MODELING ALL-OPTICAL SPACE/TIME SWITCHING FABRICS WITH FRAME INTEGRITY

by

Luai E. Hasnawi

BS in Computer Science, King Abdulaziz University, 2005

MS in Telecommunication, University of Pittsburgh, 2008

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2016

UNIVERSITY OF PITTSBURGH

TELECOMMUNICATIONS AND NETWORKING PROGRAM

This dissertation was presented

by

Luai E. Hasnawi

It was defended on

August 25, 2016

and approved by

Richard A. Thompson, Professor, Telecommunications and Networking Program

David Tipper, Professor, Telecommunications and Networking Program

Hassan Karimi, Professor, Information Science & Technology Program

Balaji Palanisamy, Assistant Professor, Information Science & Technology Program

Rami Melhem, Professor, Computer Science Department

Dissertation Director: Richard A. Thompson, Professor, Telecommunications and Networking Program

MODELING ALL-OPTICAL SPACE/TIME SWITCHING FABRICS WITH FRAME INTEGRITY

Luai E. Hasnawi, PhD

University of Pittsburgh, 2016

All-optical networks have attracted significant attention because they promise to provide significant advantages in throughput, bandwidth, scalability, reliability, security, and energy-efficiency. These six features appealed to optical transport-network operators in the past and, currently, to cloud-computing and data-center providers. But, the absence of optical processors and optical Random Access Memory (RAM) has forced the optical network designers to use optical-to-electrical conversion on the input side of every node so the node can process packet headers and store data during the switching operation. And, at every node's output side, all data must be converted from its electronic form back to the optical domain before being transmitted over fiber to the next node. This practice reduces all six of those advantages the network would have if it were all-optical. So, to achieve a network that is all-optical end-to-end, many all-optical switching fabrics have been proposed.

Many of these proposed switching fabrics lack a control algorithm to operate them. Two control algorithms are proposed in this dissertation for two previously-proposed switching fabrics. The first control algorithm operates a timeslot interchanger and the second operates a space/time switching fabric - where both these photonic systems are characterized by active Feed-Forward Fiber Delay Line (FF-FDL) and the frame-integrity constraint. In each case, the proposed algorithm provides non-blocking control of its corresponding switching fabric. In addition, this dissertation derives the output signal power from each switching fabric in terms of crosstalk and insertion loss.

TABLE OF CONTENTS

NOMENCLATURE

α coupling ratio

δ_i^o the amount of delay required to switch a timeslot from input index i to output index o

δ_{max} maximum delay

λ the number of wavelengths per fiber

λ_i input wavelength index

λ_o output wavelength index

A insertion loss

B bandwidth

c stage index

C number of stages

D number of nodes

DE_i delay element index i

E number of links

F The number of consecutive frames

f_C center frequency

f_H high frequency

f_i Frame index i , where $i = \{0,...,F-1\}$

f_L low frequency

FDL_i fiber delay line index i , where here $i \geq 0$

i space input port index

k number of switches in path

L loss

m number of output space port when input ports $\neq$ output ports

M number of space output ports

n number of input space port when input ports $\neq$ output ports

N number of space input ports

o space output port index

P_{in}^1 power signal at input port 1

P_{in}^{2} power signal at input port 2

P_{out}^{1} power signal at output port 1

P_{out}^{2} power signal at output port 2

R number of rows

R_B bit-rate

R_L line-rate

S The number of timeslots per frames

S_i^x Timeslot index i going on direction x , where $i = \{0,...,S-1\}$ and $x = \{\text{in,out}\}$, [in: for incoming timeslot and out: for output timeslot] }

S_i^{in} input timeslot index i where $i = \{0,...,S-1\}$

S_o^{out} output timeslot index o where $o = \{0,...,S-1\}$

S_{size} timeslot size

SWA total number of switching assignments

swa_i switching assignment index

SXR signal-to-crosstalk ratio

t arrival time

T_f duration of a frame

T_g duration of a guard time

T_s duration of a timeslot

v_{sw} switching speed

W coupling loss

X crosstalk

ABBREVIATIONS

2D-AON 2 Dimensions All-Optical Network

AON All-Optical Network

BER Bit Error Rate

IP Initialization Phase

CDM Cumulative Delay Matrix

CPU Central Processing Unit

DC Data Center

DE Delay Element

DPAA Dilated Path Assignment Algorithm

DS Digital Signal

DSWA Dynamic Switching Assignment

DWDM Dense Wavelength Division Multiplexing

E/O Electrical-to-Optical conversion

EPAA Economic Path Assignment Algorithm

FB-FDL Feed-Back Fiber Delay Line

FCC Federal Communication Commission

FDL Fiber Delay Line

FDM Frequency Division Multiplexing

FF-FDL Feed-Forward Fiber Delay Line

FIFO First In First Out

GPU Graphics Processing Unit

GTP Guard Time Phase

HD High Definition

HDD Hard Disk Drive

HDM Hybrid Division Multiplexing

ICMP Internet Control Message Protocol

ICT Information and Communication Technology

IoR Index of Refraction

IoT Internet of Things

IP Internet Protocol

IPTV Internet Protocol Television

IPv4 Internet Protocol Version 4

IPv6 Internet Protocol Version 6

ISI Intersymbol Interference

ISP Internet Service Provider

ITU International Telecommunications Union

LAN Local Area Network

LD Laser Diode

LED Light-Emitting Diode

$LiNbO_3$ Lithium Niobate

MAN Metropolitan Area Network

MCF Multi-Core Fiber

MEMS Micro-Electro-Mechanical System

OBSN Optical Burst Switched Network

O/E Optical-to-Electrical conversion

O/E/O Optical-to-Electrical-to-Optical conversion

OFSN Optical Framed Switched Network

OSSN Optical Statistical Switched Network

OTDM Optical Time Division Multiplexing

OTN Optical Transport Network

OTT over-the-top

P_b Probability of Blocking

PC Personal Computer

QoE Quality of Experience

QoS Quality of Service

RAM Random Access Memory

SCF Single Core Fiber

SDM Space Division Multiplexing

SNR Signal-to-Noise Ratio

SOA Semiconductor Optical Amplifier

SSD Solid State Drive

SSWA Static Switching Assignment

STCA Space/Time Control Algorithm

STPAA Space/Time Path Assignment Algorithm

SWA Switching Assignment

SWC switching control

SWM Switching Module

TCP Transmission Control Protocol

TDM Time Division Multiplexing

TICA Timeslot Interchanger Control Algorithm

TSI Timeslot Interchanger

TSP Timeslot Phase

TV Television

UDP User Datagram Protocol

UHD Ultra High Definition

US United States

VDL Variable Delay Line

VoD Video on Demand

WAN Wide Area Network

WDM Wavelength Division Multiplexing

WLC Wavelength Converter

WLI Wavelength Interchanger

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

The number of devices, as well as the number of users, connected to the Internet is increasing gradually. The worldwide average number of devices connected to the Internet currently exceeds the world human population . According to Cisco [6] , there are an average of 1.7 devices connected to the Internet per person. This number is expected to grow to 2.73 devices by 2018, based on the same forecast study. The total number of devices connected to the internet is expected to be between 9 [7] to 13 [6] billion devices by 2018. By 2020, the total number of devices connected to the internet is expected to be 25 billion devices [8] [9].

The Internet is going beyond ordinary usage including, but not limited to, email exchange, web browsing, socializing, etc. We are heading towards a new era of the Internet known as the Internet of Things (IoT). The next generation of the Internet is defined as IoT, where any 'thing' can be connected to the Internet. The evolution of wireless connectivity and sophisticated power sources opened up a new horizon for future innovations by freeing us, as users, from cables and the power source restrictions that have increased our mobility. Devices and gadgets are getting "smarter" by equipping them with Internet connectivity such as: Personal Computers (PCs), smartphones, tablets, Televisions (TVs), wearable gadgets, video games, home appliances, home surveillance and security systems, health care equipments, cars, car meters, sensors in bridges, roads, power plants and other places. All, but not limited to, the aforementioned 'things' that are connected to the Internet, generate traffic inside the network causing network performance to decay as the number of connected devices increases. The amount of traffic generated by any device connected to the Internet varies from one application to another. Some devices generate tens of bits every hour such as signaling and sensors systems, while others generate gigabytes per hour such as High Definition (HD) TVs.

By far, the highest amount of traffic generated into the Internet is video content. Video content includes, but is not limited to, peer-to-peer videos, Internet Protocol Television (IPTV), Video on Demand (VoD), and broadcast TV. In 2013, 60-66% of the total Internet traffic was video content [6][10]. By 2018, video content traffic will account for 69% [10] to 79% [6] of total Internet traffic. According to Alcatel-Lucent [11], by 2018, the total amount of video traffic is expected to be around 780 Exabyte compared to 1.2 Zettabyte, according to Cisco [6], for the same year. The Internet should be ready to handle the expected traffic inflation. The Internet infrastructure must grow faster than the time needed for users to adopt new technologies, particularly bandwidth eater applications like Ultra High Definition (UHD) TV. If, for some reason, high traffic applications grow faster than the growth of the Internet's infrastructure, then Internet Service Providers (ISPs) will be unable to provide the promised Quality of Service (QoS). In addition, over-the-top (OTT) video content providers cannot promise satisfactory Quality of Experience (QoE). The worst-case scenario happens when the Internet utilization exceeds its 100% limit, which results in an enormous packet drop or collapse.

Generally, network applications are sensitive to latency, bandwidth or integrity [12]. In all cases, optical communication provides fast, high bandwidth and reliable communication infrastructure compared to electronic or wireless networks. Therefore, introducing photonics technology is mandatory. There has been a decent amount of improvement on Optical Transport Network (OTN) infrastructure lately by adding new fibers or replacing copper cables with fiber optics. In addition, the introduction of link multiplexing using Wavelength Division Multiplexing (WDM) has improved OTN utilization. However, the bottleneck is not in the transport network but in the switching fabrics. The existing electronic switches in OTN, as well as in Data Center (DC), introduce a significant amount of latency. All-optical switched networks provide higher bandwidth, more reliability and lower latency networks compared with electronic networks. Therefore, the proposed switching fabric and its control algorithm will provide an efficient all-optical network to replace the existing electronic switches in OTN and DCs. The three important motivations that led to this study includes: first, a switching fabric that can easily adapt to future improvements. Second, a switching fabric must be scalable. Last but not least, a switching fabric should reduce the energy

consumption. Each one of these motivations is discussed in great details bellow.

1.1 FUTURE TREND ADOPTION

As discussed earlier, the future of the Internet is heading toward the IoT. In the near future, there will be new devices connected to the Internet each generating traffic. Most of the forecast research indicates that the majority of future Internet traffic will be video content traffic. There are three main factors that would increase the Internet Protocol (IP) video content on the internet, which includes: the user adoption of media streaming boxes, TVs are getting smarter, and the introduction of UHD video content.

According to market survey [13], there are about 10 million media streaming boxes in the US. This number is expected to grow to 50 Million in 2017, according to the same research. This implies that 10 million households are generating video IP traffic to the Internet such as YouTube, Netflix, Hulu, etc. This third party video content is known as OTT.

In addition, almost all of the newly manufactured TVs are smart TVs. A TV is said to be smart if it has the capability to connect to the Internet to stream OTT video content, browse the Internet, and access social networks applications directly from the TV.

Lastly, TVs are entertainment devices for most people. Users like to enjoy their entertainment experience as much as possible. UHD (also known as 4K) video content was introduced to the market not long time ago. UHD video content has more pixels per inch than HD once, for a better entertainment experience. UHD video content requires that content is filmed (or produced) in UHD technology, transmission line is capable of streaming UHD video and the projection device (TV or projector) is capable of projecting UHD video content. Table 1 presents the recommended transmission bandwidth for four major OTT video content providers, in addition to what a technology-manufacturing leader recommends for a better experience.

Table 1 shows that UHD video requires more than seven times the amount of bandwidth that a Netflix video requires in standard definition and four times the amount of bandwidth for a HD Netflix video. The variation on the required bandwidth depends on the compression

Table 1: Recommended data-rates for different video streaming providers' in Mbps

	Netflix	**Hulu**	**Amazon**	**YouTube**	**Cisco**
SD (720 x 480) pixels	3.0	1.5	0.9	1.0	2.0
HD (1920 x 1080) pixels	5.0	3.0	3.5	4.5	7.2
UHD (3840 x 2160) pixels	25.0	N/A	N/A	N/A	18.0

scheme. In addition to the bandwidth requirement, the price of UHD TVs worldwide has dropped dramatically over the past few years. The average price of an UHD TV worldwide dropped from $7,851 to $1,120 between 2012 and 2014 [14]. In China, the average price of an UHD was dropped significantly from $1,000 - $973 between 2012 and 2014 [14]. Other reports present slightly different price drops. For example, according to The Wall Street Journal [15], the average price of an UHD TV was dropped worldwide from $3,313 to $1,637 between 2013 and 2014. In addition, the price of an UHD TV was dropped in China from $1,288 to $973 between 2013 and 2014. Moreover, the total number of UHD TVs sold worldwide varies from 8 million [6], 12.5 million [15] and 15.2 million [16] TVs. With the current Federal Communication Commission (FCC) regulation, Open Internet [17] ,broadband companies have no right to block any type of traffic that could flood their networks. Previously, some carriers, like COMCAST, slowed down OTT video traffic in order to avoid network overutilization [18]. Netflix agreed to pay two major broadband companies (COMCAST and Verizon) directly, in order to speed up their traffic and provide better video experience for their customers [18]. The impact of the new rules is extra video content will be transmitted over the internet without ISP restrictions. Hence, the future Internet must be capable of providing high data-rates with minimal Bit Error Rate (BER) and low latency in order to satisfy the user trend.

1.2 REDUCING THE NUMBER OF PHYSICAL PORTS

In space only switching (subsection 2.3.2), switching fabrics have a limited number of ports on each side of the fabric [19]. The input side (ingress) fans-in with N number of ports, while the output side (egress) fans-out with M number of ports. Increasing the number of ports on both sides, results in an increment in the number of simultaneous connections that the fabric can handle. However, increasing the number of ports usually results in an increase in the fabric footprint. In addition, increasing the number of ports results in adding new components to the fabric, which increases the retailer's cost. Instead of adding more ports to the fabric, utilizing the unused spectrum will increase the number of channels that the fabric can handle. TDM and WDM (subsections 2.1.2 and 2.1.3) are being used in OTN, however, channels are not being switched optically in OTN, except for WDM in some cases. Combining Space Division Multiplexing (SDM), TDM and WDM together, not only results in efficient utilization, but also results in reducing the network latency associated with the electronic switching and reducing the switching fabric footprint.

The proposed 2 Dimensions All-Optical Network (2D-AON) (chapter 8) adds more channels to the switching fabric while keeping the number of physical ports constant. Table 2 compares the number of ports in space-only switching fabric with the number of ports in the proposed 2D-AON switching fabric in terms of the number of channels. The first module represents the 2D-AON switching using the < S , T > notation. Where S: is the number of space channels and T: is the number of time channels. The total number of channels is giving by S x T. For a 4 channels rearrangeably nonblocking switching fabric, there are six 2x2 Lithium Niobate ($LiNbO_3$) switch in space only Beneš network, Figure 1 , compared to five switches in space/time switching fabrics, Figure 69. The difference is not significant, however, as the number of channels increases, the difference in the number of switches becomes significant. For instance, there are 16 switches in <4,4> space/time switching fabric, Figure 1, while the equivalent 16x16 space only Beneš switching fabric has 56 switches, Figure 2. Hence, the difference between the numbers of switches become significant.

Lastly, replacing copper cable bundle in data centers with fiber optics results on reducing the cable bundle size because the diameter of a fiber optics less than it is in copper cables.

Table 2: Comparing number of ports and channels between space only and Space/time fabric

	Number of Channels	**Number of Input/output ports**	**Equivalent Space Only**
<2,2>	4	2	4x4
<4,2>	8	4	8x8
<2,4>	8	2	
<4,4>	16	4	16x16

The save space could be used for future upgrades to the network or the data centers.

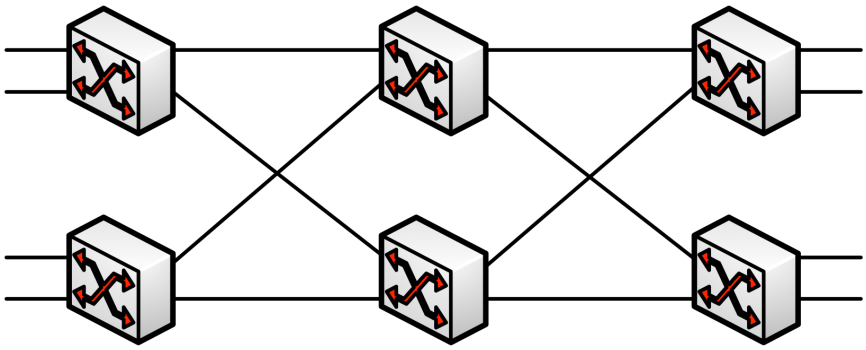


Figure 1: 4x4 Space Beneš Network

1.3 REDUCING POWER CONSUMPTION

The availability of Internet access has made companies that chose to eliminate storage from their devices, to provide replacement storage in the cloud. Recently, Cloud computing was introduced at the consumer level for a decent price after it was limited to large organizations and institutions. Cloud computing provides different services to meet the user's need, such as high computation computers, data storage, backup system, web hosting, email services,

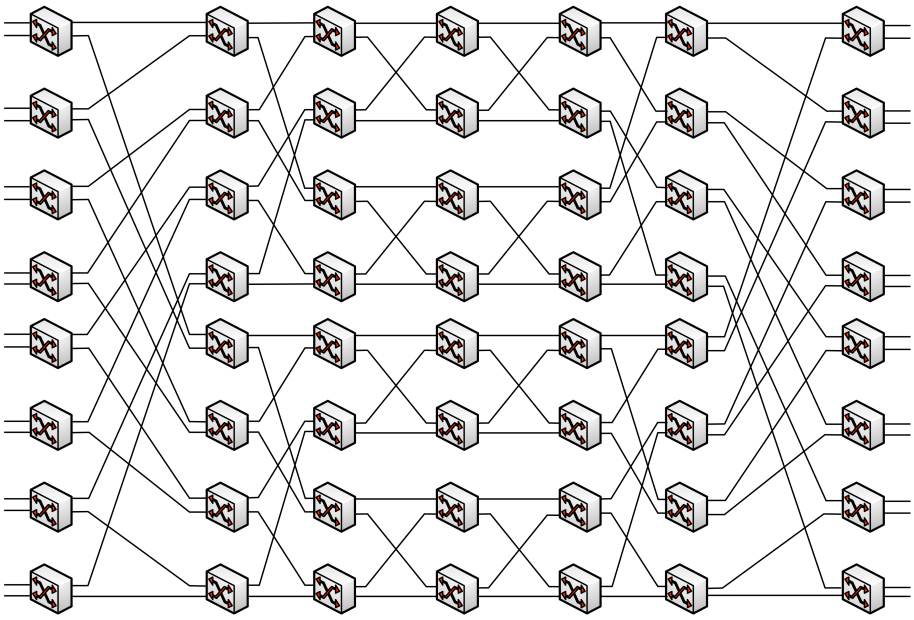


Figure 2: 16x16 Space Beneš Network

and lots of other services that are beyond the scope of this research [20]. Servers, computers, storages, switches, routers and other cloud computing essentials are hosted inside data centers. The general structure of data centers is shown in Figure 3. In the past few years, many companies have decided to produce portable, price competitive and energy efficient devices to keep up with consumer trends. Devices with large storage embedded, such as hard drives, are less portable because they are large in size and heavy in weight. Some companies replaced large size Hard Disk Drive (HDD) with smaller Solid State Drive (SSD), while others removed the storage completely from their devices to provide smaller, thinner, portable, and energy efficient products. Laptops, tablets and cell phones use small SSD such as Apple's iPad and Google's Chromebook. Streaming media devices such as RokuTV and Amazon FireTV are storage-free devices. In the current price competitive market, most companies are trying to compete on price. Some companies refuse to reduce their product's promised quality in order to reduce the price. Thus, removing the internal storage reduces the production cost. Lastly, as more devices become portable, extending the battery life

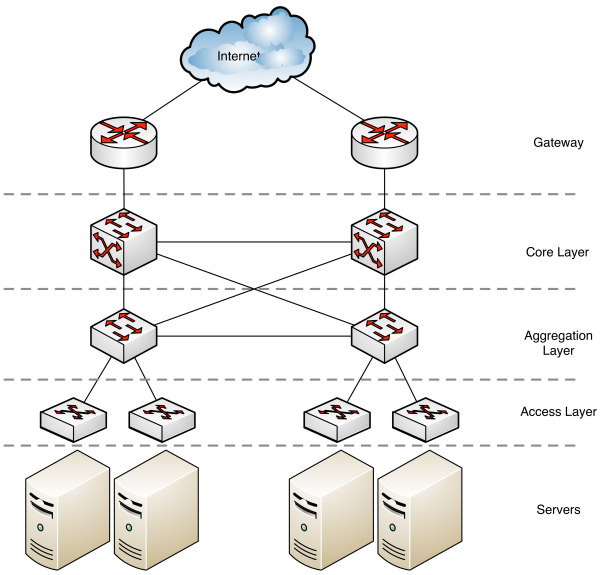


Figure 3: Generalized data center network

becomes a mandatory feature for all portable-manufacturing companies. The leftover space from replacing HDD with SSD, or by completely removing the storage, could be used to increase the chargeable battery size, if needed. Moreover, eliminating the mechanical HDD increases the battery life of the portable device.

