.

# 7. Functionality Validation

We will discuss the analysis of the network behaviors of our all-optical system in this chapter.

## 7.1. The Validation Analysis of the Ophy Module

We have to prove that the simulation of propagation delay, bandwidth, store-and-forward scheme, and the circuit and packet switching schemes are correct. We use the network topology below as our study case.

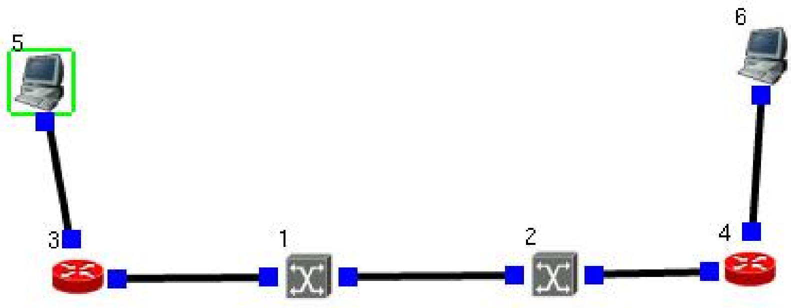


Figure 7.1: The case of validation of the ophy module

This case tests the validation of the ophy module, and therefore we use a rather simple case. There is only one wavelength channel in this case, and we set the bandwidth of the wavelength channel to 20Mbps for each optical link. The Ethernet links are set to 30Mbps, and we can see that the traffic will be bound by the optical link. If the traffic between node5 and node6 takes greedy scheme, the data flow will be bound at optical link and the data rate is about 20Mbps at most. We use this trick to prove that the bandwidth simulation in the ophy module is valid.

We set the propagation delay of each optical link to 100 micro second, and we can predict that if the packet size is 1000 byte, the time that transmit from router3 to router 4 is (8000 / 20M + 3*100) micro seconds. This is because the non-store-and-forward scheme makes the path “3→1→2→4” like a direct link from node3 to node4 with propagation delay of 300 micro seconds. The graph below shows the throughput of one greedy TCP connection between node5 and node6.

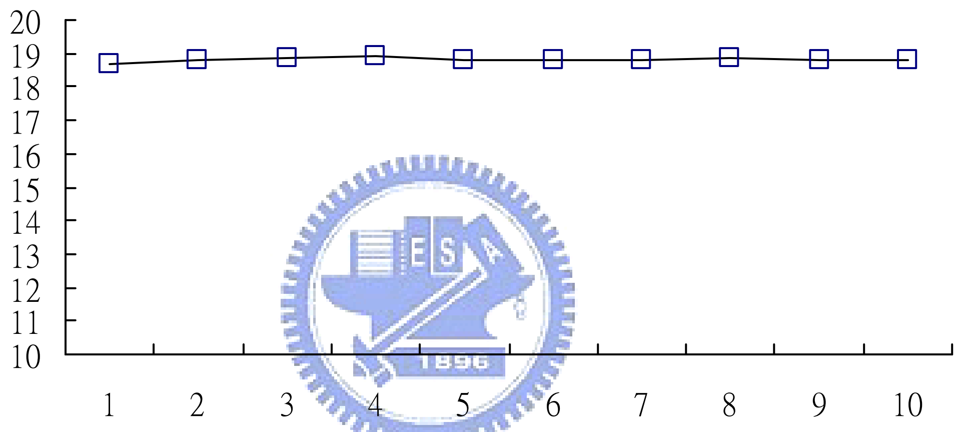
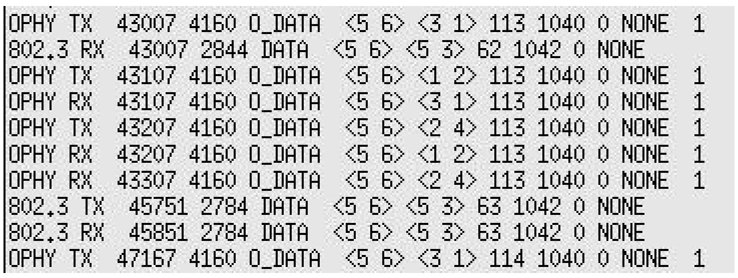


Figure 7.2: The simulation result of the ophy module

The X axis represents the time and unit is second. The Y axis represents the throughput and unit is mega bits per second. The traffic pattern is greedy UDP connection from node 5 to node 6.

From this graph we can see the throughput is about 18.8 Mbps. The bandwidth of the optical link is 20 Mbps. The range between these 2 is due to the overhead of Ethernet frame and IP header. The random back-off mechanism will gain the delay time of each packet from 1 tick to its 1/10 transmission time and therefore the bandwidth will be wasted about 1/10. From the data analysis above we prove that the simulating of bandwidth is correct.

We now take a look at the packet trace file to observe behavior of the traffic flow. The graphic below is the capture of the packet trace viewer.


```
OPHY TX  43007 4160 O_DATA  <5 6> <3 1> 113 1040 0 NONE  1
802.3 RX  43007 2844 DATA  <5 6> <5 3> 62 1042 0 NONE
OPHY TX  43107 4160 O_DATA  <5 6> <1 2> 113 1040 0 NONE  1
OPHY RX  43107 4160 O_DATA  <5 6> <3 1> 113 1040 0 NONE  1
OPHY TX  43207 4160 O_DATA  <5 6> <2 4> 113 1040 0 NONE  1
OPHY RX  43207 4160 O_DATA  <5 6> <1 2> 113 1040 0 NONE  1
OPHY RX  43307 4160 O_DATA  <5 6> <2 4> 113 1040 0 NONE  1
802.3 TX  45751 2784 DATA  <5 6> <5 3> 63 1042 0 NONE
802.3 RX  45851 2784 DATA  <5 6> <5 3> 63 1042 0 NONE
OPHY TX  47167 4160 O_DATA  <5 6> <3 1> 114 1040 0 NONE  1
```
Figure 7.3: The capture of the packet trace