The current electronic devices inside data centers, which include servers, switches, data storage, computers and other devices, consume significant amounts of energy to keep data centers running. In addition, data centers' devices produce a significant amount of heat. If the produced thermal is not reduced continuously, service outages are likely to occur. In a sophisticated system, hazards are avoided using a thermal cutoff threshold. However, thermal cutoff results in expensive service downtime that can cost an average of $14,000 per hour [1]. To overcome thermal problems, data centers' equipment and facilities are continuously cooled down. Equipment and devices are cooled down using fans mounted on them, and the facility is cooled down using large air conditioning and ventilation systems that consume massive amount of energy. Thus, data centers' power is consumed by running machines, cooling the facility and equipment, and by the backup power needed to keep data

centers running if a power failure occurs. The question is, how much power does Information and Communication Technology (ICT) consume per year? According to the United States (US) Department of Energy [21] , 120 Billion kWh was consumed by ICT in the US in 2009. This number represents 3% of the total power produced by the US versus 8% worldwide for the same year [22]. In the US, ICT power consumption is divided into 3 major categories [21]: 1. 60 Billion kWh is consumed by data centers. 2. 40 Billion kWh is consumed by cell phone towers, private exchange, local ICT equipment, and others. 3. 20 Billion kWh is consumed by large telecommunication centers and trunk line networks. Data centers' power consumption doubled in the six years between 2002 and 2008 [21]. This was due to the migration of information storage from personal devices to the cloud. According to a forecast study [23], data centers will consume 75% of the total produced power in 2025, if no power conservation is introduced to cap the growth in power consumption in data centers. Another forecast study [24] claims that the ICT power consumption will increase from 623 Billion kWh in 2009 to 1,963.74 kWh by 2020, if no power conservation is introduced. Thus, before discussing any solution to reduce data centers' power consumption, we must investigate where the power is currently going. In ICT, power is consumed by three major sources[21] [25] : 1. Equipment operation power. Power that is consumed by equipment to keep them running throughout the year, including servers, computers, network devices, racks and other devices. 2. Power that is used to cool ICT facilities and equipment, mainly air conditioning systems. 3. Power chain supply. Power that is consumed to backup systems and power redundancy.

It is important to mention that keeping ICT's power operational, requires cooling the devices using fans mounted directly to the devices components (such as Central Processing Unit (CPU)) and that other fans that are mounted to the device to cool the rest of the components. Moreover, every rack inside a data center has fans mounted on top of it to circulate air to the rack's devices. Thus, the amount of power consumed for cooling ICT equipment and facilities is huge, and exceeds 25% of the total cost [21].

The above discussion regarding power consumption in ICT shows that it is essential to reduce the amount of power consumption for ICT. However, reducing the power consumption must not affect ICT availability and functionality. There have been many proposed

solutions to reducing ICT power consumption including: using renewable energy [24], using advance cooling and ventilation systems, distributing the load between different ICT [26], using higher quality fibers [1] and introducing optical switched networks to substitute for the existing electronic switched network inside and outside ICT [21] [22] [23] [27] [28]. Introducing photonic technology to ICT increases the network's bandwidth as discussed in section 1.1, and provides energy efficient replacement for the current energy eaters' electronic network and data centers' equipment. The absence of optical RAM and optical CPUs limits the electronics' replacement to optical network switches and optical fibers. The result of this replacement is major power reduction in the amount of power required to operate optical switching components, a reduction on the amount of power required to cool down the optical switching component, and a reduction on the amount of power required to cool down the switching center or data center's facility. The expected amount of power reduction in telecommunication systems varies from 55 to 70% [1] as shown in Figure 4. This figure compares the amount of power consumption for optical switched networks and the amount of power consumed by electronic switched networks in term of the number of ports per switching fabric. In addition, the figure presents the percentage of energy saving if copper switches are replaced with optical onece. Meanwhile, the US department of Energy suggests that the amount of power consumed by data centers can be reduced by 75% [21].

Fiber optics carries enormous amounts of bandwidth that carries huge amounts of traffic. The multiplexing mechanism is used to increase the number of users per network. Thus, any cable failure is going to affect more users than would copper cables [29]. For this reason introducing photonics technology to ICT is very critical upgrade.

1.4 PROPOSED WORK

Expanding the current electronically switched network in ITC is not the solution to achieve the previously mentioned motivation. The solution must start with replacing the electronic switched network to an all-optically switched network. All-Optical Network (AON) promise to deliver ultra-fast, high bandwidth, efficiently utilized, energy efficient and reliable switched

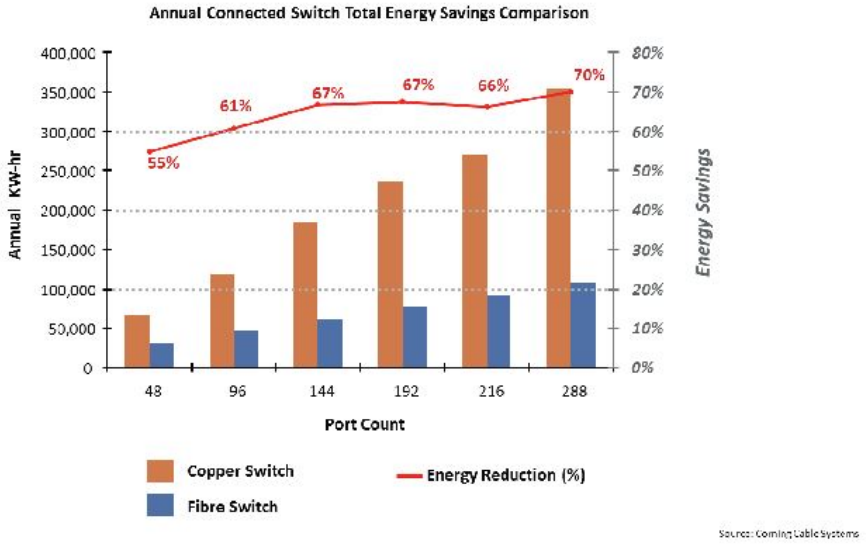


Figure 4: Expected energy reduction when optical switches are introduced [1]

network. There has been a significant amount of research that proposes switching fabric in space division and the combination of space and time division with promising results. Unfortunately, majority of the proposed switching fabrics are theoretical. Very few of them have been simulated or became products.

This dissertation takes two of the previously proposed switching fabric in time and space/time to life by building the required software to operate and control the fabrics. The proposed fabrics are assumed to be nonblocking fabrics. Both switching fabrics can be used to replace current electronic core networks in ITC or transport networks for Wide Area Network (WAN).

The first switching fabric to be modeled and simulated is a time switching fabric (also known as timeslot interchanger) for TDM network with frame integrity. The following algorithms have been developed:

- Chapter 5: Timeslot Interchanger Control Algorithm (TICA) to allow the time switching fabric to perform the interchanging operation while maintaining frame integrity.
- Chapter 6: DPAA, which assign incoming timeslots to a desired path in the fabric that

result on a crosstalk free signal.

- Chapter 7: EPAA, which assign incoming timeslots a desired path in the fabric which result on fewer number of hardware components in the fabric.

The second switching fabric to be modeled and simulated is a space/time switching fabric for TDM network with frame integrity. The following algorithm have been developed:

- Space/Time Control Algorithm (STCA).
- Space/Time Path Assignment Algorithm (STPAA).

Since both switching fabrics (time and space/time) are originally presented as nonblocking fabrics, all of the developed algorithms are implemented to be nonblocking.

Path assignment algorithms' developed for timeslot interchanger includes a power loss and crosstalk measurements. As mentioned earlier, the purpose of each algorithm is operate the switching fabrics and validate the nonblocking claim. The power loss measurement is limited to insertion loss for each component in the fabric and the amount of crosstalk produced by each switch. The value of insertion loss and crosstalk introduced by each switch will be taking from leading companies in the industry as well as number of the academic literature.

2.0 BACKGROUND

This chapter presents optics and photonics background related to the dissertation. Some concepts are presented in depth while others are presented briefly. The depth of details depends on the concept's relation to the dissertation.

This chapter begins by introducing the three basic multiplexing schemes in section 2.1. The next section 2.2, presents some photonic hardware used in the dissertation. In addition, section 2.3, discusses switching and two important concepts including: guards and channels in all domains. Lastly, because blocking is the main performance metric in this dissertation, it is presented in detail, separately, in section 2.4.

Although this dissertation models switch fabrics in space and time domains, it is improper to completely ignore concepts in wavelength domain. Thus, wavelength domain concepts are briefly discussed intentionally.

2.1 MULTIPLEXING

In telecommunication systems, multiplexing is practice to share the medium between multiple users. Multiplexing improves the network utilization and increases the line rate. Multiplexing can be achieved in different divisions: space, time, wavelength, or any combination of two or three divisions. The next subsections will discuss: space, time, wavelength and the combination of space, time and wavelength multiplexing.

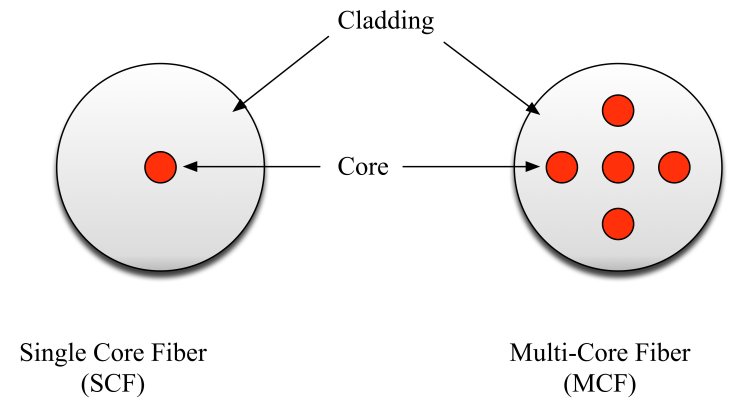


Figure 5: A schmatic representation for a Single-Core Fiber and a Multi-Core Fiber

2.1.1 Optical Space Division Multiplexing

Space Division Multiplexing (SDM) in optics has two definition: the first definition is associated with Single Core Fibers (SCFs), and the second definition is associated with Multi-Core Fibers (MCFs), Figure 5. In SCFs, providing a physical connection between two nodes in multiple individual fibers to exchange information without conversion is the definition of SDM. Meanwhile, in MCFs, a single fiber with multiple cores each carrying information in different spatial is referred to SDM [30]. For the purpose of this dissertation, the traditional SDM in SCFs is discussed in greater details.

The definition of space channel in SCF, is a physical fiber between two nodes. It takes only a single fiber to connect two nodes to communicate and exchange information. However, as the number of nodes (D) at each end increases, a direct physical connection between each of the two nodes requires extra hardware components and more fibers. Each node must have D network interfaces to connect each fiber. In addition, for D nodes, the required number of links (E) that would connect every node on one side, with every node on the other side with direct links to establish a fully connected mesh networks is given by:

$$\frac{N(N-1)}{2} \tag{2.1}$$

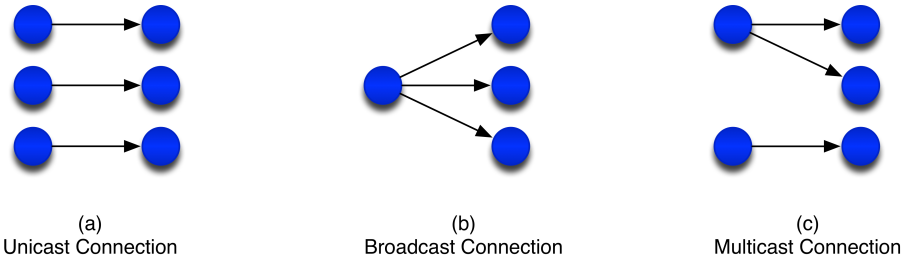


Figure 6: Three different connection categories: (a) unicast (b) broadcast (c) multicast

In order to provide multiple simultaneous connections between any two nodes, space switching fabrics are implemented. There are some concepts associated mainly with SDM that are also used in other multiplexing schemes. The first concept, is the type of connection between two or more nodes. SDM connections fall into one of three different categories: unicast, multicast, and broadcast.

- **Unicast**: This is a (One-to-One) connection, Figure 6 (a), where two nodes communicate with each other. In this dissertation, all communications are assumed to be unicast.
- **Broadcast**: This is a (One-to-All) connection, Figure 6 (b). In this type of connection, a source sends information to all nodes connection the network. The downside of this communication is that some nodes may receive undesired information.
- **Multicast**: This is a (One-to-Many) connection, Figure 6 (c). This communication category reduces the amount of traffic on the network by sending information to a desired destination, only. In addition, instead of establishing multiple unicast connections for the the same content, multicast duplicates the content and sends it to multiple nodes.

There are many advantages of using SDM multiplexing scheme over TDM or WDM, such as [30]: it has the lowest cost per bit in all multiplexing schemes; because the hardware required to build an SDM network is cheaper than it is in TDM or WDM; SDM networks improve the energy consumed by the network hardware; and lastly, integration between cross-connected networks is less of an issue in SDM networks.

Establishing connections between all nodes in SDM networks is achieved by using space

switching fabrics (or fabrics for short). Space switching is discussed in greater detail in subsection 2.3.2.

2.1.2 Optical Time Division Multiplexing

Optical Time Division Multiplexing (OTDM) inherits its name and concept from electronic TDM. The term "optical" is used to distinguish the domain. Some literature uses the OTDM when discussing AON only, but whenever the networks involve Optical-to-Electrical conversion (O/E) or Electrical-to-Optical conversion (E/O), the term TDM is used[1]. In OTDM, the medium bandwidth is shared based on time, Figure 7. The time channels' name varies; for instance, the term "timeslot" is commonly used for time channels with fixed size (or fixed duration) and the term "packet" is commonly used for time channels with variable sizes (variable duration)[2]. Optical networks benefit from TDM by increasing the number of users on the network, improves utilization and generates profit.

In OTDM, the bandwidth is sliced into channels based on time. OTDM network is classified into three classification based on time channels duration : Optical Framed Switched Network (OFSN), Optical Statistical Switched Network (OSSN), and Optical Burst Switched Network (OBSN). OTDM networks with fixed time channel size is commonly known as OFSN (e.g. circuit switched network). OTDM networks with a variable time channel size are commonly known as OSSN (e.g. Packet switched network). The third type of OTDM network is a hybrid between OFSN and OSSN, known as OBSN. These three different classifications of OTDM networks (OFSN, OSSN and OBSN) are discussed below in greater detail. Note that this dissertation focuses on OFSN.

2.1.2.1 Optical Framed Switched Network (OFSN). There are two main criteria that distinguish this type of network: connection setup and fixed timeslot size. In OFSN, the source requests a channel reservation from the fabric (or network) controller. When resources (timeslots) are available, the request is granted and a connection is established.

[1] In this dissertation, TDM refers to electrical domain while OTDM is used otherwise.

[2] In this work, "timeslot" is used for fixed size time channel and "packet" is used for variable size time channels

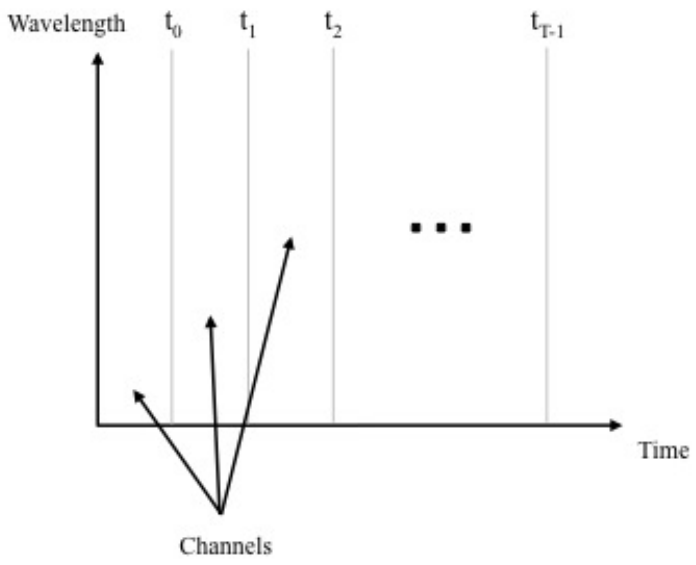


Figure 7: A schematic representation for TDM scheme on a medium bandwidth

On the other hand, if resources are not available, the connection setup request is rejected. Connection setup and resource availability is discussed in detail in subsection 2.3.3. OFSN header processing does not exist, although some OFSN timeslots contain headers, they are not processed at every switching node. Connection setup introduces latency (delay) to the total end-to-end delay, however, the introduced latency during connection setup becomes negligible as the amount of transmitted data increases [12]. In addition, connection setup reduces the Probability of Blocking (P_b) by finding the best route, with sufficient resources, for any desired connection [31].

The second criterion that makes OFSN unique, is the fixed timeslot size. Every user is assigned to a single (in single-rate network) or multiple (in multi-rate network) timeslot. Timeslots are repeated for as long as the network is operated, as it appears in Figure 8. A user uses the entire spectrum for a certain amount of time. If the size of the data is greater than the size of the timeslot, the user resumes transmission when a timeslot is repeated. Once the user finishes transmitting data, the connection is terminated and timeslots are set to vacant for other users.