Here we introduce the packet trace format quickly. The third field is the time point that this packet is sent or received, and the 7th field is the source and destination node of the current hop. From this capture we can find that the time of the first bit of the packet 113 being sent from node3 is 43007th tick, and the time of the first bit of the packet 113 being receive by node 4 is 43307th tick. The time that the travel takes from node3 to node4 is 300 ticks (43307 - 43007). Because one tick is equal to100 nano seconds, 300 ticks is equal to 30 micro second. The result fits our prediction. Now we take a look at the last row of this picture. The packet ID 114 was sent from node3 at 47167th tick, and the packet ID 113 was sent from the same ophy module at 43007th tick, their range is 4160 which is equal to (1040*8/20)/0.1 (the unit is tick) fits the transmission time of the packet. The validation of simulating propagation delay, bandwidth and transmission schemes are all proved

## 7.2. The Validation Analysis of Protection Ring Behaviors

In this section we take a mesh ring protection case for an example. The graphic below shows the topology of our simulation case.

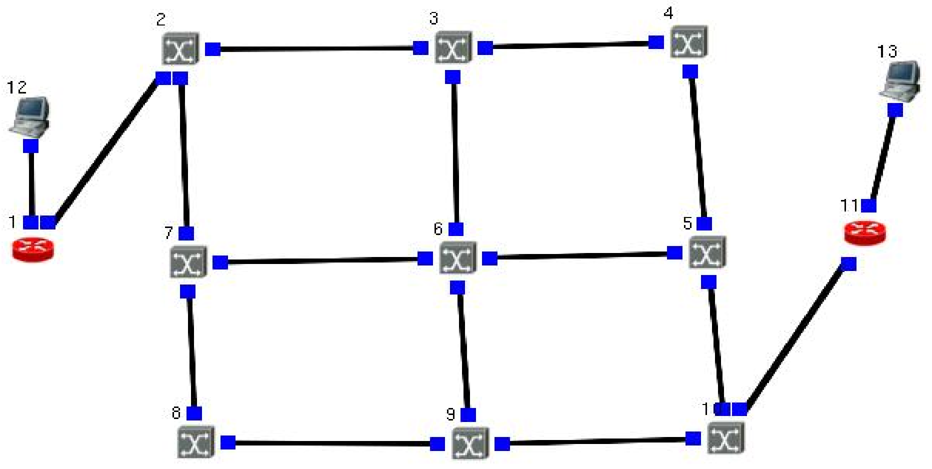


Figure 7.4: The case of validation of the protection mechanism

The ring assignment is that “2→3→6→7” is a ring, “3→4→5→6” is a ring, “7->6→9→8” is a ring, and “6→5→10→9” is a ring. The link fail will be at link “4→5” during 3rd to 5th second, and the TCP traffic from node12 to node13 will take “1→2→3→4→5→10→11” as its light path. When the simulation starts, the traffic flow on the networks looks like the picture below during 1st ~3rd second.

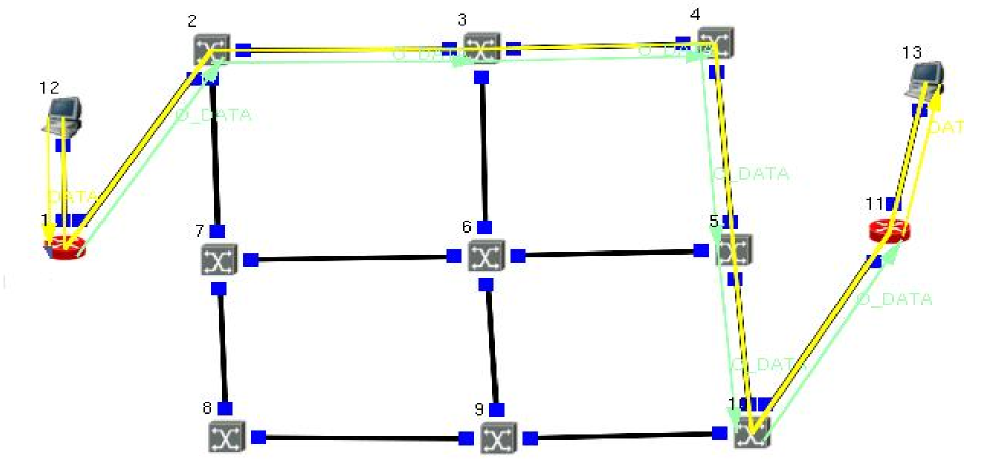


Figure 7.5: The normal situation of the traffic flow

When the 3rd second comes and the link “4→5” crashes, nodes senses that the working path of this ring is broken and it commands the traffic to take another route. The three pictures below shows the progress of the protection switching.