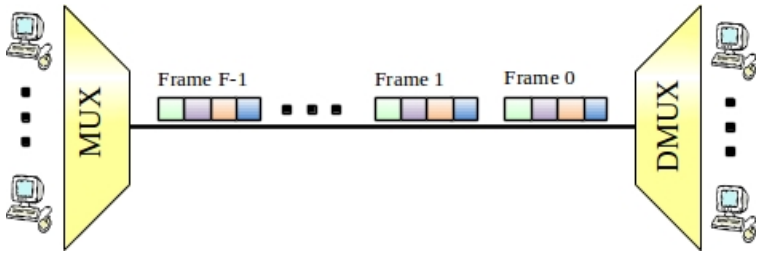


Figure 8: OFSN schematic diagram

Timeslots can be as short as a single bit or multiple of bits. The network is said to be bit multiplexed if the block size is one bit, meanwhile, the network is said to be block multiplexed if more than one bit is multiplexed in a block. A T1 network is an example of OFSN, where each timeslot is a block of 8 bits of data.

Generally, OFSN is a synchronized network; frames and timeslots are more predictable (in interval and size) than in an OSSN. OFSN system's operation and management is easy compared to OSSN. Such a system is ideal for time sensitive applications, including voice and videoconferencing. However, a major drawback of OFSN is that these systems are centralized, and a system controller manages switching decisions. Hence, the availability of the system is highly dependent on its controller. In addition, adopting new applications is not as easy as it is in OSSN.

2.1.2.2 Optical Statistical Switched Network (OSSN). The second classification of OTDM network is Optical Statistical Switched Network (OSSN). The main two criteria that distinguishes OSSN from OFSN are the absence of connection setup and header processing. Time channels in OSSN (packets) consist of two parts, a header and payload. Headers carry control information such as: routing, security and error control, while payload carries the actual data. Unlike OFSN, connection setup does not exist in OSSN, hence, the delivery of the content is not guaranteed from the first attempt. The network does its "best effort" to deliver packets to its desired destination. In OSSN, the source slots the information into packets and attaches headers to each packet before it is sent to the network. Once a packet

arrives to an OSSN switching node (switch or router), the switching node processes the packet's header and directs the packet to either the destination or to the next switching node that would lead to either the next switching node or destination, and so on until the packet reaches its destination.

In an OSSN network, there is no network controller as in OFSN, hence, the network status (e.g. network load, end-to-end delay, packet drop rate ... etc.) is unknown to almost all network nodes. However, some control protocols can scan the network and report the current network status, but they cannot forecast the network status. Sadly, some control messages have been misused. For instance, some Internet Control Message Protocol (ICMP) have been used to lunch malicious attacks. This behavior has forced some ISPs to configure their firewalls to deny (drop) ICMP traffic.

The absence of network status and network a controller results on contention at output ports. Packets contention occurs when two or more packets are destined to the same output port at the same time. If contention occurs at any node, depending on the traffic protocol and the configuration of the node, packets might be dropped, deflected or queued [32]. Packet drop reduces network throughput and causing additional traffic from retransmitting dropped packets. However, deflection and packet queuing are a better solution than dropping the packets. Deflecting packets happens when the buffering time is expected to be long or a queue is about to reach a certain threshold, hence the controllers deflect the packet to a different route or different buffer (if possible). Queuing packets is another solution. It is important to mention that in some applications (e.g. voice), queuing packets might result in an unacceptable delay. Buffering is a bottleneck on any network. Head of queue blocking is a serious problem in the existence of queues.

In OSSN, packets generally do not have a fixed size. Packets' headers usually have a fixed size while packets' payloads have a maximum size as opposed to a fixed size in OFSN. The header processing operation increases the end-to-end delay, which is not preferred for certain types of applications. Internet Protocol (IP) network is an example of OSSN.

2.1.2.3 Optical Burst Switched Network (OBSN).

OBSN is a hybrid combination between OSSN and OFSN, where connection establishment is required while burst size is

variable. In OBSN, a source node transmits a single header followed by a burst of packets. OBSN consists of two channels: a control channel and a transmission channel. A control channel usually has a lower bit rate than a transmission channel. Using such techniques provides a guaranteed service grade while reducing the processing overhead. For a stream of packets in a typical OSSN, each packets' header is processed at each node, while in OBSN, only one header is processed per connection, followed by a stream (burst) of payload without headers. OBSN eliminates data buffering, whereby network performance improves. Bursts have a longer duration than packets and timeslots. OBSN is beyond the scope of this work (more details are available in [33] [34])

2.1.3 Optical wavelength Division Multiplexing

Sharing the medium bandwidth based on frequency is a common multiplexing scheme to increase the number of users, improve utilization, and generate profit. Slicing the medium bandwidth based on frequencies is known as Frequency Division Multiplexing (FDM), which is widely practiced in wired and wireless electronic systems. In FDM, multiple users transmit their data simultaneously across a single medium (cable or air). However, FDM is represented in wavelengths instead of frequencies in an optical domain, and it is known as WDM. The relationship between frequency and wavelengths is giving by equation 2.2, where c is the velocity of light in the vacuum and f is the frequency.

$$\lambda = \frac{c}{f} \tag{2.2}$$

In WDM, multiple users transmit their data on a single fiber simultaneously without interference, Figure 9. Each user transmits data on different wavelengths to a multiplexer. A multiplexer is a hardware device that couples connections from different inputs on different wavelengths to a single fiber. Multiplexers are built using different approaches, however, opto-electronics multiplexers are the most used approach. A light beam carrying data is coupled optically, then processed electronically, and then sent to a laser to transmit it optically over fibers. The fiber ends at a demultiplexer on the receiver side. A demultiplexer is a hardware that reverse a multiplexer's process by decoupling (separating) each connection into

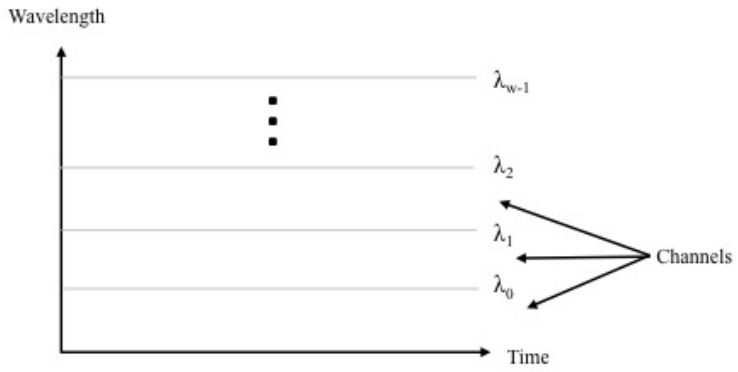


Figure 9: A schematic representation for a WDM scheme on a medium bandwidth

different output ports as demonstrated in Figure 10. Demuliplexers are also opto-electronic devices, where data is received optically, processed electronically and transmitted optically to the desired output ports. All-optical multiplexers and demultiplexers are still in the development stage [35].

Wavelength Division Multiplexing is widely used in OTN, however, there are some disadvantages associated with WDM networks including: high cost, Optical-to-Electrical-to-Optical conversion (O/E/O), and scaleability. WDM networks are expensive to build and maintain, because lasers and photo-detectors are expensive hardware and require periodic replacement. In addition, the presence of O/E/O conversion in multiplexers, demultiplexers, Wavelength Converters (WLCs) and other photonics components introduces a bottleneck in fast optical networks because they operate on an electronic clock speed that is slower than the speed of light in the fibers (photonics hardware components are discussed in section 2.2). Moreover, WDM networks do not scale well as the size of the network increases. To add more users to the network, while assigning each user to a different wavelength in a WDM network, a massive upgrade must be done, such as: using expensive hardware that supports a narrow-band spectrum to accommodate a high number of wavelengths. Dense Wavelength Division Multiplexing (DWDM) shares the same concept as WDM, however, it operates at $\lambda = 1500nm$, which requires expensive hardware, such as lasers, switches, fibers, etc. Typical WDM networks carry between 10 to 32 channels, while DWDM networks carry between

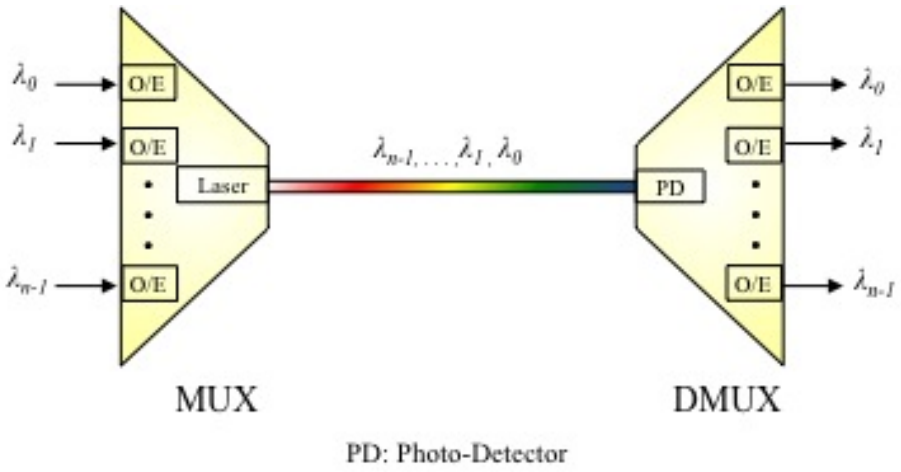


Figure 10: An example of a WDM network using opto-electronic multiplexer and demultiplexer

40 to 100 channels [3] [35] [**?**]. In a C-Band spectrum, the International Telecommunications Union (ITU) proposed the standard DWDM channel spacing on 100 Ghz as it apears in Table 18 in Appendix B.

2.1.4 Hybrid Division Multiplexing

In Hybrid Division Multiplexing (HDM) scheme, a combination of two or three multiplexing schemes (SDM, TDM or WDM) is used to increase the number of users on the network and effectively utilize the medium. The bandwidth in a fiber optics is beyond the use of any end user. If only TDM is used, there will be as many users as the number of timeslots per frame. In addition, the amount of bandwidth for a certain period of time equals the entire spectrum of the band. For end users, there will be an unused spectrum in the assigned timeslot. The unused spectrum is a waste resource. The same scenario applies to a WDM scheme. However, in HDM, the bandwidth is sliced based on wavelength, and each wavelength in also sliced in time. The total number of users that can share the system equals the number of

[3]The number of channels varies based on different factors, such as: wavelength band, spacing between the channels, optical source, fiber material, fiber mode and other factors

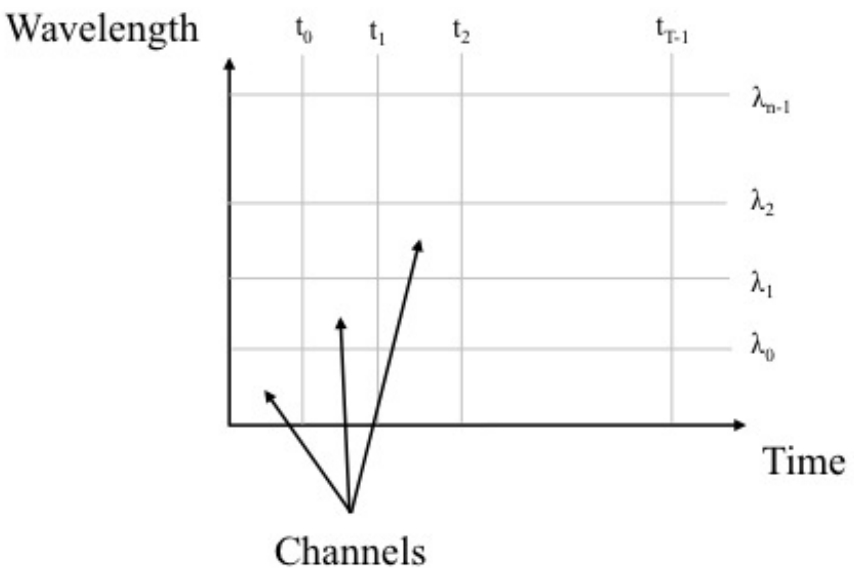


Figure 11: A schmatic representation of SDM over TDM over WDM, which is defined in the work as HDM

wavelength per fiber times the number of timeslots per frame. The schematic representation of a single space fiber that is multiplexed in wavelength and time is presented in Figure 11.

2.2 PHOTONICS HARDWARE

An optical network requires photonic devices to operate and function. This section presents some of these devices. Only devices used in this dissertation are going to be presented in this section. The physical layer and manufacturing process is beyond the scope of this dissertation. This section starts by presenting light sources used in optical networks, followed by elementary switching devices, followed by Fiber Delay Lines (FDLs). Although wavelength converters are not used in this dissertation, it is intentionally presented in order to describe some concepts demonstrated below, such as continuity constraint, and switching in wavelength domain.

2.2.1 Light Sources

In optical networks, data is modulated into an optical signal. These signals are produced by transmitters that have the ability to produce light. There are two types of transmitters (light sources): Light-Emitting Diode (LED) and Laser Diode (LD). An LEDs is a cheaper hardware compared to a LD, however, they produce a wider band compared to LD. Hence, the number of channels in LD is greater than it is in LED. LDs are used to provide narrow-band channels that are found in 1500nm bands, therefore, using an LD transmitter is suitable for WDM and DWDM networks. In addition, the amount of attenuation in LD over distance is smaller than it is in LED.

The bottom line is, LED is suitable for short distance applications, such as Local Area Network (LAN) and Metropolitan Area Network (MAN) and moderate bandwidth requirements. While LD is suitable for high bandwidth application and long distance networks such as WAN and cross ocean fibers. In this dissertation, an abstract light source is used regardless of its type.

2.2.2 Switches

A photonic switch is a device that redirects optical signals from one input port to a specific output port. There are several types of switches with different technology that have been implemented to serve this purpose, such as: Micro-Electro-Mechanical System (MEMS) switches, Thermo-Optic switches, Semiconductor Optical Amplifier (SOA) , and electro-Optic switches [36] [37]. This dissertation focuses on electro-optical switches that are made of $LiNbO_3$. $LiNbO_3$ switches provide higher speed than other types of switching including MEMS. For instance, $LiNbO_3$ have been reported to provide a 1 nm switching speed. Meanwhile, a typical switching speed for MEMSs switching is around 1 msec. As switching speed gets faster, the amount of guard-time (subsection 2.3.1) between channels get shorter. When guard-time is shorten, the medium is better utilized. On the other hand, when the switching speed is slow, the guard-time (unused spectrum) must be longer in order for the switch to change it's current state without interrupting the transmitted data.

$LiNbO_3$ falls into an active switch category, where the term "active" is used when an external input (voltage) is required to change the state of a switch. $LiNbO_3$ switches are also known directional couplers, because the couple a light signal from one input port and direct it to a desired output port. Directional couplers are designed with various numbers of input and output ports. However, a 2x2 directional coupler (2 input ports and 2 output ports) is used in this dissertation. The switch has two switching states: BAR and CROSS. In the BAR state, Figure 12 (a), an optical input signal arriving at the upper input port, exits at the upper output port, while an optical input signal arriving at the lower input port exits at the lower output port. In the CROSS state, Figure 12 (b), an optical input signal arriving at the upper input port, exits at the lower output port, while an optical input signal arriving at the lower input port exits at the upper output port. The idle state for a directional coupler is CROSS. To change the switching state for a directional coupler, an external input signal ($\approx 5V$) is applied that would change the state to BAR as long as the voltage is applied. Once the voltage is released, the coupler returns to its idle stage. Theoretically, once voltage is applied, the switching state is changed instantaneously; however, the time it takes the switch to change its state, once voltage is applied, is known as switching speed. The faster the switching speed, the better the switch. Fast switches are expensive, but they are always preferred. A one nanosecond switching speed for a directional coupler has been reported [4].

In this work, three different directional couplers are used: 2x2, 1x2 and 2x1. The terms: switch, splitter, combiner are used, respectively, to refer to each type of directional coupler. Note that all of the three different type of switches are active switches. The BAR and CROSS states for a 1x2 directional coupler are demonstrated in Figure 13. In addition, Figure 14, demonstrates the BAR and CROSS state for a 2x1 directional coupler.

In a perfect directional coupler, input and output signals are identical ($P_{in} = P_{out}$). However, the existing directional couplers are not perfect, and directional couplers are not identical. Since directional couplers are manufactured by different suppliers, each product has its own technical specifications. For the purpose of this dissertation, limited technical specifications are considered, including: crosstalk (X), loss (L), coupling loss (W), and

[4]JDS Uniphase Corporation. 2X2 High speed lithium Niobate Interferometric switch, product number: 10022467.

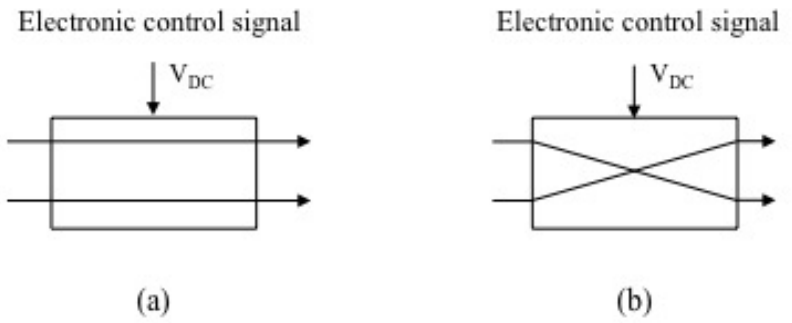


Figure 12: A schematic diagram for a directional coupler switch in (a)BAR (b)CROSS state

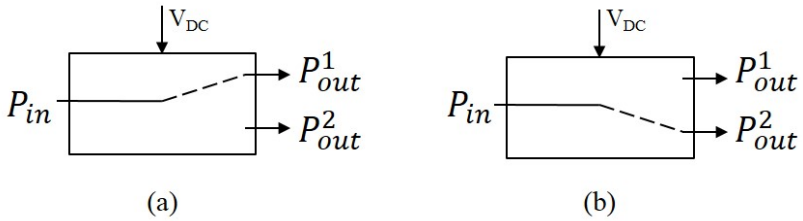


Figure 13: A schematic diagram for a splitter in (a)BAR (b)CROSS state

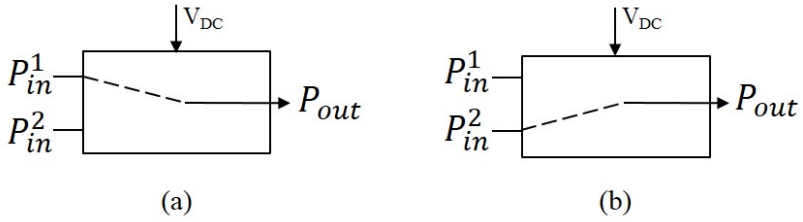


Figure 14: A schematic diagram for a combiner in (a)BAR (b)CROSS state

switching speed (v_{sw}).

$LiNbO_3$ are fast switches but they produce higher crosstalk compared to other switches such as MEMS. But the advantage of higher switching speed overcome the drawback of the high crosstalk. In a 2x2 directional coupler that is made of $LiNbO_3$, the input signal at a given input port should exit the switch at the desired output port in full power strength. The coupling ratio (α) in a perfect directional coupler is $\alpha = 1$. However, fraction of the input signal power $1 - \alpha$ is leaked to the undesired output port. If a single input is used at a time (dilated switch, discussed in chapter 6), the amount of leaked power can be ignored or eliminated using proper optical filters. But when two simultaneous signals are inserted to a directional coupler, a power signal at input port 1 (P_{in}^1) and a power signal at input port 2 (P_{in}^2), each input signal will leak a fraction of its signal power to the undesired port causing noise to signal. This type of noise is known as crosstalk (X). Crosstalk is a negative number in decibels (dB). An acceptable value for a given directional coupler is -25 dB. As the number of switches in the path increases, the number of switches in path (k), the amount of crosstalk is accumulated and the received signal is corrupted.

In addition to crosstalk, when an optical signal passes through a photonics device, the signal's power strength is degraded due to the nature of the materials and the quality of the manufacturing of the device. The amount of loss L to the input signal's power strength resulting from passing an optical signal into a directional coupler is small. An acceptable loss value for a directional coupler is .25 dB per switch. Figure 15 demonstrates the power signal at output port 1 (P_{out}^1) and the power signal at output port 2 (P_{out}^2), when 2 simultaneous input power ,P_{in}^1 and P_{in}^2, is inserted to a directional coupler [35] and it is given by:

$$P_{out}^1 = l \cdot P_{in}^1 + l \cdot x \cdot P_{in}^2 \tag{2.3}$$

$$P_{out}^2 = l \cdot P_{in}^1 + l \cdot x \cdot P_{in}^1 \tag{2.4}$$

As the light signal passes through multiple switches in a fabric k, the amount of crosstalk is increased. The output signal suffers from accumulated undesired crosstalk. The estimated

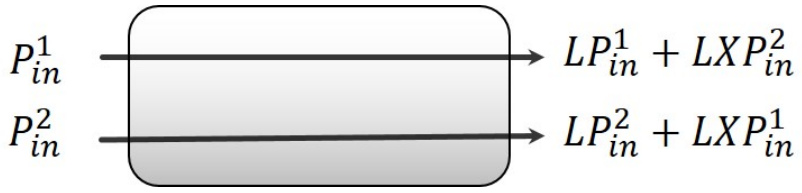


Figure 15: The effect of loss and crosstalk on directional couplers on the input power

signal-to-crosstalk ratio (SXR) (in dB) for a switching fabric with k is given by [35]:

$$SXR \approx -X - 10log_{10}k \tag{2.5}$$

In this dissertation, the simple power analysis provided by [35] in equations 2.3, 2.4 and 2.5 will be used to evaluate the output signal power strength.

2.2.3 Fiber Delay Lines

FDLs are used to overcome the absence of RAM. An FDL is created by a spool of fiber with a precise length. The length of fiber in the spool equals the time required for a timeslot to travel inside the fiber at the speed of light in the fiber. The duration of travel could be single, multiple, or fraction of a duration of a timeslot, depending on the purpose of the FDL. There are two different type of FDLs: FF-FDL, Figure 16 (a) and Feed-Back Fiber Delay Line (FB-FDL), Figure 16 (b). The differences and similarities between FDLs is discussed below and summarized in Table 3.

- **Implementation**: FF-FDL is implemented by inserting an FDL between two switches (generally directional couplers). In this implementation, light signals pass through a switch only once. In addition, the signal never exits an FDL before it passes through the loop completely. At the ingress switch, Figure 16 (a), when the switch is set to BAR, the light signal from the upper input enters the delay stage, while the signal from lower input skips the delay stage. On the other hand, when the switch is set to CROSS, the lower input's signal is delayed, while the uppers input signal skips the delay stage. The

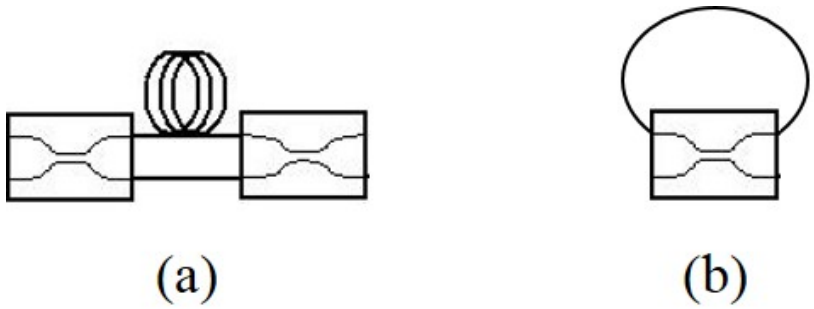


Figure 16: (a) Feed-Forward Fiber Delay Line and (b) Feed-Back Fiber Delay Line

fabric controller is responsible for changing the switches' states.

On the other hand, FB-FDL is implemented by inserting an FDL on top of a 2x2 directional coupler. In this implementation, the light signal circulates in the FDL passing through the switch in every circle. The signal always enters a FB-FDL from the lower input of the switch, Figure 16 (b). If the signal is to be delayed, the switch is set to CROSS, otherwise, the switch is set to BAR. Once the light signal is inserted completely into the FDL, the switch is set to BAR if delay is required, otherwise, the switch is set to CROSS, forcing the signal to exit the FB-FDL. The fabric controller is responsible for changing the switches' states.

- **Spool size**: As the number of timeslots per frame increases, FF-FDL requires a massive amount of fiber compared to FB-FDL (discussed later in subsection 3.1.1). This is an advantage of FB-FDL over FF-FDL.

- **Number of switches**: The number of switches in FF-FDL scales better than FB-FDL as the number of timeslots per frame increases. For instance, to interchange 256 timeslots with frame integrity (discussed in subsection 2.3.3) the required number of switches is eight using FF-FDL. On the other hand, in FB-FDL, every timeslot must have a dedicated switch to be able to provide the interchanging process without blocking. Notice that the cost of a switch is higher than the cost of a spool of fiber. Therefore, the cost of a complete switch (switches and FDL) is cheaper when FF-FDL is used.

- **Delay**: FF-FDL provides a delay equals to an integer multiple of timeslot duration. On the other hand, FB-FDL provides a delay equal to fraction, or multiple, of a timeslot duration. The advantages and disadvantages of each type depends highly on the application (below).
- **Applications**: Because FF-FDL provides an integer multiple timeslot duration, it is ideal for interchanging timeslots with each other. Meanwhile, FB-FDL is also capable of interchanging timeslots but at the expense of a power penalty. FB-FDLs are used to solve unsynchronized timeslots in synchronized networks because they have the ability to delay timeslots with fraction, or multiple, of timeslot duration [38]. In addition, FB-FDL is used for contention resolution in OSSN [39]. Lastly, FB-FDLs are used when priority queue algorithms are required [40], while FF-FDLs do not have this ability because they form a First In First Out (FIFO) queue.
- **Frame Integrity**: Switching timeslots in a time domain with frame integrity is a complicated process, subsection 4.2.1. This is achieved using FF-FDL, because once the light signal is inserted into an FDL stage, it must finish the complete delay before it exits the FDL, while the controller remains idle until it is time to change the switching state of the egress port. On the other hand, in FB-FDL the controller must change the state of the switch every time before the light signal passes through the switch. Hence, as the number of switches increases, a single controller will not be able to keep up with changing the state of every switch.
- **Signal-to-Noise Ratio (SNR)**: FF-FDL was invented first to improve a signal's SNR . The light signal in a 2x2 switch with FDL suffers from power penalty caused by crosstalk from other signal that share the same switch, and loss from the nature of optics and switch. SNR analysis in FB-FDL is found in [41].
- **Crosstalk**: Unless a switching fabric is dilated, two simultaneous optical signals are allowed to share the same switch if, and only if, they both require the same switching state. Due to the imperfection of the directional couplers, signals are leaked to the undesired port, causing noise to the desired signal in the form of crosstalk. As the light signals circulate in a FB-FDL, new incoming light signals share the directional coupler that adds crosstalk to the current signal. More details about crosstalk in FF-FDL and

FB-FDL is found in [41].

- **Power Consumption**: As discussed earlier, the idle state for a directional coupler is the CROSS state. Hence, in FF-FDL the voltage at the ingress switches is applied once before the signals enter the FDL; then, once the optical signal passes the switch, the applied voltage is removed. At the egress switch, the applied voltage used to change the state is applied before the signal exists the switch. No voltage is applied to either the ingress nor egress switch while the signal is delayed in the FDL.

  On the other hand, in FB-FDL once the signal enters the FDL, a DC voltage is applied to maintain the signal inside the loop by retaining the switch to the BAR state (not the idle state). Therefore, as the number of timeslots per frame increases, the duration of the applied voltage increases.

Note that attenuation must also be considered; however, because the loss caused by attenuation is small, it is ignored in both FDLs.

2.2.4 Wavelength Converters

In WDM networks, data is carried over multiple wavelengths. In some scenarios, data should be switched (shifted) from one wavelength to another, (discussed later in subsections 4.1.2 and 2.3.4), and it cannot be switched without WLCs. WLCs are expensive hardware devices that switch data from one wavelength to another. Because they are expensive devices, many research papers are heading towards eliminating WLCs while providing nonblocking switching fabrics (or networks). There are many benefits of using WLCs in WDM networks, including, but not limited to: improving utilization, improving P_b and improving cross carriers switching [42]. The presence of WLCs in a WDM network improves the overall utilization by relaxing the continuity constraint, subsection 4.1.2. In addition, the use of WLCs reduces the probability of blocking in the network by switching data to different wavelengths when blocking is likely to occur. Lastly, the Internet is built out of multiple carries connected with each, but due to the competitive market, carriers do not share their network statues.

WLCs are classified based on many criterion, but the range of conversion and technology used to build the hardware are discussed here. In WDM, there is the number of wavelengths

Table 3: Comparison between FF-FDL and FB-FDL

	FF-FDL	FB-FDL
Spool size	Large	Moderate
Delay	One or multiple of duration of a timeslot (T_s)	One, multiple or a fraction of T_s
Application	Timeslot interchanging	Timeslot interchanging Synchronization Contention resolution priority algorithm
Frame integrity	Preserve frame integrity while interchanging	Preserve frame integrity while interchanging is challenging
SNR	High	Low
Crosstalk	Low	High

per fiber (λ) per fiber. Input data is carried over input wavelength index (λ_i) where i is input wavelength index, which is a natural number between 0 and $\lambda - 1$. On the other hand, the data is carried over output wavelength index (λ_o) after it is converted (or not), where o is the output wavelength index, which is a natural number between 0 and $\lambda - 1$. WLC could have one of three different ranges of conversion:

- **Full range WLC**: This type of hardware can convert data from any input wavelength λ_i to any output wavelength λ_o. Such a converter is expensive because it contains multiple photo-detectors at the input side and multiple transmitters (lasers or LEDs) at the output side. On the other hand, this type of hardware provides a wide range of conversion that increases the flexibility of the network and reduces the network's P_b.
- **Fixed range WLC**: This type of hardware can convert data from an input wavelength λ_i to an output wavelength λ_o, only. This type of WLC is considered the cheapest converter, because it consists of a single photo-detector and a single output transmitter. Some full range WLCs are built out of multiple fixed WLCs and a space switching fabric. This type of hardware helps increase network utilization, improve P_b at a reasonable cost. However, network management becomes challenging.
- **Limited range WLC**: A limited range wavelength converter is a hybrid solution between full range and fixed range WLCs. They are built at reasonable cost and a wider range of conversion degree. The term "conversion degree" is defined as the number of output wavelengths that a WLC can switch. For instance, a conversion degree of 4 means that the hardware can convert an input wavelength λ_i to four different output wavelengths λ_o. Many researchers suggest that limited range WLCs can perform almost identical to full range WLCs [43] [44].

The second WLCs classification is based on the technology used to convert data from λ_i to λ_o. There are four different technologies to build a WLC: Opto-electronic, optical gating, interferometric and wave mixer. Only opto-electronic WLC is turned to a commercial product, while the rest remain in the research phase. Opto-electronic WLCs involves O/E/O conversion, which is not preferred in AONs. In addition, such types of converters are slower than the rest of the type because of the O/E/O conversion. An opto-electronic WLC can

perform full, fixed or limited ranges of conversion. In fact, opto-electronic WLCs are known as fixed or tuned WLCs. Tuned WLCs have the ability to change the output wavelength to number of wavelengths.

2.3 SWITCHING

The term "switching" is a well-known term in telecommunication systems. In fact, it is the core concept of modern telecommunication systems. It is the process of moving data from a source to a desired destination. Optical switching is used to distinguish the optical domain from the electronic one. However, in some literature, the word optical is used to describe AON that do not involve O/E or E/O conversion; this dissertation uses this definition. The following subsections describe switching in three different domains in greater detail. But before discussing the switching, guards in switching is explained.

2.3.1 The Guards

In telecommunication systems, guards are spacing left intentionally between two adjacent channels. In optic, there are two different types of guards: a guard band and a guard time. A guard band is the spacing between two adjacent pulses. A guard time is a spacing left between two adjacent timeslots. These guards improve telecommunication system performance. Guard bands improve system performance by reducing pulse dispersion. Whereas, guard time improves the system performance by providing synchronization tolerance [42] and provides a safe area for timeslot switching. The major disadvantage in leaving unused spacing in the medium results in reducing medium utilization.

2.3.2 Switching In Space Division

In optical domains, space switching is the process of providing a physical connection between one fiber and another such that two nodes can exchange information with each other. As

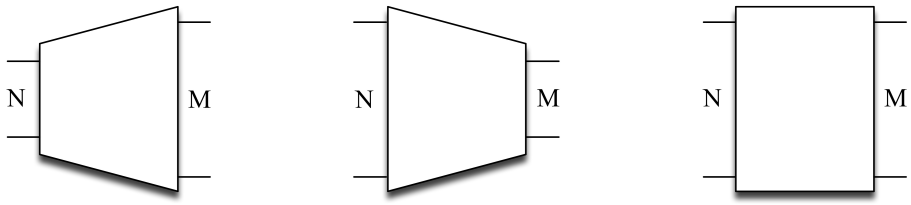
mentioned in section 2.1.1, this dissertation discusses only point-to-point (unicast) connection. Most space switching fabrics support multiple simultaneous unicast connections.

Building a complete optical space switching fabric is achieved by wiring a number of switching elements together with optical waveguides in a spacial pattern to form a complete fabric. There are a number of different categories of switching elements(discussed in subsection 2.2.2) but building all optical switching fabrics out of electro-optics switches made out of $LiNbO_3$ provides ultra-fast switching speeds.

A complete space switching fabrics are catogries on many criteria [45] [46] [42] including, but not limited to: number of sides, number of input/output ports, number of paths, number of stages, blocking condition and type of switches. Three of the space switching fabrics' categorizations are tightly related: number of input/output ports, blocking condition, and number of paths. When the number of input ports is less than the number of output ports (number of space input ports (N) < number of space output ports (M)), the fabric is said to be an "expander" and it does not block, Figure 17 (a). In contrast, when the number of input ports is greater than the number of output ports ($N > M$), the fabric is said to be a "concentrator" and it blocks, Figure 17 (b). Lastly, when the number of input equals the number of output ports ($N = M$), the fabric is said to be a "distributor" and it may or may not block depending on the fabric control algorithm, Figure 17 (c). The number of alternative paths is essential to provide a non-blocking switching fabric. In this work, the focus will be on two-sided, multiple input/output ports, multi-paths, multi-stage, nonblocking, and fabrics made out of $LiNbO_3$ as switching elements.

The most popular multi-stage space switching fabrics are Clos [47] and Beneš [48] switching fabrics. In fact, Beneš switching fabric is a special case of Clos, where $N = M = 2$. Because the proposed algorithm for space/time switching fabric (Chapter 8) is transformed from Beneš fabric, this subsection will only discuss Beneš switching fabric.

Beneš switching fabric is created using simple 2x2 switches as switching elements. Any switching element technology (subsection 2.2.2) can be used to create a complete Beneš network, however, $LiNbO_3$ directional coupler is widely used in optical networks because of its fast switching speed and low power consumption. The process of building a complete N x M fabric is beyond the scope of this dissertation. A complete 4x4 (N x M) Beneš fabric is



(a) Expander (b) Concentrator (c) Distributer

Figure 17: Space switching fabric categories based on the number of input/output ports

presented in Figure 1, while Figure 2 shows a complete 16x16 Beneš fabric. There are some facts about Beneš fabric that must be stated before proceeding [35] :

- Beneš fabric is a distributer where $(N = M)$.
- Since a number of space input ports (N) equals a number of space output ports (M), then Beneš switching fabric is rearrangeably nonblocking, section 2.4.
- The total number of stages (C) is given by :

$$= 2Log_2N - 1 \tag{2.6}$$

- The total number of rows (R) is given by :

$$= \frac{N}{2} \tag{2.7}$$

- The total number of switches in path (k) in the fabric is given by :

$$\begin{aligned} &= NLog_2N - \frac{N}{2} \\ &= NLog_2N - R \end{aligned} \tag{2.8}$$

- Every signal passes through C switches before leaving the fabric.
- The SXR on an optical signal that passes through k switches is given by :

$$\begin{aligned} &= -X - 10log_{10}(2log_2N - 1) \\ &= -X - 10log_{10} \cdot C \end{aligned} \tag{2.9}$$

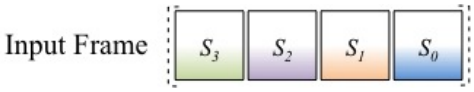


Figure 18: Input frame with 4 timeslots/frame

- The insertion loss (A) on the optical signal, when exiting the fabric with L and coupling loss (W), is given by:

$$\begin{aligned} &= (2log_2 N - 1)L + 2W \\ &= C \cdot L + 2W \end{aligned} \tag{2.10}$$

2.3.3 Switching In Time Division

Switching channels in TDM networks in a time domain is the process of rearranging the order of timeslots within a frame. This process has been practiced for many years in electronic networks, however, the process is challenging in optics. Switching two timeslots with each other in an electronic domain requires a temporary storage (memory) to perform the conventional swap operation. But swap operation is challenging in optical domains due to the absence of "optical" RAM in optical networks. Several optical Timeslot Interchangers (TSIs) have been proposed to provide time switching using photonics technology. However, they are yet to be commercial products.

TSIs are switching fabrics designed for TDM networks[5]. There are many ways to build TSIs, subsection 3.1.1 demonstrates different technologies used to build TSI in greater detail. This subsection will discuss switching time channels (timeslots) in time domains using TSIs that are built using FF-FDL (see subsection 2.2.3).

Interchanging (rearranging) the order of timeslots within a frame is achieved using two approaches: interchanging timeslots in time domain **with** frame integrity and interchanging timeslots in time domain **without** frame integrity. Frame integrity is discussed in great

[5]For the purpose of this subsection, switching in time domain for synchronized TDM network is discussed only.