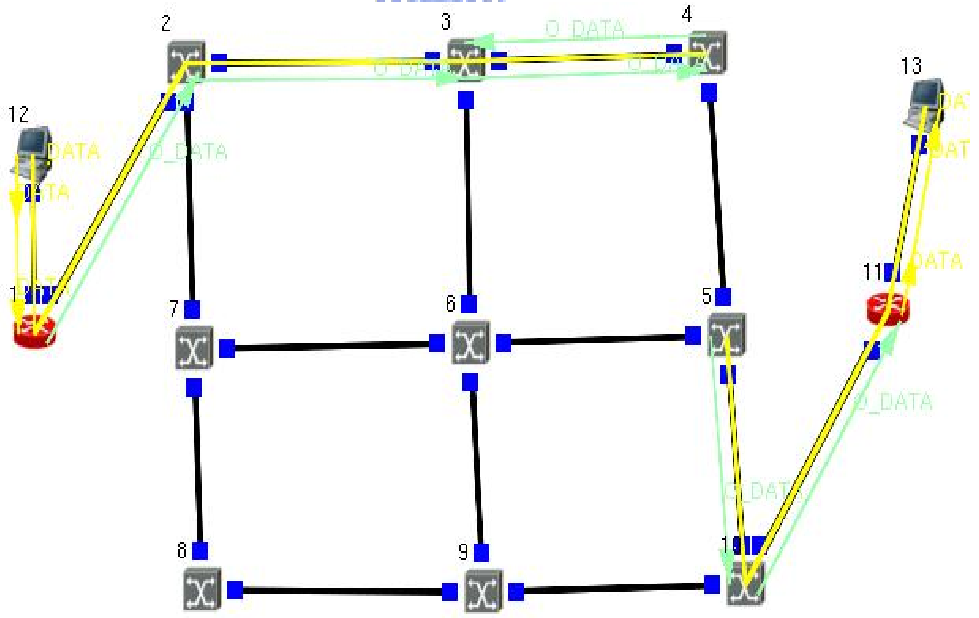


Figure 7.6: Step1 when protection activates

Step1. Link 4→5 is broken and the traffic take protection path. We can see the traffic is switched back to node3

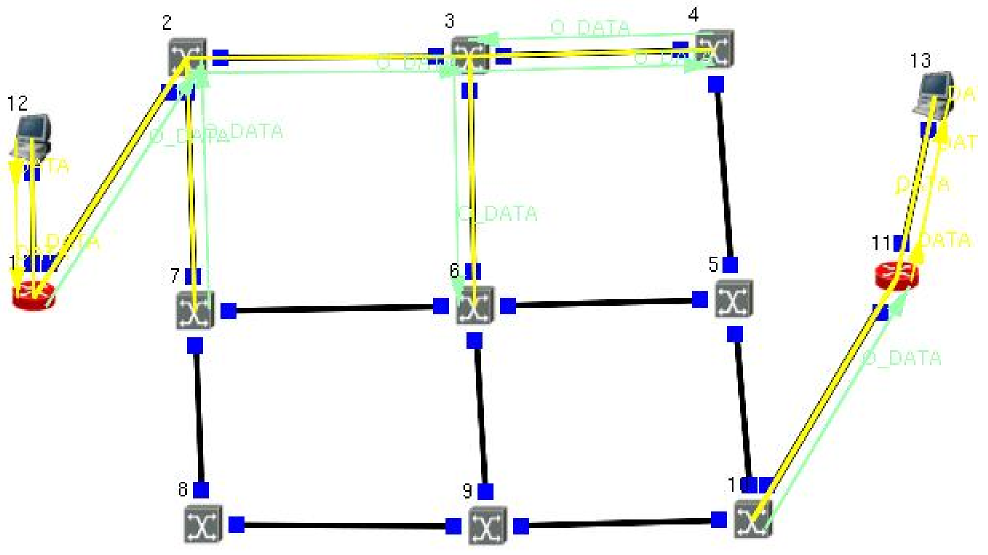


Figure 7.7: Step2 when protection activates

Step2. The traffic follows the protection path to node 6. The direction of protection path is "3→6→5→4".

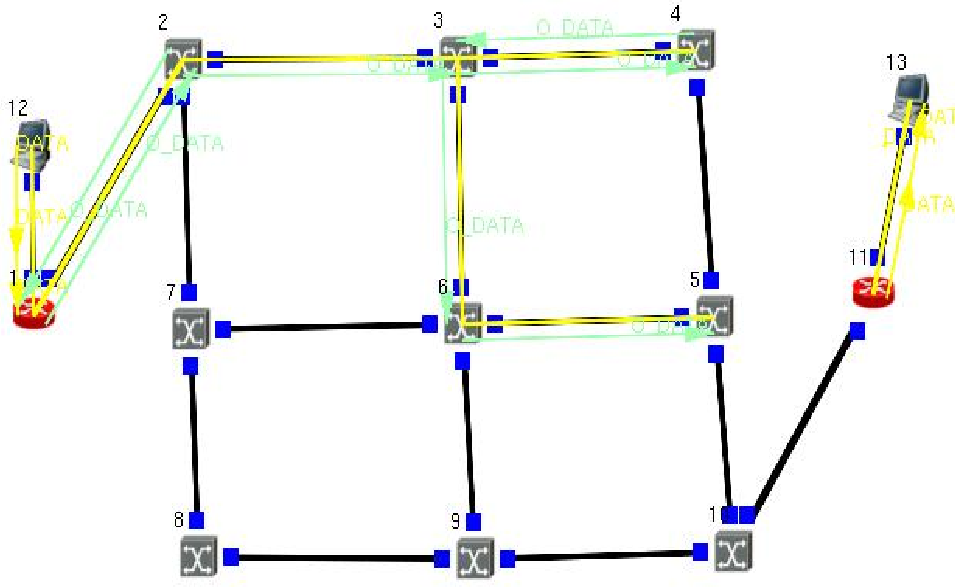


Figure 7.8: Step3 when protection activates

Step3. The data flow comes to node 5, and it finds the next node is node 4. Because node5 is the final destination of the protection path, we have to find another virtual ring to continue the journey.