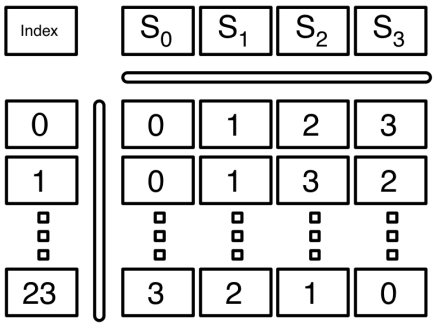


Figure 19: Sample of SWA matrix for S=4

details chapter 4.2. Assuming a synchronized TDM network with four timeslots per frame, timeslots at the input frame are ordered $\{S_3^{in}, S_2^{in}, S_1^{in}, S_0^{in}\}$ as it appears in Figure 18. The order of timeslots at the output frame depends on the Switching Assignment (SWA). For S timeslot/frame, the total number of switching assignments (SWA) per frame when all frames in the system have the same SWA is given by:

$$A = S! \tag{2.11}$$

That is the number of permutation in the order of timeslots in a frame. But, if every frame in the system has an independent SWA, then the value of SWA is given by:

$$A = (S!)^F \tag{2.12}$$

Switching assignment indexs are labeled swa_i, where $i = \{0,...,A-1\}$. Figure 19 is a sample from a complete SWA matrix for $S = 4$. As shown in the figure, SWA=24 and swa_i ranges from 0 to 23. Cyclic notation is used to represent SWA, such that swa_1 is represented as (0)(1)(23) and is read as S_0^{in} and S_1^{in} remain in their places, while S_2^{in} is interchanged with S_3^{out}, and S_3^{in} is interchanged with S_2^{out}. The process of interchanging timeslots with and without frame integrity is discussed in subsection 4.2.2

2.3.4 Switching In Wavelength Division

In WDM, switching in wavelength domain is the process of shifting data from one wavelength to another. This process requires WLCs, which are expensive devices and still in the development stages. As mentioned in subsection 4.1.2, switching in wavelength domain depends on the continuity constraint requirement. Switching in wavelength domain is beyond the scope of this dissertation.

2.4 BLOCKING

A switching fabric is said to be a blocking fabric if the probability of blocking is greater than zero ($P_b > 0$). On the other hand, a switching fabric is said to be a nonblocking fabric if the probability of blocking equals to zero (Pb = 0). However, building a non-blocking system with firm zero probability of blocking is not economical. In the optical world, a small probability of blocking (10^{-12} or less) is sometimes considered nonblocking. This criteria is the most used criteria to describe any switch fabric. Most proposed switching fabrics are described as nonblocking fabric associated with other adjective such as economical, fast, scalable, etc. Two different types o blocking : Internal blocking and network blocking.

2.4.1 Internal blocking

Internal blocking occurs when an idle (unused) input port cannot establish connection with any idle (unused) output port. Note that a new connection cannot be establish with a non-idle port in circuit switched network. This scenario is known as "busy" case.

There are three different type nonblocking fabric [42] [46] wide sense, strict sense and rearrangeably nonblocking switching fabric. In a space switching fabric, let's assume that the number of input ports is N and the number of output ports is M then: the switching fabric is said to be a strict sense nonblocking if there is always a path between any idle input and idle output regarding of the status of the network. Meanwhile, the network is side to be wide sense nonblocking if there exist an algorithm (or rules) to establish any new connection.

Lastly, the network is said to be a rearrangeably nonblocking if establishing a new connection requires the network to rearrange the existing connection in order to accommodated the new one.

The most number of switching elements, which determines the cost of the switching fabric, is found in strict sense nonblocking fabric. Whereas, the least number of switching fabric is found on rearrangeably nonblocking fabrics. However, a control algorithm for a rearrangeably nonblocking fabric is the most complex switching fabrics' control algorithm.

2.4.2 Network blocking

building a network out of multiple nonblocking switching fabrics, does not result on a nonblocking network. Switching fabrics in the network are connected to independent nodes, each node generates random traffic. Establishing a new connection from a given source to a given destination may require rearranging lots (or all) of existing connections. In addition, traffic pattern of the network is not predictable. For these reason and more, network blocking is more complex than internal blocking. In this dissertation, only internal blocking is discussed.

3.0 RELATED WORK

As mentioned in subsection 1.4, two different control algorithms for two different switching fabrics are proposed in this work. The first proposed control algorithm is a time switching fabric (also known as timeslot interchanger), with active FF-FDL and frame integrity constraint. The second proposed control algorithm is a space/time switching fabric with active FF-FDL and frame integrity constraint. However, before presenting the related work to the proposed control algorithms, a literature review for related time and space/time switching fabrics will be proposed.

3.1 SWITCHING FABRICS

There is an enormous amount of optical switching fabrics that have been presented over the history of photonics switching. They can be developed to satisfy different performance metrics, fabric size and footprint, different numbers of stages, and different domains or technologies. This dissertation focuses on switching fabrics and their control algorithms that have been built to switch time channels in time and space/time domains using FF-FDL and directional coupler as the primary switching components for the switching fabric. Moreover, this study focuses on blocking only as the performance metrics. The next two subsections present a number of related works to timeslot interchangers followed by a number of related works to space/time switching fabric, respectively. Later on, a number of closely related control algorithms for time and space/time switching fabrics are presented, respectively.

3.1.1 Time Switching Fabric

Timeslot interchangers are used in optical networks to rearrange the order of timeslots inside frames. The operation of rearranging timeslots inside a frame is discussed in great detail in subsection 2.3.3. Optical switching returned to being an interesting area for research as cloud computing and data centers have become popular.

As discussed earlier, the hardest part of building a timeslot interchanger is to find a RAM substitute that can store light beams. This dissertation classifies timeslot interchangers into three main classifications: timeslot interchanger using FF-FDL, timeslot interchangers using FB-FDL and timeslot interchanger using other delay technologies. For the purpose of this research, timeslot interchangers using FF-FDL, will be presented in-depth because the proposed control algorithms are built to control switching fabrics with FF-FDL. The other two classifications will be briefly mentioned.

3.1.1.1 Photonic Timeslot interchanger with feed-forward fiber delay lines:

Timeslot interchangers using FB-FDL were proposed during the early stages of optical switching, but the first proposed implementation of such fabric was in [49]. The amount of crosstalk and SNR discovered was high and cannot be neglected. Then, FF-FDL was first introduced [41] as a solution to overcome the accumulated crosstalk and SNR caused by FB-FDLs. FF-FDL does not eliminate crosstalk and SNR; thus using dilation, where no two signals arrive to a DC switch simultaneously, will further reduce the amount of crosstalk and SNR as suggested in [50]. The downside of this practice is adding an additional number of switches to the switching fabric.

There are many research papers that advocate against the usage of TSI in optical networks due to cost, latency [12], complexity [51] and power loss; however, it was proven that TSI reduce the P_b for different network topologies [52] [53]. The rest of this subsection will go over previously proposed photonics timeslot interchangers. It is important to mention that only switching fabrics that depend on FF-FDL to perform the TSI operation will be discussed.

1. Single Stage TSI with FF-FDL Single stage TSI is one type of a time switching fabric

that uses FF-FDL [54][55][56]. Single stage TSI consists of parallel FF-FDLs, each corresponding to a different delay as it appears in Figure 20. The top delay row has a delay equal to the minimum delay required to delay a timeslot, while the last row has a delay equal to the maximum delay required to delay a timeslot. This type of TSI is easy to implement and easy to control. On the other hand, a study suggest that controlling a single state TSI is a complicated process [55]. In my opinion, single state control algorithms are easy to implement because only input space switches require control signals (active switches), while, output space switches stage can be simple passive space switches that direct any incoming light signal to the output port of the switching. Single stage TSIs with this type of configuration are nonblocking switching fabrics [55]. Unfortunately, single stage TSIs are not recommended due to the huge amount of fiber required to build the parallel delay lines. In TDM over WDM network, single stage TSI does not scale well ,as the number of required stages increases with the number of wavelengths [56]. To better visualize the difference in the amount of fiber required by single stage TSI and compare it with the multi-stage TSI (discussed later in this section) using the results from [57] and subsection 2.3.3, lets assume that delaying each timeslot by the duration of one timeslot requires 1 km of fiber. In addition, lets assume that frame integrity is considered. Then, the required fiber to delay any timeslots without blocking in single stage TSI (Figure 20 versus multi-Stage TSI (Figure 45) in the present of frame integrity is demonstrated in Table 4. The first column represents the number of timeslots per frame. The second column represents the maximum delay required to delay any timeslot in a frame with frame integrity consideration δ_{max}. The third and fourth columns are used to compute the number of stages in the single stage TSI (represented in parallel rows, as it appears in Figure 20) and the length of fibers required for FDLs. In multi-stage TSI, the number of stages required is more complicated than single stage. It is discussed in great detail in the next subsection. Note the number of stages calculated using the result from [57]. The length of fiber required to build a nonblocking multi-stage TSI is presented in column six. The difference between the length of fiber for a single stage and multi-stage TSI is presented in column number seven. Lastly, the amount of fiber saved is presented in the last column.

Table 4: A comparison between single stage and multistage TSI

#TS/Frame	δ_{max}	Single Stage TSI		Multi Stages TSI		Difference (Km)	Difference (%)
		# Stages	Fiber length	(Stage/Delay Element (DE)) x (#DEs)	Fiber length		
2	3	3	6	2x1	3	3	50%
4	7	7	28	3x2	14	14	50%
8	15	15	120	4x3	45	75	62.5%
16	31	31	496	4x4	124	372	75%

Single stage TSI can be either the switching fabric, as discussed above, or could be part of a switching fabric. For example, a label switching timeslot interchanger using single stage TSI was proposed in [2]. The proposed switching fabric performs switching in three domains (space, time and wavelength). The motivation behind the proposed fabric is scalability, low power penalty and the ability to reprogram the fabric as needed. However, the amount of hardware required to build the fabric is enormous, Figure 21. In addition, the existence of two different types of wavelength converters (fixed and tunable) makes the cost of building the fabric expensive [51]. Moreover, it was proven that introducing wavelength converters to a time switching fabric does not have a significant improvement compared to switching in time domain only [44] [58] .

2. Multi-Stage TSI with FF-FDL

   The motivation beyond introducing multi-stage TSI with FF-FDL is to reduce crosstalk and SNR caused by an accumulation of light signals repeatedly passing through the same switch [41] and to reduce the amount of fiber loops used in the timeslot interchanger. A multi-stage TSI with FF-FDL is a single space input/output port switching fabric with a number of FF-FDL inserted between 2x2 switches. For the purpose of this work, time channels assumed to arrive to a TSI synchronized (if not, a synchronization stage must be placed before the input port [59]) and aligned to avoid switching in the middle of a timeslot. There are two basic architectures for building multi-stage TSI with FF-FDL: Thompson's [41] and Hunter's [3] architecture. Both serve the same purpose with

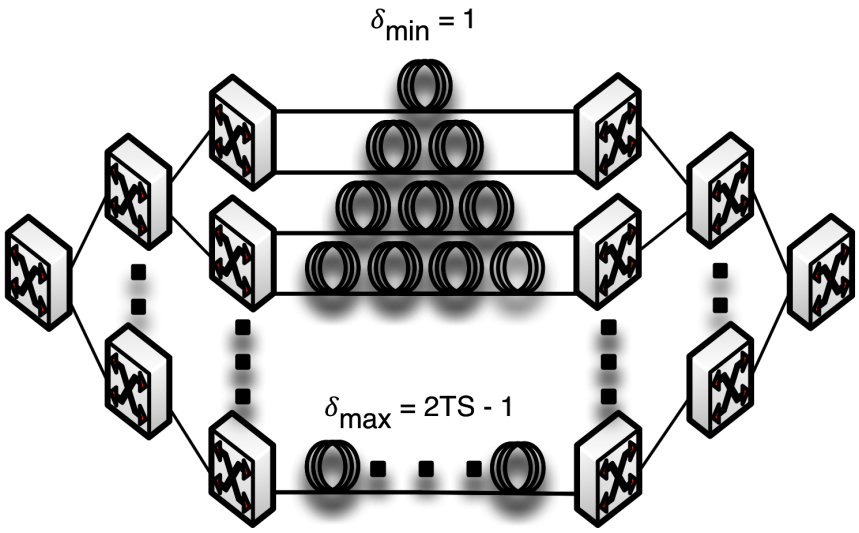


Figure 20: Single Stage TSI

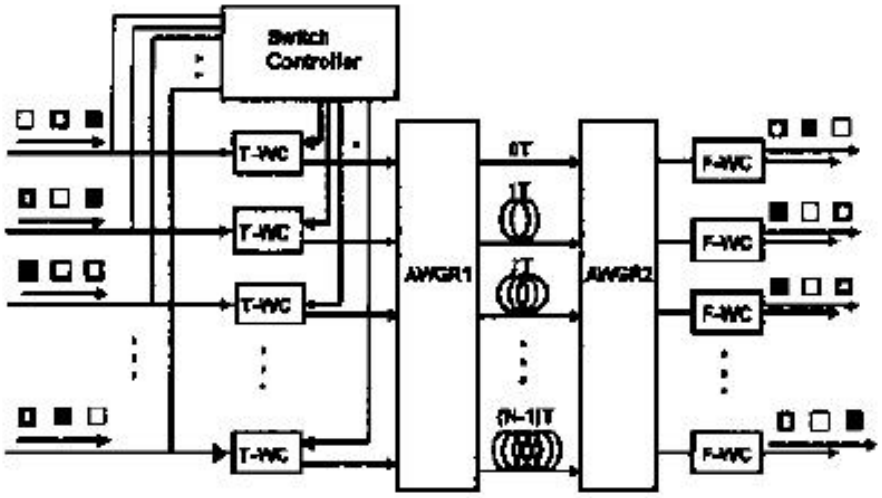


Figure 21: A three division switching fabric using single stage TSI [2]

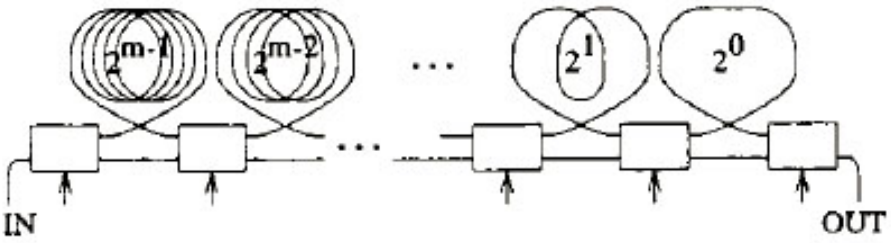


Figure 22: Thompson general TSI

Figure 23: Hunter general TSI

some common features. They both use FF-FDL and both consider frame integrity. The difference starts with the number of switches in each architecture, followed by the number of stages and the maximum delay required. Table 4, compares Thompson's and Hunter's TSI model. For better visualization, let us assume that the number of timeslots per frame is equal to 16. Hence, below each formula, there is an example for the formula when S = 16.

The maximum delay required to interchange the first timeslot with the last is less by a duration of one timeslot for Thompson's architecture compared to Hunter's. The difference is not significant, however, the variation between the number of stages in Thompson's and Hunter's is significant. The impact of reducing the number of stages results in reducing the number of switches used to build a complete TSI. As the light signal passes through switches, the signal suffers from loss, crosstalk, SNR and attenuation. Hence, reducing the number of switches that the signal passes through results in improved received. Thompson's model uses fewer switches and fewer stages as it appears in the example.

Table 5: A comparison between two TSI's models

	Thompson's Model [41]	**Hunter's Model [3]**
Figure #	22	23
FDL Type	FF-FDL	
Frame Integrity	Switching with frame Integrity	
Frame Delay	$\delta_{frame} = S$	$\delta_{frame} = \frac{1}{12}(13S - 16)$ if S is power of 4 $\delta_{frame} = \frac{1}{12}(13S - 8)$ if S is not power of 4
δ_{frame}	$\delta_{frame} = 16$	$\delta_{frame} = S$
Maximum Delay	$\delta_{max} = 2S - 1$	$\delta_{max} = 2S$
δ_{max}	$\delta_{max} = 31$	$\delta_{frame} = 32$
Number of Stages	$K = log_2 S + 1$	$K = 2log_2 S + 1$, for $S < 32$ $K = 2log_2 S + 3$, for $S >= 32$
K	K=5	K=9
Number of Switches	$SW = log_2 S + 2$	$SW = 2log_2 S + 2$, for $S < 32$ $SW = 2log_2 S + 4$, for $S >= 32$
SW	SW=6	SW=10

3.1.2 Space/Time Switching Fabric

There are many ways to build space/time switching fabric. One way to build such fabric is to divide the switching fabric into stages, space and time switching stages. This approach is presented in [53] [60]. This approach requires each stage to have an independent controller. The other approach is to merge space and time switching components into single switching fabric. This practice is presented in [12] [3] [41]. The number of space channels versus time channels must be considered wisely. Increasing the number of space channels will result in adding additional switching components to the fabric, which would result in poor crosstalk, SNR and an increase in the footprint, unless expensive space hardware components are used. On the other hand, increasing the number of time channels results in signal attenuation caused by longer fiber delay lines and a larger footprint. One common practice to avoid crosstalk is to dilate the switching fabric [61]. The dilating process prohibits any two simultaneous signals from entering the same switch. However, dilation in space domain results in doubling the amount of switches [61]. On the other hand, dilation in time domain results in less than double the number of hardware [61]. Hence, it is preferable (theoretically) to increase the number of time channels rather than increase the number of space channels.

There were a limited number of works that introduced space/time switching fabrics using FF-FDL with frame integrity. The work that came closest is the one proposed regarding TDM network space/time switches [3]. The proposed a space/time switching fabric transformed from the original Beneš rearrangable nonblocking space switching fabric [48]. The study [3] focused on the number of switches, delay lines, crosstalk, attenuation and ways to improve the received signal using network dilation. However, the discussion was theoretical. The switching fabric lacked a control algorithm to operate it.

Another space/time study from a network level (higher than a switch level) was proposed in [51]. The study proposed an analytical model for probability of blocking for a space/time switching network in an optical domain. Although the proposed analysis claimed timeslot interchanging in each switching node, the study neither discussed the type of TSI nor the technology used to build the TSI. It is assumed there is a fully function TSI that would interchange timeslots without blocking. The general conclusion from that study [51] is that

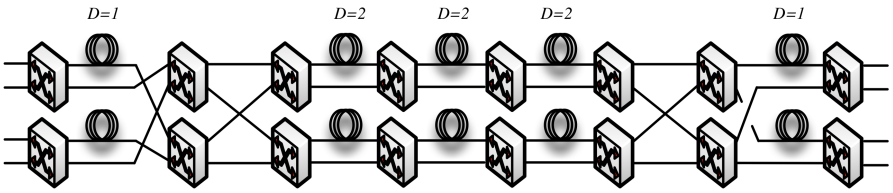


Figure 24: A 4x4 Space/Time Switching Fabric with Frame Integrity [3]

introducing timeslot interchangers to space switches improves the probability of blocking of the network (network used : unidirectional ring, bidirectional ring and mesh network), compared to space only switched networks. In addition, multi-wavelength with wavelength converters, network with space/time to perform almost identically as a space/time switched network. However, multi-wavelength adds additional expenses related to additional receivers and transmitters to the fabric.

A feedback blocking space/time switching fabric is proposed in [4]. Although, single stage TSI was stated as a non-blocking TSI, a switching fabric with a shared FB-FDL base TSI is used due what they claim ease to control, Figure 24. The results are promising (blocking probability $\approx 10^{-9}$ for offered load $\rho = .9$). The present of FB-FDL and the mesh network topology is not preferred because FB-FDL is a source undesired noises and fully connected mesh networks are expensive to build. In addition, the proposed switching fabric, by itself, is a blocking switching fabric as stated in the paper.