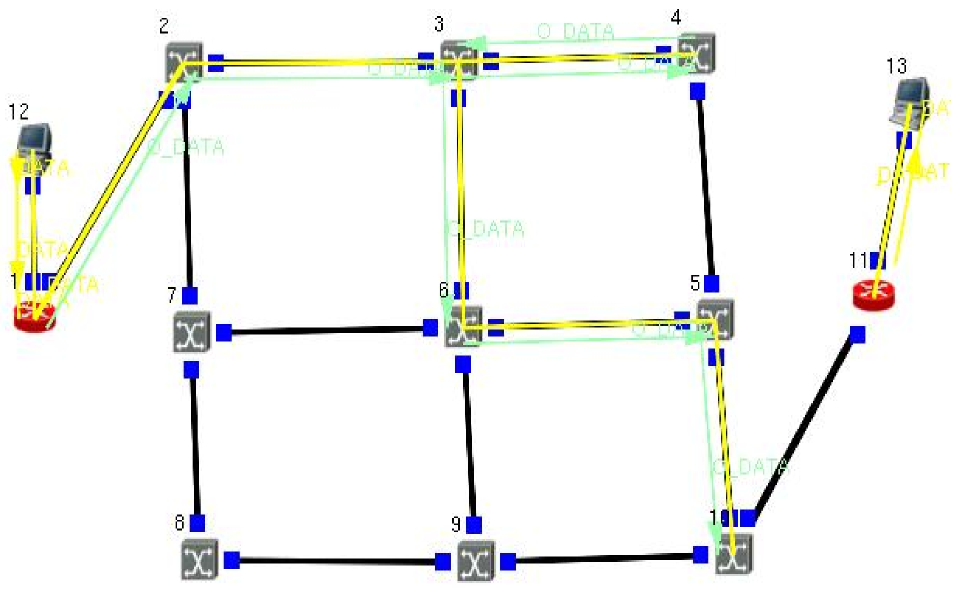


Figure 7.9: Step4 when protection activates

Step4. We find that node 5 joins another ring and one of this ring's (the ring is "6→5→10→9→6") working path (it is link "5→10") is on the light path from router 1 to router 11, and therefore node5 switched the traffic to node 10 and the traffic will take the original light path to the destination.

We find that the virtual ring protection is successful, and the ring protection is one special case of the mesh virtual ring protection. The behaviors of the ring protection are the same as we respected, and therefore this function is valid.

## 7.3. The Validation Analysis of the Optical Burst Switching Networks

The special behavior of the optical burst switching networks is burst contention and the behavior changes with various scheduling schemes and parameters. The picture below represents the topology of our simulation case.

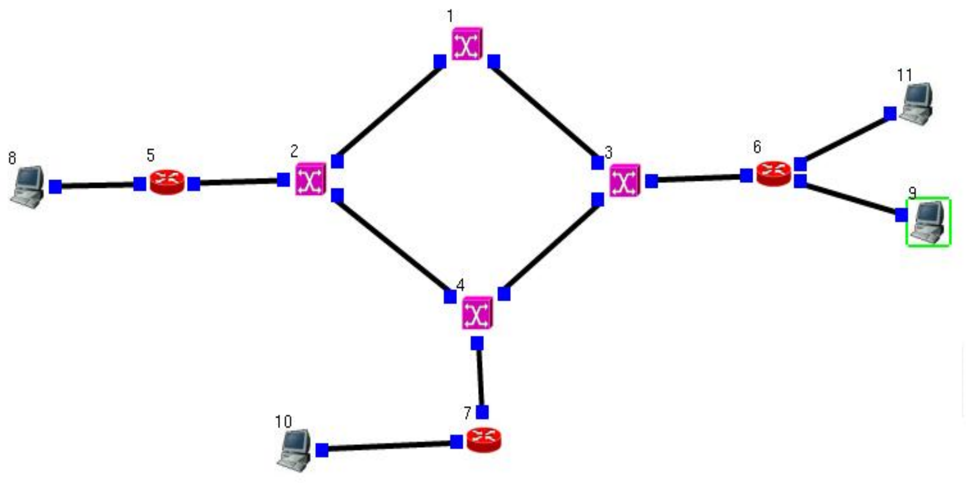


Figure 7.10: The case of the validation of the OBS

There are two UDP connections, one is from node8 to node11 from 0th to 20th second, and the other is from node10 to node9 from 3rd to 20th second. The light path assignment is that “5-2-1-3-6” is a light path, and “7-4-3-6” is a light path. The bandwidth of each optical channel is 10Mbps, and the propagation delay of each optical link is 1 microsecond. The value of the parameter in Ethernet links is the same. The burst length is 16000 bytes, and the burst gathering timeout is 10 micro second. Each optical link has only one data wavelength channel for data bursts.

At the start of the simulation, the “8-11” UDP connection will get the full speed to transmit packet, but when the third second comes up, the data flows of the 2 UDP connections will content at switch3. The burst reservation and dropping algorithms will affect the traffic distribution. The graph below shows the detail.

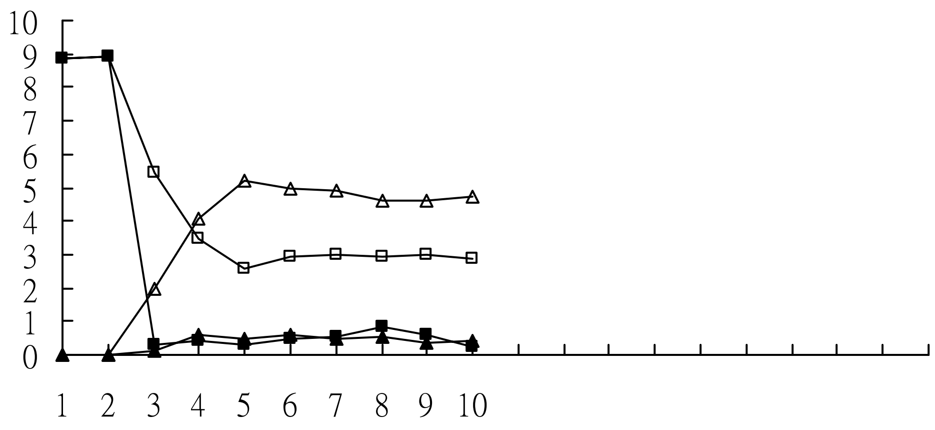


Figure 7.11: The simulation result of the OBS case

The X axis represents the time (seconds) and the Y axis represents the throughput (Mbps). This result is based on using random drop for contended part of bursts and burst segmentation. In this case, the line with hollow square dots represents UDP connection 1 which is from node10 to node9 and the line with hollow triangle dots represents UDP connection 2 which is from node8 to node11. In the first two seconds, UDP connection 1 can get full speed, but after the 3$^{rd}$ second, UDP connection 2 rises and the two UDP connections start competing the resource at node3. Because the UDP connection 2 takes the shorter path, its reserving rate is higher and the UPD connection gets more bandwidth. The random drop algorithm for contending bursts will randomly select a contended burst to drop, and therefore the competition of the 2 UDP connections will reach a dynamic balance that they are in the ratio of 1:2

sharing the whole bandwidth. The reason why the UDP2 can have double bandwidth than the other is that the control packet travel time from node7 to node3 of UDP 2 is 5 micro second, and the travel time of UDP1 from node node5 to node3 is 9 micro second. It means that the average reservation rate of UDP1: UDP2 is about 5: 9, and therefore the bandwidth usage ratio is about 5: 9.

Now let us take a look at the result of dropping without segmentation. The line with solid square points represents UDP1 without segmentation, and the line with solid triangle points represents UDP2 without segmentation. We find that the throughput is very low and there is no special relationship between the 2 UDP connections. This is because the drop is not only drop the contending part, but also the whole burst. We do not implement any arrangement mechanism among bursts, and therefore most of bursts crush and the throughput is very low.

# 8. Command Console

## 8.1. Introduction

The NCTUns provides a few interfaces for observing the simulated nodes at runtime, and most of them are used for watching modules. For example, we can watch the routing table and look at the current routing entry, or we can query the current queue length in the packet scheduling module. However, this is not enough. Sometimes we want to take a look at the whole information of network interfaces, but NCTUns does not provide such function. We not only want to schedule the applications on the application list, but also we want to give several commands or run applications on nodes at runtime. For providing users a convenient way of operating the simulation nodes, we develop an interface called command console to do such work.

Command console is a very convenient tool, and it is a shell program that you can type your commands and execute your applications on the command console, such as traffic generators, ifconfig program, and ping program. The picture below is a capture of command console.

## 8.2. Design Considerations

To design such an interface program, we have some considerations:

- The main goal of developing the command console is providing users an interface which is operated like a normal computer. The operating methods must be the same or similar to those of the current devices, and the interface must look like that of the shell program of the computers. Therefore we pick the TCSH program as the reference of our system design. Besides the source code of TCSH program is available and we need only to modify the program for our specified features.

- The simulation of NCTUns is based on the virtual time, and we know that the applications running on the simulation nodes need to be given the virtual clock to make the performance and functionality correct. Therefore, we need to add the virtual-time-compatible feature to the TCSH program. When our shell forks a new process, the shell has to register the new processes by the system call which is provided by the NCTUns for giving the process virtual time.
- Because NCTUns is a network simulator and it does not use the general network interfaces, the processes have to tell the operating system that they run on the NCTUns and please do not treat us as normal applications. When our command console forks new processes (especially traffic generators), it has to register the new processes by the system call which is provided by the NCTUns. After that the operating system will know the processes are forked by the NCTUns and the packets they generate are put into the mapped tunnel interfaces.
- The command console has to present the view point of a node. When entering a node and opening its command console, this command console is treated as the node. For example, we click the command console bottom and login node1, the shell program in this command console must be presented as node1. However, we know that all nodes in simulation are in the same computer. If we want to tell the differences of these nodes, we have to complete some modifications:

I. The command console needs to know what node it lies on. When we open the command console, we have to bring the node ID into the command console.

II. We have to provide the mapping among virtual network interfaces and the tunnel interfaces of the operating system. When we run ifconfig on the original TCSH program, the all tunnel interfaces that the simulation case uses will be shown. A node has only one tunnel interface as an end host or several tunnel interfaces as a router. Therefore, we need to use the system calls that the NCTUns provides to find virtual network such mapping for command console.

III. We use the ifconfig program watching or configuring interfaces of a node, or we use tcpdump program listening the network interface or capturing the packets from certain interface of a node. This consideration is similar to consideration II. We have to convert the names of the tunnels to those that are viewed from a node. For example, a router uses tunnel interfaces tun3, tun4, tun5, and tun6 as its network interfaces, but we can't display "tun3", "tun4", "tun5", and "tun6" on the command console. When we type "ifconfig –a" on the command console, it should display "fxp1", "fxp2", "fxp3" and "fxp4" on FreeBSD, or "eth1", "eth2", "eth3" and "eth4" on Linux. That is the appropriate display of interface name. Also, we won't type "tcpdump –i tun4" (we even barely know what tun4 represents) on the command console. What we know is that we want to listen on the second interface of the router, and we will type "tcpdump –i fxp2" on FreeBSD or "tcpdump –i eth2" on Linux. Therefore, we have to find the mapping among the virtual interfaces and tunnel interfaces. After that the command console converts the input and output strings to the appropriate name.

## 8.3. Implementation

### 8.3.1. System Architecture

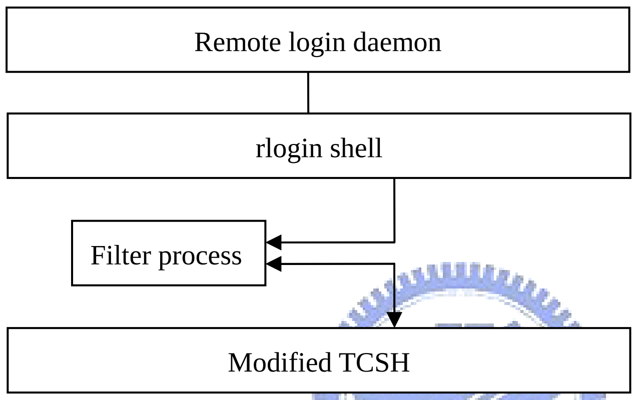


Figure 8.1: The architecture of the command console

The graph above represents the architecture of the command console. There are three main parts of the command console. One is the filter process, another is modified TCSH program, and the other is the cooperation with system and the GUI of the NCTUns.