A three division switching fabric with space/time/wavelength channels is proposed in [62]. This study uses what they define as "tunable devices", which include TSIs and WLC. Since there is no timeslot or wavelength interchanging in this study, the result shows that the number of timeslots is the denominator. Thus, for a fixed number of links and hops, as the number of timeslots increases, the probability of blocking decreases at a given load. The absence of TSI led to $P_b = 10^{-2}$ at an offered load $\rho = .4$ compared to other studies that were able to achieve $P_b = 10^{-9}$ at $\rho = 4$ Erlang with the presence of TSI [51].

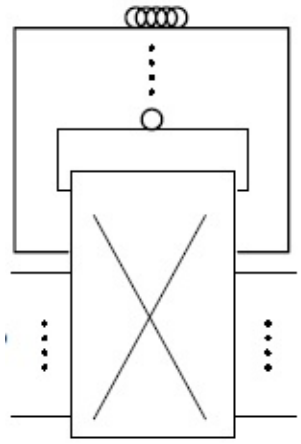


Figure 25: Space/Time switching fabric using shared FB-FDL [4]

3.2 FABRICS' SOFTWARE

There are not a lot of proposed control algorithms that have been proposed to operate similar switching fabrics, neither for TSI nor for space time switching fabric. Most of the proposed control algorithms were used to operate space only switching fabric. For example, Beneš switching fabric [48] has attracted many researchers to propose control algorithms that would work. Probably, the most common control algorithm is published in [63].

The idea behind the proposed control algorithm in this dissertation is to build up an abstract version of the switching fabric, presented in matrices, and operate simple matrices operations, such as Boolean logical operations. Surprisingly, a similar approach was used to control a space only Beneš network in [64]. During that same year, 2014, the first version of the control algorithm using a matrices operation approach [65] to control a time only switching fabric (timeslot interchanger) was published. The performance metric for this research is algorithm complexity, while this work focuses on the algorithm itself.

Timeslot interchanges are usually built with unpublished control algorithms. In some cases, TSIs are advertised to be nonblocking because the number of "servers" equals the number of timeslots. Hence, based on the blocking classifications, such systems are known as

wide sense nonblocking fabrics. Control algorithms are avoided for its complexity. Therefore, there was not any published control algorithm that was proposed to operate switching fabrics in time (Figure 44), or space/time (Figure 69) domain using multi stage FF-FDL and frame integrity constraints. A scheduling algorithm for a single stage TSI is proposed in [55]. The purpose of this algorithm is not to interchange timeslots, but to solve contention at the output port. Therefore, each FF-FDL is used as a FIFO buffer with a different delay. Unfortunately, the study avoided the non-blocking TSI with FF-FDL and used the blocking FB-FDL, due to, what they claim to be, ease of implementation. It is obvious that, for contention resolution, the delay required to buffer timeslots is not an integer number of timeslot duration, and in some cases could be a fracture of timeslot duration. This amount of delay can be achieved using FB-FDL but not FF-FDL.

Building a single stage non-blocking TSI (not for contention resolution) with frame integrity is much simpler than a multi-stage timeslot interchanger. The idea is simple: 1. Before a timeslot enters TSI, the output timeslot must be selected. 2. Once the output timeslot is selected, the delay is computed using Equation 4.1. 3. The control sends control signals to every switch in the input side to change the state of the switch (BAR or CROSS) to the desired state to direct the signal to the desired FDL. 4. The switches on the output sides can be passive switches that do not require a control signal. Hence, an incoming signal from any input port will be directed to the only output port of the switch.

A control algorithm that is used to interchange timeslot using multi-stage serial FB-FDL, Figure 25, is proposed in [5]. The performance matrix used in this study is code complexity and blocking. Although the study claims a non-blocking switching fabric, it was tested on a single frame. If two consecutive frames were sent to the input port of the fabric, the results would have been different, as discussed in section 2.4.1. There must be new switching stages added to be nonblocking for more than a single frame. In addition, the study stated that the control algorithm is less complex and uses less parts than the one published in [41]. It is well known that reducing the fabric component results in increasing the control algorithm complexity [50][66][67].

Control algorithms for space/time switching fabrics that use FF-FDL with frame integrity did not receive a significant amount of attention. Another control algorithm for a

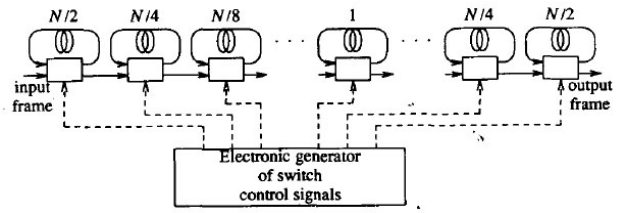


Figure 26: Muti stages FB-FDL TSI [5]

space/time single stage shared, Figure 24, optical switch, using FB-FDL is proposed in [55]. In addition to the control algorithm, three FDL assignments are proposed: 1) sequential FDL assignment, 2) multi-cell FDL assignment and 3) parallel iterative FDL assignment. The difference between timeslot assignments is beyond the scope of this work. However, the switching fabric is not a nonblocking switching fabric. This work does not consider frame integrity since maximum delay is required equal to S-1 (where S is the number of timeslots per frame).

4.0 TWO OBSERVATIONS OF TDM AND WDM

There are two important concepts that is known for many switching scholars, but they are briefly discussed in the literature. In this chapter, continuity constraint in TDM and WDM network for both single-rate and multi-rate networks is discussed in section 4.1. In the next section (section 4.2), switching (interchanging) timeslots in time domain **with** and **without** frame integrity is presented in great detail.

4.1 CONTINUITY CONSTRAINT

Continuity constraint is considered during the connection establishment phase in a circuit switched and burst switched networks, but not in a packet switched network. Mainly, it is considered in multi-hop TDM and WDM networks only. It is obvious that single-hop networks do not require switching, regardless of the multiplexing schemes. The first type of continuity constraint is associated with single-rate TDM and WDM networks, where all users have the same data rates. In these types of networks, establishing a connection requires a vacant channel from source to destination. In the presence of continuity constraint, the exact channel index must be vacant in every hop from source to destination in order to establish a connection. On the other hand, in the absence of continuity constraint, any vacant channel on every hop from source to destination is sufficient to establish a connection.

The second type of continuity constraint is associated with a multi-rate network for both TDM and WDM. In the presence of continuity constraint in a multi-rate networks, not only does the exact channel index need to be vacant on every hop, but all vacant channels in all hops must have the exact data-rate. Meanwhile, in the absence of continuity constraint

in multi-rate TDM and WDM networks, it is more complicated than single-rate networks. The rest of this subsection will explain the presence/absence of continuity constraint for single/multi-rate TDM and WDM networks in greater detail with examples.

4.1.1 Continuity Constraint in TDM network.

The presence and absence of continuity constraint in TDM networks is discussed in this part of the subsection. Continuity constraint for single-rate TDM networks is discussed first, followed by continuity constraint in multi-rate TDM networks.

- **Continuity constraint in single-rate TDM networks**

  In TDM networks, channels are represented by timeslots (subsection 2.1.2). In order to establish a connection in single-rate networks, in the presented on timeslot continuity constraint, the exact timeslot index must be vacant on every hop of the network from source to destination. However, in the absence of timeslot continuity constraint in single-rate networks, a vacant timeslot on every hop from source to destination is sufficient to establish a new connection, regardless of its index. Establishing a connection in the absence of timeslot continuity constraint, requires a hardware device that has the ability to switch the new connection from one index to another, or rearrange the existing connections' index in order to accommodate the new incoming timeslot. This device is TSI.

  The following example explains single-rate TDM networks with the presence/absence of timeslot continuity constraint. For example, Figure 27 represents a multi-hop TDM network with a single source and three destinations labeled Sink 1 through 3. There are three space (only) switching nodes labeled switch 1 through 3. Note that any connection will pass though three hops from the source to any destination. Each block on top of the links represents a time channel (timeslot) and they are labeled S_0 through S_3. Timeslots have identical timeslot durations (because it is a single-rate network). The colored blocks represent occupied (busy) timeslots, while the bright blocks presence vacant (idle) timeslot. Assuming that a connection request is sent by the source to the network controller to establish a new connection in the present of timeslot continuity

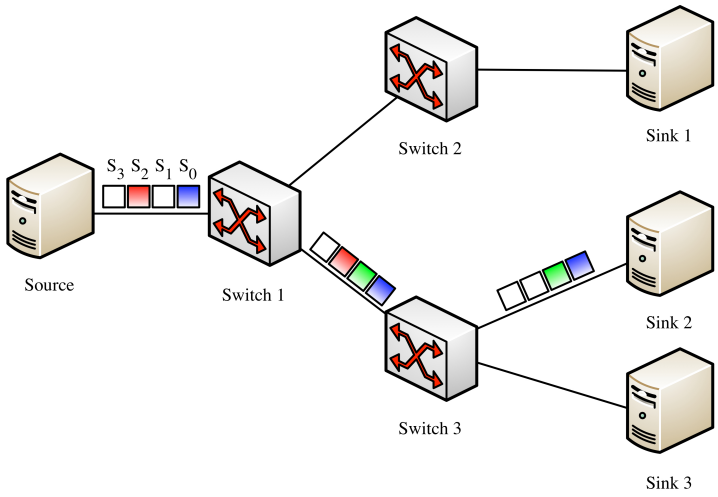


Figure 27: Timeslot continuity constraint example - 1

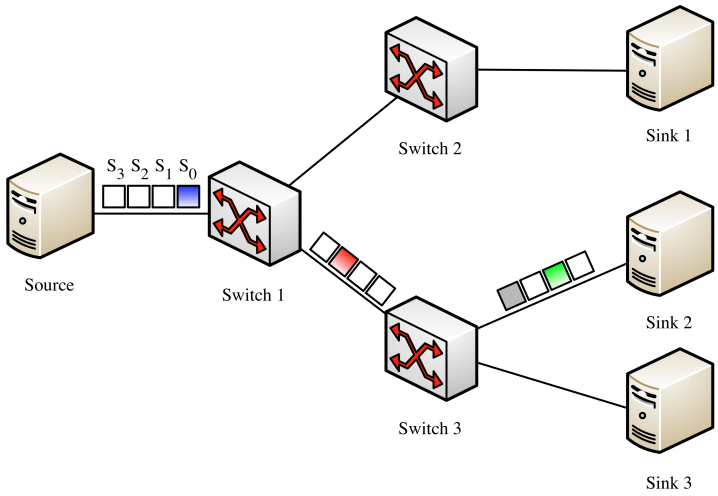


Figure 28: Timeslot continuity constraint example - 2

constraint. The controller will return timeslot index 3 (S_3) because it is the only vacant timeslot on every hop. On the other hand, using the exact network at different state as it appears in Figure 28, assuming the same above connection establishment request, the request will be blocked due to lack of resources (presented in timeslots). In figure 28, there isn't a timeslot that is vacant in all hops to establish a connection in the presence of timeslot continuity constraint. Relaxing the continuity constraint requires a TSI on every switching node in Figure 28. Hence, all switching nodes will perform as space and time switches. In fact, adding only one TSI to Switch 1 in Figure 28, allows one new connection to be added to the network, such that the new connection will occupy S_1 in the first hop (between Source and Switch 1), and at switch 1, the connection will be switched to S_0 in the next two hops (Switch 1 - Switch 3 - Sink 2). Moreover, adding two TSIs to Switch 1 and Switch 3 will allow two additional connections to the networks.

- **Continuity constraint in Multi-rate TDM networks**

  ISPs tend to provide different grades of services for different users. Usually, the higher the grade of service, the higher the cost. ISPs that use TDM in their transport network, assign extra channels (timeslot) for users with higher grades of services. The number of extra channels varies depending on the grade of services. Such networks are known as multi-rate networks. Multi-rate networks introduce more complicated continuity constraints than single-rate networks. To better understand the concept of continuity constraint in multi-rate networks, let us assume that a connection is established, if and only if, the exact data-rate is vacant in the network to accommodate the new incoming connection. Hence, if the new connection request has a lower data-rate than the vacant timeslot, the connection request will be rejected. When a new connection request is sent from the source to the network controller in the presence of continuity constraint, the control response will vary based on one of the following network existing states:

  State I : **The network controller found a vacant timeslot on every hop of the network with an identical index and identical data-rate**, Figure 29. In this case, the controller will reserve the vacant timeslots for the new connection.

  State II : **The network controller found a vacant timeslot on every hop of the network with an identical index and different data-rate**, Figure 30. In this

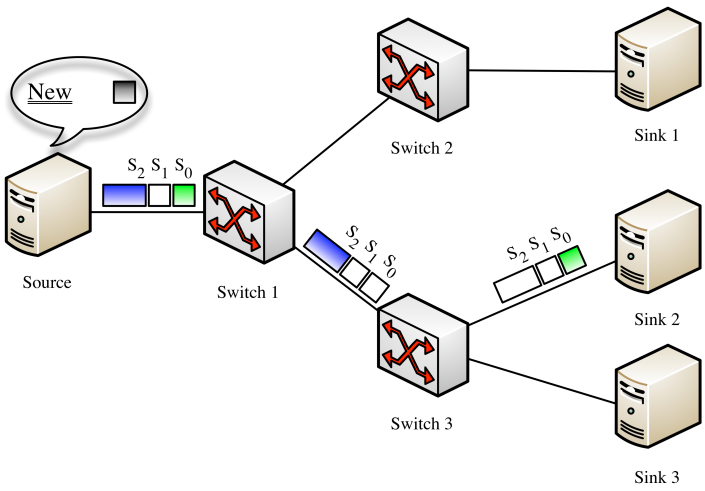


Figure 29: Timeslot continuity constraint in multi-rate TDM networks - State I

case, the controller will reject the request.

State III : **The network controller found a vacant timeslot on every hop of the network with a different index and identical data-rate**, Figure 31. In this case, the controller will reject the request.

State IV : **The network controller found a vacant timeslot on every hop of the network with a different index and different data-rate**, Figure 32. In this case, the controller will reject the request.

State V : **The network controller could not find a vacant timeslot on at least one hop of the network, regardless of the data-rate requested**, Figure 33. In this case, the controller will reject the request.

Introducing TSI to multi-rate TDM networks is more complicated than single-rate networks. So, what would happen when a new connection request is sent from the source to the network controller in the absence of timeslot continuity constraint, and the presence of TSIs on every space switching node? The control response will vary based on one of the following network existing states:

State I : **The network controller found a vacant timeslot on every hop of the**

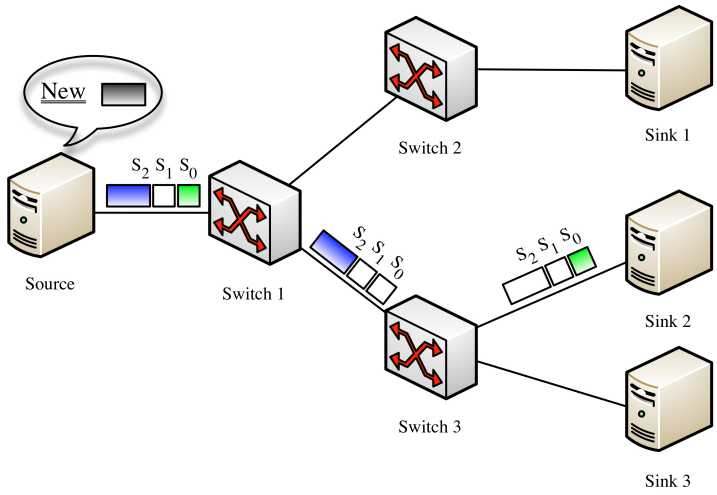


Figure 30: Timeslot continuity constraint in multi-rate TDM networks - State II

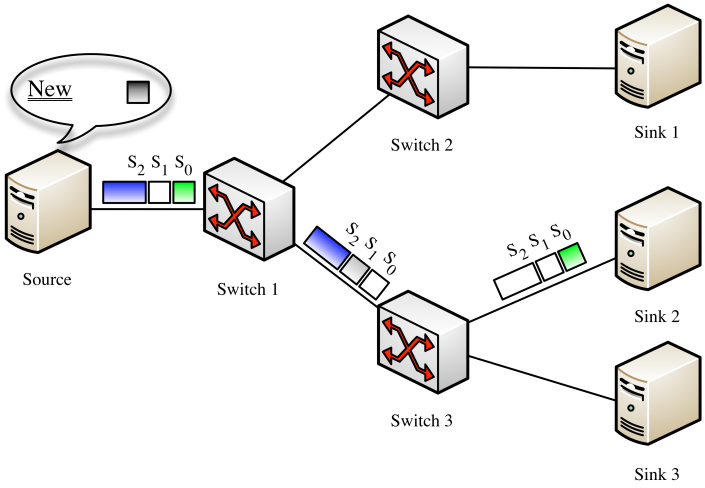


Figure 31: Timeslot continuity constraint in multi-rate TDM networks - State III

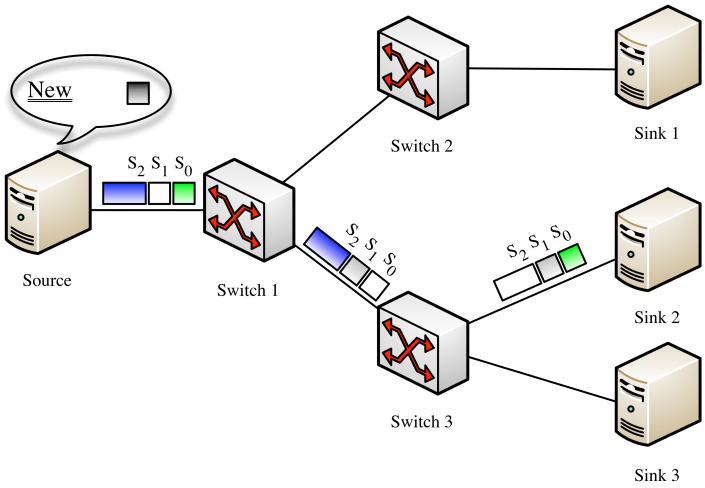


Figure 32: Timeslot continuity constraint in multi-rate TDM networks - State IV

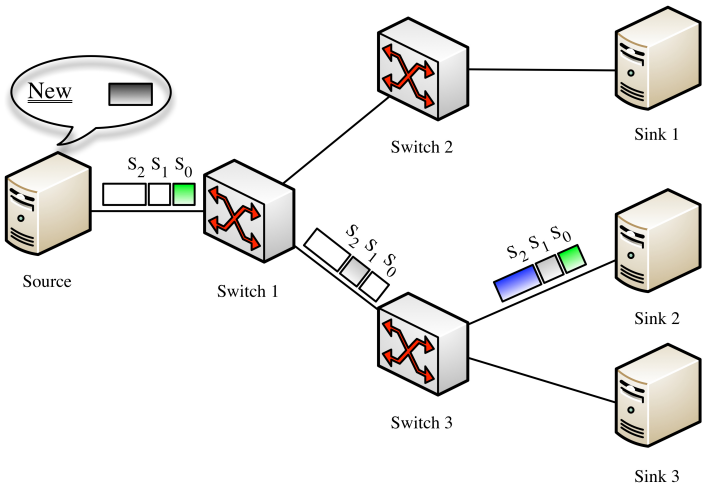


Figure 33: Timeslot continuity constraint in multi-rate TDM networks - State V

network with an identical index and identical data-rate, Figure 29. In this case, the controller will reserve the vacant timeslots for the new connection.

State II : **The network controller found a vacant timeslot on every hop of the network with an identical index and different data-rate**, Figure 30. In this case, the controller will reject the request.