Now we explain how these blocks make command console work. Remote login daemon and rlogin shell are system provided. In this part, what we have to do is adding some system configuration so that the remote login shell can automatically fork the filter process when command console executes. We do not want to input password at each time that command console logins, and therefore we also add some configurations to the system to ignore the password. The filter process is forked by the rlogin shell, and this process is used mainly for converting and filtering strings to the appropriate content. The modified TCSH program is the main part of the command console, and we give the commands and execute them on it. The main works of the modified TCSH program are the process registry and changing the priority of the processes.

### 8.3.2. Design Considerations

- Modified TCHS:

I. At first, we have to know what node this command console is on. We modify the parameter structure of the program to bring the node number to the program.

II. Each process forked by the modified TCSH on command console should be given virtual clock which is generated by the simulation engine. After the processes being forked, the command console needs to register the processes to the kernel to tell the operating system that the processes are running on the simulation engine and based on the virtual time. Another system-call for registry provides the mapping among virtual interfaces and tunnel interfaces, and therefore the naming simulation IP can be identified and converted to SSDD format in the kernel [1].

   There are two forking points for TCSH program. One is in the function "pfork()" of the file "sh.proc.c" and the other is in the function "execute()" of the file "sh.sem.c". We find the forking points and add system call 261 and system call 262 to register the process.

III. Another problem is the priority consideration. The events of simulation have to be processed as quickly as possible, or we will face the problem of the dead traffic generator. Therefore, the simulation engine has to get higher priority than normal processes for the correct simulation results. Besides, all the simulation related processes need higher priority than the simulation engine. If we don't give these processes a higher priority, we will face problem of getting an incorrect round-trip time. For example, the ping program will get a wrong round-trip time because the ping program can not return the ICPM packets back immediately due to the lower priority. The RTT time will become too big. The picture below is the illustration of the problem. We have to know that all the processes, including normal processes, simulation engine, command console, and simulation related processes are in the same operating system, and they need to be scheduled to run.

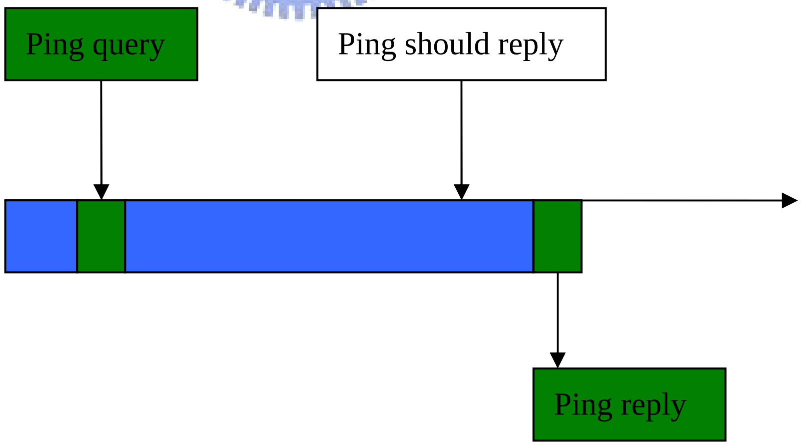


Figure 8.2: The illustration of the delayed response

IV. From this picture, we can see that the blue blocks are the scheduled time slots for simulation engine execution, and green blocks are the time slots for ping program. We know that the simulation network runs on one computer, and the ping program won't reply the ICMP messages until it gets its execution time. In this case the ping program is considered to reply at the time that the white block points, but the simulation engine is still on execution. Therefore, the ping program will be delayed and the returned RTT is wrong.

In Linux, we give these simulation related processes real-time process priority, and the operating system will use round-robin algorithm to schedule these processes. This problem is now solved.

- Filtering Process:

The graph below illustrates the interactivity between filtering process and modified TCSH.

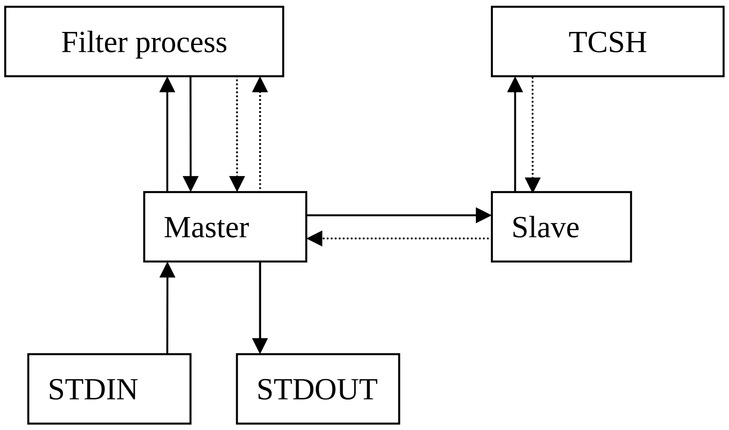


Figure 8.3: The architecture of the filtering process

I. We use the psudo terminal library to build a stream pipe between the filtering process and the modified TCHS, and we can intercept and filter the input characters from STDIN and output strings from STDOUT.

II. Taking a look at the graph above, the doted line represents as the output of the process forked by the modified TCSH, and the physical line is the input from STDIN.

    When we key in a alphabet from the keyboard, the STDIN gets a character and goes to the filter process though the master terminal. The way that the input character goes from filtering process to modified TCSH is "the master terminal → the slave terminal → modified TCSH", and the character follows the same way back to filtering process. After that the iput character will be sent to STDOUT through the master terminal.

III. The output string will be caught by the filtering process, and the filtering process will convert the string to the appropriate one. For example, when we use ifconfig, we type "ifconfig -a". The output of the program will be the whole activating tunnel interfaces. This output is wrong, and therefore we have to discard the strings of the real network interfaces of the computer and the tunnel interfaces which are not used as interfaces of this node. At last the filtering process changes the name of those tunnel interfaces to virtual network interfaces, such as "eth4" or "fxp4", for example. The sequence of tunnel interfaces is converted to the sequence of the node's network interfaces.