State III : **The network controller found a vacant timeslot on every hop of the network with a different index and identical data-rate**, Figure 31. In this case, the controller will reserve the vacant timeslots for the new connection.

State IV : **The network controller found a vacant timeslot on every hop of the network with a different index and different data-rate**, Figure 32. In this case, the controller will reject the request.

State V : **The network controller could not find a vacant timeslot on at least one hop of the network regardless of the data-rate requested**, Figure 33. In this case, the controller will reject the request.

In summary, to accommodate a new connection in TDM multi-rate networks and the presence of timeslot continuity constraint, there has to be a vacant timeslot on every hop of the network with an identical timeslot index and identical data-rate (state I). On the other hand, to accommodate a new connection in TDM multi-rate networks in the absence of timeslot continuity constraint and the presence of TSIs , there has to be at least one vacant timeslot on every hop of the network with identical data-rate, regardless of the timeslot index (state I and state III). Table 6, compares the controllers' response to the source connection establishment request in the presence and absence of timeslot continuity constraint.

4.1.2 Continuity Constraint in wdm network.

In WDM networks, channels are represented by wavelengths. The effect of the presence and absence of continuity constraint in WDM is similar to TDM, but channels are represented in wavelengths. To establish a connection in Figure 34 between the source and sink2 (destination) in the presence of wavelength continuity constraint, only λ_3 is suitable to accommodate

Table 6: Comparison between the present and absence of timeslot continuity constraint in multi-rate TDM networks

	Response to connection request in the presence of continuity constraint	Response to connection request in the absence of continuity constraint and TSI
State I	Request is accepted	Request is accepted
State II	Request is rejected	Request is rejected
State III	Request is rejected	Request is accepted
State IV	Request is rejected	Request is rejected
State V	Request is rejected	Request is rejected

the new connection, because it is the only vacant λ_i that is available in every hop. On the other hand, connection establishment requests will be blocked in Figure 35 in the presence of wavelength continuity constraint due to lack of resources (presented in wavelength). In WDM, the absence of continuity constraint is practiced more than the presence of wavelength continuity constraint because it increases the network utilization and reduces the P_b. To establish a connection in the absence of wavelength continuity constraint, Wavelength Interchanger (WLI) must be added to the network in all or some of the nodes. A WLI is a hardware device that has the ability to interchange incoming connections from wavelength λ_i to a different λ_o. WLI are expensive devices, therefore, more network research papers are heading towards avoiding adding WLI to the network or adding limited number of them. In Figure 35, introducing WLI to every node allows a new connection to be established in the absence of wavelength continuity constraint. In fact, adding only one WLI to Switch 3 in Figure 35, allows one new connection to be added to the network, such that λ_1 in the first and second hop (between Source - Switch 1 - Switch 3) is switched to λ_0 or λ_2 in the last hops (Switch 3 - Sink 2). Moreover, adding two WLI to Switch 1 and Switch 3 allows two additional connections to the networks.

Note that the networks in Figure 28 and Figure 35 are less utilized compared to the networks in Figure 27 and Figure 34, respectively. It was proven that the presence of timeslot

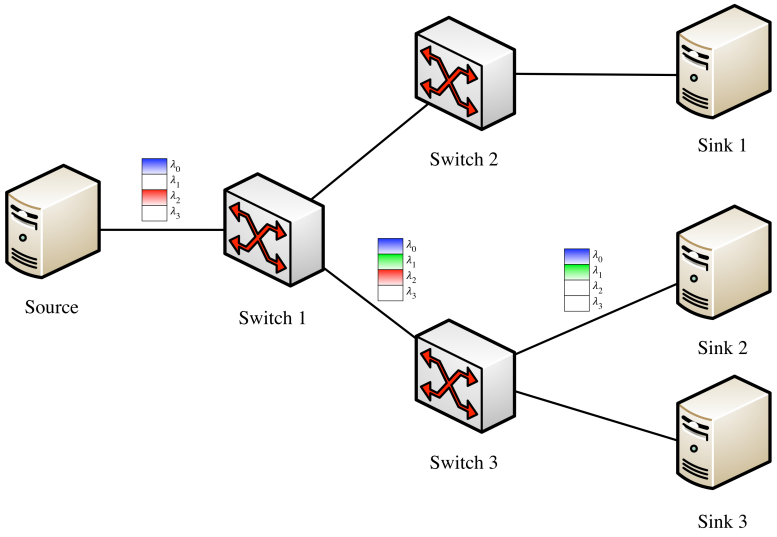


Figure 34: Wavelength continuity constraint example - 1

and wavelength continuity constraint dramatically increases P_b and reduces network utilization [52]. To efficiently utilize the network, and reduce the amount of blocked connection establishment requests, timeslot and wavelength interchangers are introduced to TDM and WDM networks, respectively. Introducing timeslot and wavelength interchangers to the network relaxes continuity constraints and improves P_b.

The presence or absence of continuity constraint for TDM and WDM networks is left to the network operator. It is a trade-off between cost, complexity, and utilization. The presence of continuity constraint, saves operators some money on extra hardware (including TSIs and WLIs) and reduces the complexity of channels' routing, assignment and switching. However, the presence of continuity constraint leaves the network under-utilized. Hence, since every channel on the network is worth some money, leaving underutilized mediums (unused channels) reduces the carriers' profit. Meanwhile, the absence of continuity constraint increases the carrier's profit at the expense of extra hardware cost (including TSIs and WLIs) and complex channels' routing, assignment and switching. Some research papers suggest that using limited interchangers results in identical results as full range [68] [43].

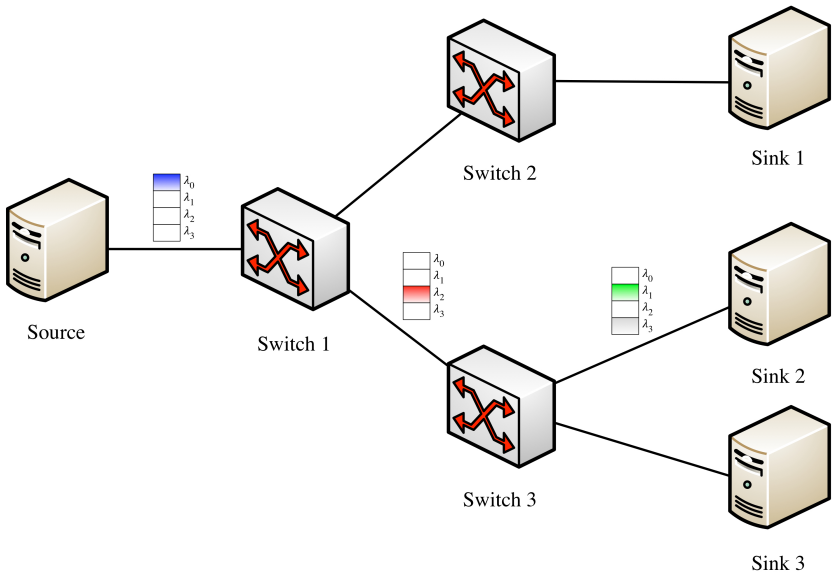


Figure 35: Wavelength continuity constraint example - 2

4.2 FRAME INTEGRITY IN TDM NETWORK

4.2.1 Switching In Time Domain with Frame Integrity.

A switching fabric is said to preserve frame integrity if timeslots inside a frame remain in the same frame after they are switched. In other words, timeslots are interchanged within the boundary of the frame. In practice, a switching with preserved frame integrity is reliable and more practical than the one without frame integrity [46].

As discussed earlier, the existing practice to substitute the absence of RAM is by delaying the light beam to a predetermined duration. Because we cannot go forward in time, in the presence of frame integrity, every incoming frame is entirely delayed by the duration of one frame T_f and switched simultaneously. Hence, the incoming frames f_0 and f_1 , arrive at TSI at time 0 and time T_f, respectively, and leave TSI at times f_1 and f_1+T_f, respectively. The amount of delay required to switch a timeslot from input index i to output index o in the

present of frame integrity is given by:

$$\delta_i^o = S + i - o \tag{4.1}$$

Assuming no timeslot interchanging is required for any timeslot in an incoming frame (swa_0 in Figure 19), then the entire frame must be delayed by T_f. Hence, using equation 4.1, every timeslot will be delayed by a duration of 4 timeslots. The longest delay occurs when the first timeslot S_0 is interchanged with the last timeslot in the frame S_{S-1} . Using the equation 4.1 for $S = 4$ results in $\delta_0^3 = 7$. Thus, for any given number of timeslots per frame S, the maximum delay (δ_{max}) required to interchange S_0 with S_{S-1} is given by equation 4.2. On the other hand, the shortest delay occurs when the last timeslot in the frame S_{S-1} is interchanged with the first timeslot S_0, where $\delta_{S-1}^0 = 1$, equation 4.1.

$$\delta_{max} = 2S - 1 \tag{4.2}$$

The following example demonstrates the interchanging process in the presence of frame integrity for a synchronized TDM system consisting of four timeslots per frame, Figure 18. Each timeslot has a fixed duration T_s that equals to the timeslot size (S_{size}) divided by data-rate data-rate (R) as it appears in the following equation:

$$T_s = S_{size}/R \tag{4.3}$$

Timeslots are logically grouped into frames, and a frame duration T_f is equal to the duration of all timeslots plus the duration of all guard times T_g prior to every timeslot, as it appears in the following equation:

$$T_f = S(T_s + T_g) \tag{4.4}$$

For simplicity, T_g will be ignored. Thus, the frame duration will be $T_f = S * T_s$. Note that δ_{max} is always less than $S * T_s$. Which means, the longest delay δ_{max} for interchanging S_0^{in} in frame f_0 is completed before the arrival of the first timeslot S_0^{in} in frame f_2. In other words, all timeslots from frame f_0 depart the TSI before the arrival of S_0^{in} in frame f_2. This

Table 7: Delay required to switch every timeslot for SWA a_{21}

S_i^{in}	S_o^{out}	δ
S_0^{in}	S_3^{out}	7
S_1^{in}	S_1^{out}	4
S_2^{in}	S_2^{out}	4
S_3^{in}	S_0^{out}	1

observation is important because the value of SWA increases dramatically as F increases. The observation concludes that, regardless of the value of F, equation 2.12 is going to be:

$$A = (S!)^2 \tag{4.5}$$

Resuming the example by assuming that $swa_{21} = (03)(1)(2)$, Figure 73, the amount of delay required to interchange each timeslot is presented in Table 7. Timeslot S_0^{in} must be delayed by $\delta = 7$ in order to be switched to S_3^{out}, Figure 36. Meanwhile, timeslot S_1^{in} and S_2^{in} must be delayed by $\delta = 4$ in order to be switched to S_1^{out} and S_2^{out}, Figure 37 and Figure 38, respectively. Lastly, timeslot S_3^{in} must be delayed by $\delta = 1$ in order to be switched to S_0^{out} , figure 39.

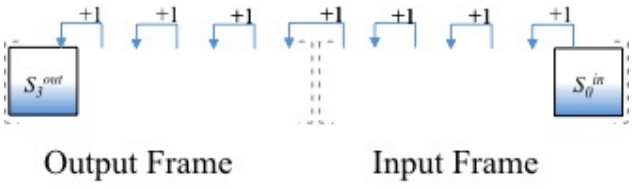


Figure 36: $\delta_0^3 = 7$ in the presence of frame integrity

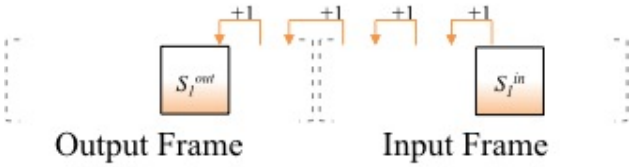


Figure 37: $\delta_1^1 = 4$ in the presence of frame integrity

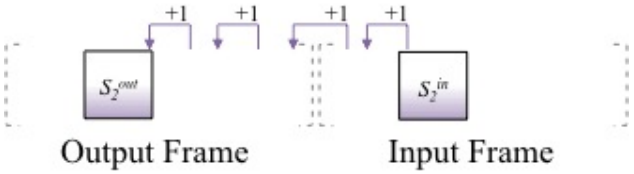


Figure 38: $\delta_2^2 = 4$ in the presence of frame integrity

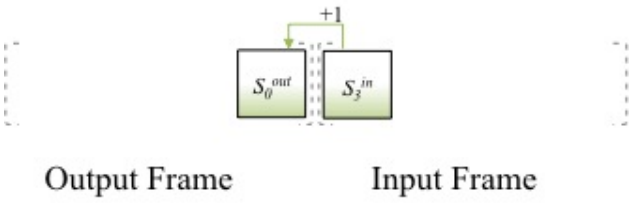


Figure 39: $\delta_3^0 = 1$ in the presence of frame integrity

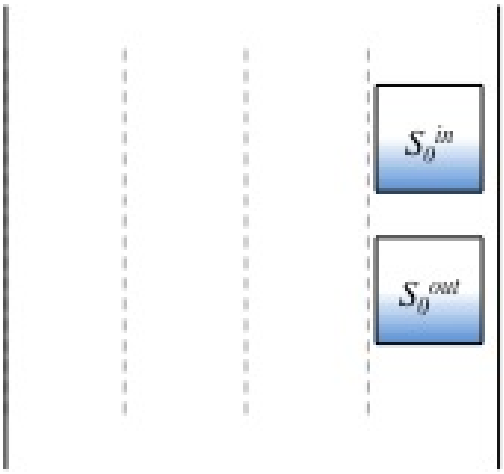


Figure 40: $\delta_0^0 = 0$ in the absence of frame integrity

4.2.2 Switching In Time Domain without Frame Integrity.

A switching fabric does not preserve frame integrity if timeslots inside a frame can be switched with any other timeslot at any frame. In other words, timeslots may cross frame boundaries after they are interchanged. A switching fabric without frame integrity is not preferred [3] [41]. Ignoring frame integrity results in many drawbacks including: missing timeslots, timeslots arrive out of order and underutilized medium. Each one of these disadvantages will be presented at the end of this subsection.

The process of interchanging timeslots, without frame integrity, is different than it is with frame integrity. Unlike the previous process where every frame is delayed by T_f, in this process a frame delay is not necessary. If a timeslot does not require switching, it is not delayed, it simply passes through TSI fabric. Continuing the example at the beginning of this subsection will help better explain the process. Assuming that $swa_5 = (0)(13)(2)$ is the switching assignment for an incoming frame, S_0^{in} arrives to the fabric at arrival time (t) $t = 0$. Based on the SWA, no interchange process is needed. Hence S_0^{in} will pass through the fabric with $\delta_0^0 = 0$, as it appears in Figure 40.

At $t = T_s$, S_1^{in} arrives at the fabric and, based on the switching assignment, it must be interchanged to S_3^{out}; , hence, $\delta_1^3 = 2$, as it appears in Figure 41.

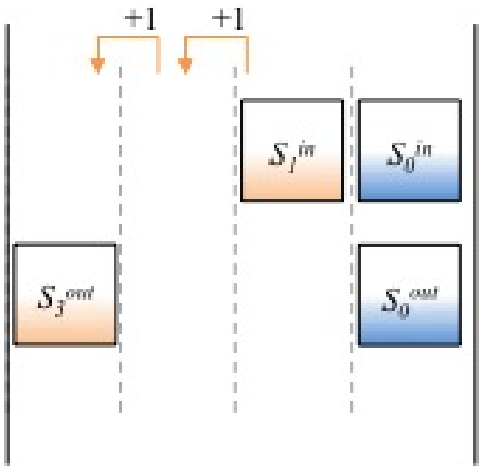


Figure 41: $\delta_1^3 = 2$ in the absence of frame integrity

At $t = 2(T_s)$, S_2^{in} arrives at the fabric and, based on the switching assignment, S_2^{in} does not require switching. Therefore, it will pass through the fabric with $\delta_2^2 = 0$, as it appears in Figure 42. During this time, S_1^{in} has already completed a delay equal to T_s at an FDL.

At $t = 3(T_s)$, S_3^{in} arrives at the fabric and based on the switching assignment, S_3^{in} must be interchanged to S_1^{out}. Note that since we cannot go forward in time, S_3^{in} cannot be interchanged with S_1^{out} in the same frame. Therefore, it must be delayed until it is the slot of S_1^{out} in th next frame. Thus, $\delta_3^1 = 2$ as it appears in Figure 43. In addition, at $t = 3(T_s)$, S_1^{in} will complete the required delay to be placed at S_3^{out}.

In Figure 43, timeslots at the output frame are out of order, as the new order in the output frame is $\{S_0^{out}, S_2^{out}, S_3^{out}, S_1^{out}\}$. In addition, the out of frame timeslot S_1^{out} will be discarded (missed) because it is outside the frame boundary. Missing one timeslot per frame is not a big of a deal when S is small (e.g. $S = 4$), however, as the number of timeslots per frame increases, the missing timeslots will be increased. Lastly, in Figure 43 there is a vacant timeslot that occurs at the output frame after S_0^{out} due to the process of interchanging without frame integrity. These are the three main disadvantages in interchanging timeslots without frame integrity.

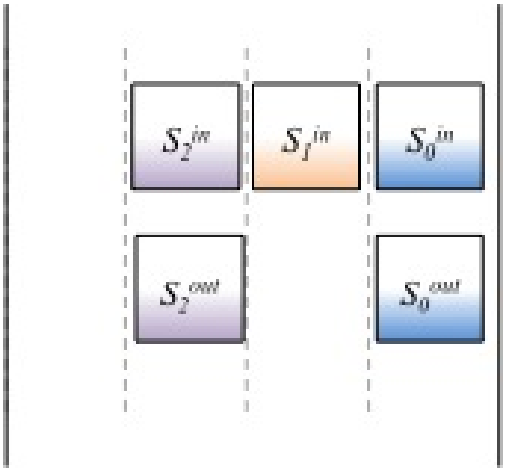


Figure 42: $\delta_2^2 = 0$ in the absence of frame integrity

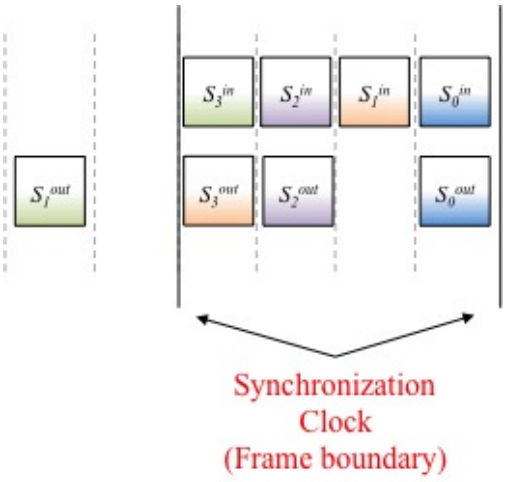


Figure 43: $\delta_3^1 = 2$ in the absence of frame integrity

5.0 TIMESLOT INTERCHANGER CONTROL ALGORITHM

This chapter presents a control algorithm for the previously-proposed all-optical TSI. Path assignment algorithms are discussed in the next two chapters. This chapter assumes that the path is selected based on either DPAA or EPAA.A complete TSI consists of a parallel set of basic time switching components known as Delay Elements (DEs). The initial proposed delay element [69] is found in Figure 22. A schematic representation for a single DE for S= 4 is presented in Figure 44. In addition to DEs, a complete TSI has a tree of splitters, a tree of combiners and a controller. A complete TSI for S= 4 is shown in Figure 45. This chapter is organized as follows: the assumptions used in modeling and running this simulation are stated in the first section. The next section presents one of the contributions in this dissertation that leads to the development of the control algorithm.