IV. The input will be caught by the filtering process, and the filtering process will convert the information to the appropriate information. For example, we want that users can operate the simulated nodes as normal computers. If the user want to use tcpdump to catch the packets, they will type "tcpdump –i eth(number)" (in Linux). However, we know that the virtual network interfaces are all named "tun(number)" in the kernel. We have to convert the string "eth" to "tun", and the sequence number of node's interfaces to the real tunnel interface number. After such processing the system can identify the names and provide the information of the tunnel interfaces.

### 8.3.3. Details of The Program "script"

The program "script" is the main body of the filtering process.

#### 8.3.3.1. Functions of the Program

I. Int convert_fxp_tun(): This function converts the virtual interface port number to the tunnel interface port number. The filter process has a table containing the mapping of the virtual port numbers and tunnel interface numbers. This function gets its job done by checking this table.

II. Int convert_tun_fxp(): This function converts the tunnel interface port number to the virtual interface port number. It is the inverse function to "convert_fxp_tun()".

III. Void Inttostr(): This function converts the integer to string of decimal digits.

IV. Void getcommand(): The work of this function is gathering the typed-in characters from STDIN.

- The filter process has a variable named "mode". When we push the “Enter” key on the keyboard, the variable “mode” will change to “1” to represent that the command input is finished. After receiving the “Return” character, the function sends the command to the filtering process.

- The filter process has a variable named “command_type”, which is used to record type of the input command. After variable “mode” changes to 1, the command string will be parsed to see what type it is. If the string contains “ifconfig”, the fi.ter process marks the variable “command_type” to 1. It means that the command is “ifconfig”. The value “2” of “command_type” represents that the command is “tcpdump”, and the value “0” means the command the other types.
- When receiving a character, this function checks the input character. If it is a control character, the function ignores it and continues the gathering work. Of course, control character should not be added to the command string. Besides, the special control pattern which represents some special key on the key board such as “Esc”, “→”, and “←” must be blocked.
- The graph below illustrates the work flow of the “getcommand()” function.

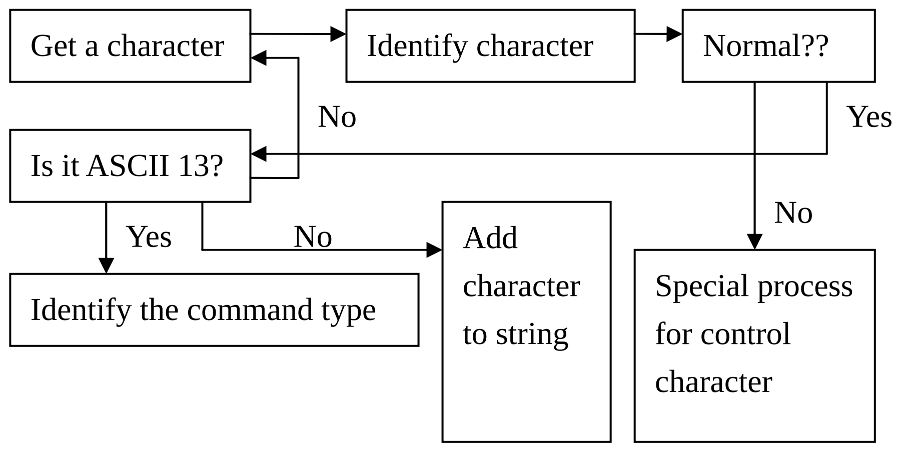


Figure 8.4: The work flow of the filtering process

### 8.3.3.2. The Main Program Activities of Filter Process

I. At first, the filter process uses system calls to get the mapping among the virtual interface port and tunnel interface port. The filter process uses this information to create a mapping table.

II. In the main program body, we use the function “select()” to poll between STDIN and STDOUT. If there is input or output appearing on the STDIN and STDOUT, the program grabs them and handles the conversion and filtering.

III. In polling STDIN, if the program intercepts some characters from the keyboard, it uses the “getcommand()” function to gather the input string and handles the string conversion. For example, if we type “ifconfig eth1”, the program has to convert this string to “ifconfig tun3” (we assume that the virtual-tunnel port mapping is 1 → 3) and sends the string to the TCSH. If we type “tcpdump –i eth1”, the program has to convert the string to “tcpdump –i tun3” and tells the simulation engine to open the tcpdump module flag.

IV. How and when does the program convert the input command? The answer is that when we type the "enter" key, the "getcommand()" function will pack the input string and identify what command it is, and it calls the "convert_fxp_tun()" function to convert the string.

V. In polling STDOUT, the program has to discard the information which is not needed and converts the filtered information to the node's point of view. For example, when we type "ifconfig -a" in the normal shell, the shell will list all the network interfaces of the computer. However, in the simulation environment, what we need are only the virtual interface data of a node. We want the program to list only the interfaces of the node, and discard the rest strings. For example, node1 has three ports and the three interfaces are called "eth1", "eth2", and "eth3". In the operating system they are presented as "tun5", "tun6", and "tun7". The program needs to wipe the fxp0, lo, tun0~4, and tun8~tun4095 and converts the strings "tun5", "tun6", and "tun7" to "eth1", "eth2", and "eth3".

### 8.3.4. Combining with NCTUns and Operating System

- When we want to execute the command console, the GUI has to send the node number and user name to the coordinator. The command console then logins the computer with the user account and node ID.
- This paragraph discusses the configuration of the environment.

  I. When we start the command console, the command console obtains the IP of the computer, node number, and user name, and it calls a program to configure the remote login environment.

II. The environment setup program “tsetenv” will create a file called “NCTUNS_SETENV”, and the file contains the information of configuring the environment. The information includes the environment variables and the calling of filter process. The file “.bashrc” (or “.tcshrc” for TCSH) will be added a line to execute the batch file “NCTUNS_SETENV”.

III. For supporting starting command console without entering password, we need to add a file “.rhosts” in the home directory of that user with a line “(GUI IP) (username)”. After that the remote login from the GUI by this username will no longer need password. However, in Linux, we have some extra work to do for supporting such feature. We have to add “rlogin” this service to the configuration file “/etc/securetty”, and then the authentication will be passed from operation system to rlogin daemon.

# 9. Scalability and Further Work

Because our system is based on the NCTUns, the growth of the system load in our simulation case is the same as that of other cases in the NCTUns. The length of the simulation time depends on the number of the events. If the traffic load is large, the NCTUns needs much time to finish the simulation. The NCTUns uses the kernel of the operating system to simulate the TCP/IP protocols, and therefore the simulation result is accurate. However, the simulation is costly. If there are more than 100 TCP connections with average bandwidth 20Mbps or higher, the computer needs a lot of time to finish the simulation. How do we simulate the internet with thousand nodes and connections? The solution is that the user can develop their own traffic modeling program and traffic generator to simulate hundred or more traffic flows.

So far we implement the optical burst switching network system and the traditional all-optical network system. We provide several empty program functions for users to specify their algorithms for burst scheduling, burst assembling method and contention drop choice in optical burst switching networks. We also provide the same things for users to specify their ring management methods, RWA algorithms and the protection mechanisms in traditional all-optical networks. Also we can add the QoS module which the NCTUns provides in our simulated nodes, or we can modify our modules to support the QoS in the all-optical networks. You can draw an extra-large network with all-optical networks in the backbone to simulate an internet operating in the real world.

We can develop more new subsystems or functions. We only use JET method (Just-Enough-Time) in the optical burst switching networks, and we can develop Tell-and-Go method as our control packet sending method. For the simulation of fiber delay line and optical packet switching networks, the current method is changing the queue length in the optical FIFO module to non-zero and activating the store-and-forward function. However, the most accurate way is setting a timer for each simulated fiber delay line and stretching the propagation delay of the buffered packets longer. Using only a queue to simulate optical packet switching networks is rough, and therefore we can work on developing the accurate scheme of simulating fiber delay line.

# 10. Conclusion

Nowadays the scale of internet grows very fast, and therefore the bandwidth of backbone networks and the processing speed of network devices need to power up to handle such huge traffic flows. All-optical devices are the most suitable equipments and the all-optical networks are one of the most popular computer network research areas. Therefore, the tools which help researchers to research all-optical networks become more and more important.

Simulation is a way of analyzing and gathering the research data. It is more convenient than experiments with the real hardware and more accurate than the mathematical modeling. We use a famous simulator, NCTUns, to develop our system. Because the NCTUns uses real-simulation method, we can get more accurate simulation result. It uses C++ as its developing language, and therefore we save a lot of time and we can focus on our system design and implementation.

We design the system of all-optical networks, including circuit switching scheme, protection ring mechanism, optical burst switching networks, RWA algorithm, and several tools. Also we develop a command console program for monitoring the traffic and controlling the device dynamically.

The performance is reasonable and acceptable. The resources requirement of our system is not exceed that of the NCTUns, and the growth of the requirement depends on the traffic flow. It is the same as the NCTUns. Also we analyze our system. The data of our simulation result is explainable, and the performance and the behaviors of the networks fit system design and the network architecture.

Our system can cooperate with other systems. We can combine other modules or other nodes to simulate merged networks. We also provide some function points so that users can develop their own algorithms or protocols. We hope that our work can help saving the experiment and data analyzing time of the research in all-optical networks, and researchers can focus on their design and implementation.

## References:

[1]. S.Y. Wang, C.L. Chou, C.H. Huang, C.C. Hwang, Z.M. Yang, C.C. Chiou, and C.C. Lin, "**The Design and Implementation of the NCTUns 1.0 Network Simulator**", Computer Networks, Vol. 42, Issue 2, June 2003, pp. 175-197. (EI)

[2]. Rajiv Ramaswami, Kumar N. Sivarajan, San Francisco, "**Optical networks: a practical perspective.**" Morgan Kaufmann Publishers, c1998

[3]. Guido Maier, Achille Pattavina, Simone De Patre, Mario Martinelli, "**Optical Network Survivability: Protection Techniques in WDM Layer**", Photonic Network Communications, 4:3/4, 251-269, 2002.

[4]. C. Qiao and M. Yoo, "**Optical Burst Switching - A New Paradigm for an Optical Internet**", Journal of High Speed Networks, Special Issue on Optical Networks, Vol. 8, No. 1, pp.69-84, 1999

[5]. M. Yoo and C. Qiao, "**Just-Enough-Time (JET): A High Speed Protocol for Bursty Traffic in Optical Networks**", IEEE/LEOS Conf. on Technologies for a Global Information Infrastructure, pp. 26-27, Aug. 1997

[6]. Vinod Vokkarane, Jason Jue, Sriranjani Sitaraman, "**Burst Segmentation: an Approach for Reducing Packet Loss in Optical Burst Switched Networks**", Proceedings IEEE, International Conference on Conference (ICC) 2002, New York, NY, April-May 2002.

[7]. Bo Wen, Nilesh M. Bhide, Ramakrishna K. Shenai, and Krishna M. Sivalingam, "**Optical Wavelength Division Multiplexing (WDM) Network Simulator (*OWns*): Architecture and Performance Studies**", School of Electrical Engineering & Computer Science Washington State University, Pullman, WA 99164.

[8]. Lisong Xu, Harry G. Perros, George N. Rouskas, **"A Simulation Study of Access Protocols for Optical Burst-Switched Ring Networks",** Proceedings of Networking 2002, May 19-24, 2002, Pisa, Italy.

[9]. Alexios Louridas, Kalliopi Panagiotidou and Nathan J. Gomes, "**Simulation of Optical Burst Switching Protocol and Physical Layers**", London Communications Symposium 2002, 2002

[10]. Web Site http://www.opnet.com/products/wdmguru/roi.html **Guru: Simulation of WDM environment of optical networks on OPNET**