5.1 ASSUMPTIONS

For every simulation run, the following assumption will be considered regardless of the timeslot assignment algorithm:

- The source generates two consecutive frames per simulation run ($F = 2$). Because the maximum delay for any timeslot is always less than the duration of two consecutive frames (Equation 4.2), the number of frames in the fabric at any moment is always less than or equal to 2. Therefore, there isn't any obligation to generate more than two frames.
- Each frame generated by the source consists of $S = 2$, 4 or 8 timeslots.

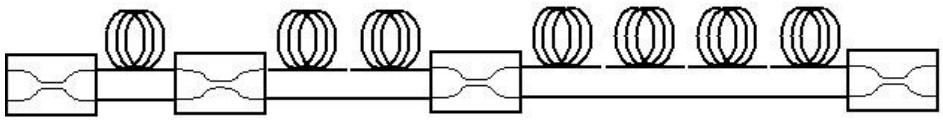


Figure 44: Delay Element for TS=4

- The size of each timeslot generated by the source is fixed, and $S_{size} = 10^6$ bits. The size of a timeslot does not affect the simulation process, thus, it is arbitrarily selected.
- Timeslots are transmitted at data-rate $R = 10^9$ bps. This number is found on most optical literature.
- The duration of a frame is given by Equation 4.4.
- There exists a guard-time period prior to every timeslot (including the first timeslot S_0^{in}). All guard-times are equal. The duration is set to be 10% of the duration of the timeslot.
- This study will run all possible permutations. In every simulation run, a new swa_i is assigned to one or both frames. More details about swa_i are discussed below.
- Frame integrity is considered for all frames.
- Each run starts with an empty fabric; then timeslots are generated at the deterministic rate of T_s.
- Every connection is unicast.
- Each simulation run is independent.

5.2 THE CONTROL ALGORITHM

The main purpose of this algorithm is to operate the interchanger's fabrics with $P_b = 0$. Operating the fabric includes the ability to change the switching state of any switch when needed. In addition, operating the interchanger includes keeping a log of the current state of the fabric and any scheduled operation in the future. It is really important to emphasize that this control algorithm is a hardware dependent. Thus, this control algorithm only work

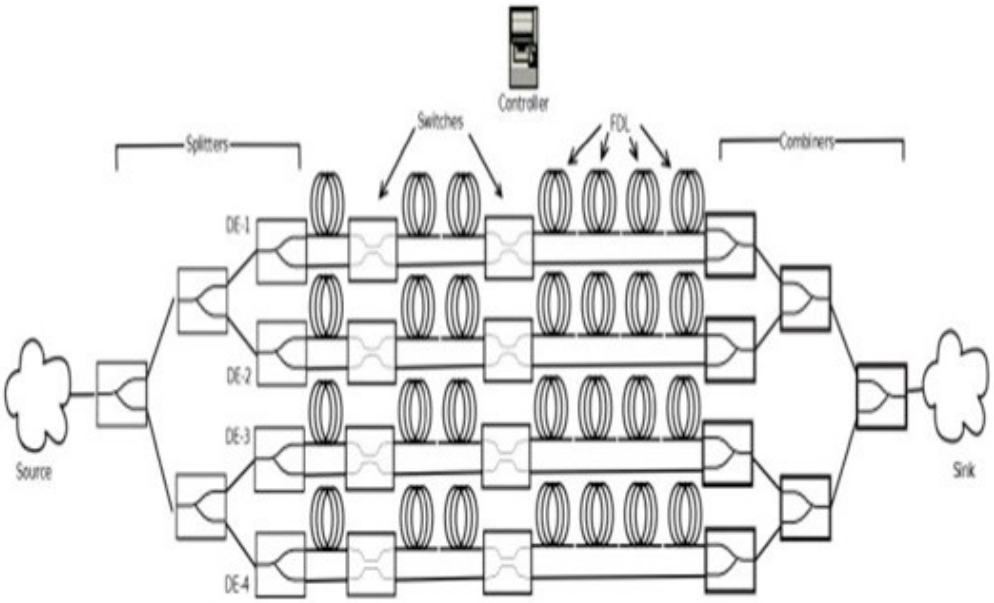


Figure 45: A complete TSI for TS = 4

for this switching fabric. If the hardware of the time switching fabric is modified, the control algorithm must be, also, modified to avoid undesired scenarios. The details of the algorithm are discussed in greater detail below. However, before discussing the control algorithm, my contribution to this work is stated first.

As of the day of this dissertation, there is not any such work that has been published for the exact timeslot interchanger's fabric. Therefore, the entire simulation is designed and implemented from scratch, and includes: hardware components and software.

5.2.1 Hardware components

Seven different hardware components are simulated in this dissertation:

1. **Source:** The source is a node responsible for generating timeslots at predefined intervals and is located at the ingress side of the fabric. The source can be a hardware device or it could be a fiber that feeds traffic from another network. In this dissertation, the source is a hardware device that generates timeslots at deterministic intervals.

2. **Sink:** The sink is a node that is responsible for receiving timeslots and keeping statistics for all arriving timeslots. The sink can be a hardware node (destination node), switching nodes, or a fiber that feeds into another network. In this dissertation, the sink is a hardware device that receives timeslots and computes some statistics. After statistics are collected and computed, timeslots are dumped. The sink is important to ensure that no timeslot is blocked or has gone missing.
3. **FDLs:** Since FF-FDLs are used in the modeled TSI, and physical layer parameters are not the major performance metric used in this study, FDLs are logically replaced with a FIFO queue with a variance service rate and queue length equal to zero. The service rates for each FDL is set to be equal to the amount of delay that a fiber loop would provide. For instance, every FDL in Figure 44 is replaced with a FIFO queue with a service rate equal to T_s. Hence, delay stages 1, 2, and 3 are replaced with 1, 2, and 4 FIFO queues, respectively.

   An FDL with a service rate greater than T_s can be replaced by either a single FIFO queue with service rate $D * T_s$ or replaced by D FIFOs queues chained together, each with a service rate of T_s. In the modeled simulation, the second configuration (chained FIFO queues) is chosen. This practice is closest to reality because it allows multiple timeslots to share the FDL stage when the queue length is set to zero.
4. **Switches:** Switches (1x2, 2x2, and 2x1) are built to serve the purpose of this study. The term "Switching Module (SWM)" is used to generalize the three different types of switches. There are two types of ports in every SWM: data ports and control ports. Data ports are input and output ports that handle timeslots, while control ports handle boolean control singles. Incoming timeslots are forwarded to the desired output port based on the switching states. Switching signals arrive during guard time, prior to every timeslot. The controller sends 0 for CROSS and 1 for BAR.
5. **Controller:** The controller is the brain of this switching fabric. In reality, the controller runs at the CPU clock speed. However, in this model, this issue is not considered. The controller holds a database for every component in the fabric. In addition, it holds a log file for the activities inside the switch. Almost all of the following discussion is about the proposed algorithms that take place at the controller.

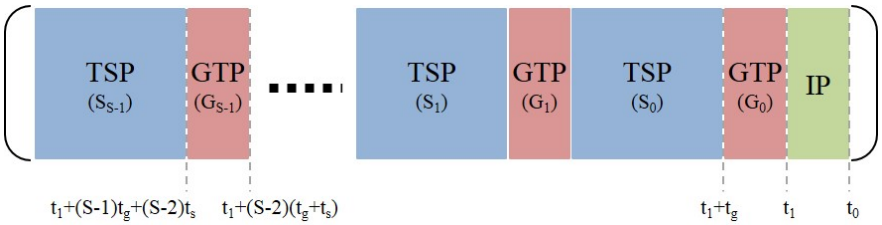


Figure 46: TSI's Control Algorithm Phases

5.2.2 Software

Every line of code written for the purpose of this dissertation falls under software contributions. Mainly, the algorithms used to control the fabrics are the most valuable piece of software. Note that, since hardware components are simulated not physically built, they are also considered software contributions.

The control algorithm built to operate the previously proposed TSI in Figure 45 is divided into three phases, as shown in Figure 46: Initialization Phase (IP), Guard Time Phase (GTP), and Timeslot Phase (TSP).

1. **Initialization Phase (IP)**: This phase starts at time $t = 0$ and occurs once for every simulation run. In this phase of the simulation, the controller starts by broadcasting the simulation parameters' packet to every module[1] in the fabric with different details depending on the type of module. The source and sink receive the following parameters: number of frames per simulation run, number of timeslots per frame, timeslot size, data-rate, timeslot duration, guard-time duration, and switching assignment. Switches receive guard-time duration, which is used to set the switching speed. Lastly, FDLs receive timeslot duration to set service time equal to the duration of a timeslot. This step is used for simulation purposes because the code is built to satisfy any number of timeslots per frame, such that $S = 2^2, 2_3, 2^3, ...2^k$. In addition, some simulation parameters such as "Module ID" changes in every run. In reality, the controller is aware of every detail in

[1]The term "Module" in this context refers to the fabric's hardware components in the simulator, regardless of its type. There are four module types: source, sink, switch, and FDL.

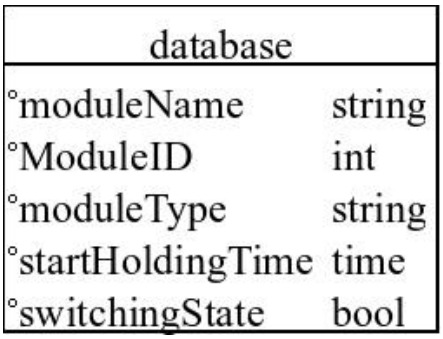


Figure 47: Switches database attributes

the the fabric while switches are simple devices that do not have processing or storage capabilities.

Once hardware components receive the parameters' packet from the controller, they save the parameters in their own memory and reply back with its module information, which includes: module name, module ID, and module Type. The controller starts building a database for every switch in the TSI as it appears in Figure 47. Note that *startHoldingTime* and *switchingState* will remain empty and will be filled after the path is selected (discussed later).

Then, the controller computes the maximum delay required to switch timeslots from the delay class using Equation 4.2. The last step in this phase is setting timeslot and frame counters to zero to maintain synchronization between the source, sink and controller, and that conclude Initialization Phase. Figure 48 summarizes the above steps in a sequence diagram.

2. **Guard Time Phase (GTP)**: This is a repetitive phase that starts at time $t = 1$ and repeats itself after every t_s for S times per frame, as it appears in Figure 46. In every simulation run, there are $S * F$ guard-time periods, one prior to each timeslot generated by a source. During the guard-time period, the controller does a sequence of operations that sets up a path for the incoming timeslot. These sequences of operations result in changing the switching state for all switches in the path from source to sink. The sequence of operations that leads to the optimum path in the fabric is demonstrated in

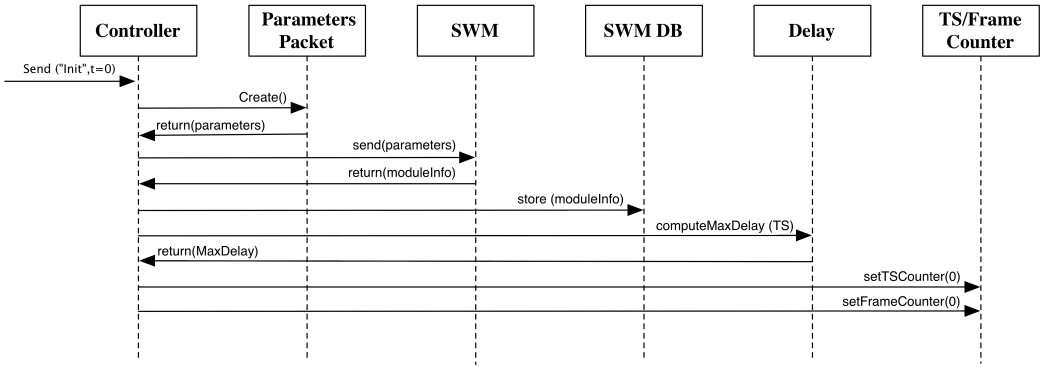


Figure 48: Initialization Phase Sequence Diagram

a sequence diagram in Figure 52, and is discussed below.

GTP starts at the controller by checking if the incoming timeslot is within the range of the simulation $0 \leq S_i^{in} \leq (S-1) * F$. If it is TRUE, the controller will proceed to deal with a new incoming timeslot (new timeslot direction). If the condition is FALSE, the simulation will deal with existing timeslots in the fabric until all timeslots are departed (existing timeslot direction). This step does not happen in a real system (precisely), because the trail of timeslots never ends. Both directions are presented in Figure 53 and stated below. However, before proceeding in detail about the sequence of operations that take place during GTP, three matrices must be presented and explained in order to fully understand GTP. These matrices are: Switching Assignment (SWA) matrix, delay and cumulative delay matrices, and switching control (SWC) matrices.

Switching Assignment (SWA): A SWA matrix is built to include all switching permutations for all input timeslot index i where $i = \{0,...,S-1\}$ (S_i^{in}) to all output timeslot index o where $o = \{0,...,S-1\}$ (S_o^{out}). Each input timeslot S_i^{in} may be switched into S possible output timeslots S_o^{out}. For any frame, there are $S!$ possible SWAs. A complete SWA matrix is [S!][S] two dimension matrix. The matrix rows are indexed from 0 to $S! - 1$, which is the switching assignment index (swa_i), and the matrix columns are indexed from 0 to $S-1$, which represent S_i^{in}. The value

of each cell is S_o^{out}. A sample from a complete SWA matrix for $S = 4$ is shown in Figure 19 and the complete SWA matrix is shown in Figure 73 in Appendix A. For instance, the intersection cell's value for row 1 (swa_1) and column 4 S_3 is equal to 2. It is read as follows: in a switching assignment swa_1, S_3^{in} is assigned to be switched to S_2^{out}.

The value of swa_i is assigned to the simulation during the initialization phase every simulation run. Since $F > 1$, the total number of permutations follows two cases:

> **SSWA**. In this case, the switching assignment is constant for every frame. In other words, both frames in every simulation run have the exact swa_i. Therefore, $SWA = S!$.
>
> **DSWA.** In this case, the switching assignments is allowed to be changed at the frame boundary. In other words, each frame in every simulation has an independent swa_i. Therefore, $SWA = (S!)^F$. DSWA is more realistic than SSWA because in reality frames and timeslots are independent from each other.

This dissertation is going to apply every possible SWA for $S = 2, 4, 8$ for SSWA, but only $S = 2, 4$ to test for DSWA. For $S = 8$, the total number of SWA exceeds 1.6 billion assignments.

Delay and Cumulative Delay Matrices: The delay matrix is built to improve TSI utilization by sharing DEs among multiple timeslots. Instead of reserving the entire DE from the time of generating a timeslot by source to the time of delivering the timeslot to sink, each switch in a path is reserved during the guard-time period prior to the arrival of the timeslot, and released (end of reservation) once a timeslot exits the switch (at the end of the guard-time period). The delay matrix is a [k] [δ_{max}], where k is the number of switches in path and δ_{max} is maximum delay. Figure 49 demonstrates a complete delay matrix for S = 4. The value of the delay matrix corresponds to the amount of delay introduced to the timeslot after each FDL stage at a given value of δ. For instance, assuming that $S = 4$ and the required delay to interchange S_2^{in} to S_0^{out} is $\delta = 2$ using Equation 4.1, then the third row

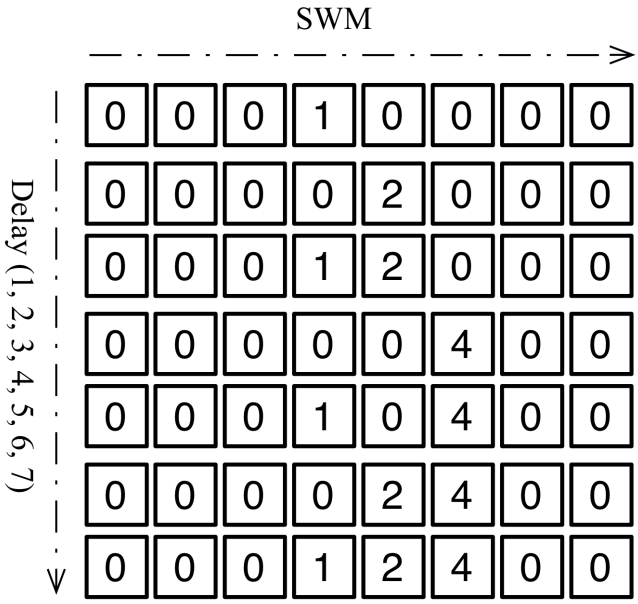


Figure 49: A delay matrix for $S = 4$

in Figure 49 corresponds $\delta = 2$ [0,0,0,0,2,0,0,0]. Given the current simulation time t_{sim}, S_2^{in} will reach the first four switches at time $t = t_{sim} + 0 * t_s$, the (0) value in the equation is taken from the matrix. At $t = t_{sim} + 2 * t_s$, S_2^{in} will arrive at the fifth switch after being delayed by $\delta = 2$, and no further delay S_2^{in} will experience. This concludes that S_2^{in} will be delayed at a single FDL stage.

However, the delay matrix is not helpful the way it is. This is because when a timeslot is delayed at any FDL, this timeslot will reach any switch with a delayed arrival time. For instance, in the above example, S_2^{in} will pass through the fabric without delay for the first four switches in Figure 45. Then after S_2^{in} is delayed by $\delta = 2$ at the second FDL stage, S_2^{in} will arrive at every switch after that after having been delayed by $\delta = 2$ at $t = t_{sim} + 2 * t_s$. Therefore, a cumulative delay matrix is introduced. In a cumulative delay matrix, the value of every column is added to the previous columns as it appears Figure 50 for $S = 4$.

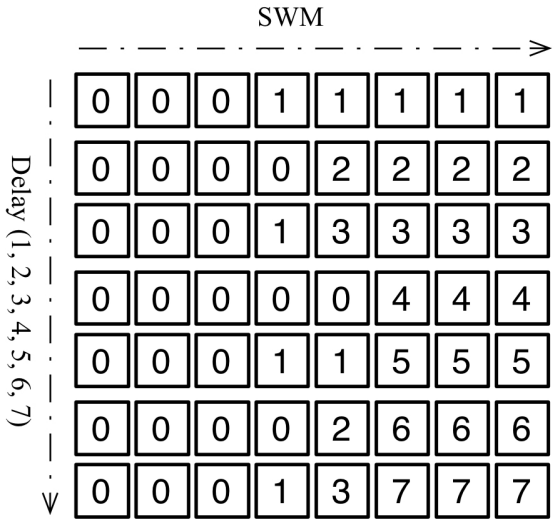


Figure 50: Cumulative Delay Matrix for $S = 4$

The difference between the delay matrix and cumulative delay matrix is better illustrated by a numeric example. Continuing the above example, the row that corresponds to $\delta = 2$ in the cumulative delay matrix is the third row in Figure 50, [0,0,0,0,2,2,2,2]. This is read as S_2^{in} will arrive at the first four switches in a path at time $t = t_{sim} + 0 * t_s = t_{sim}$ (without delay). Meanwhile, S_2^{in} will arrive at the last four switches in the path at time $t = t_{sim} + 2 * t_s$ after having been delayed by $\delta = 2$ on the second FDL stage. The benefit of the cumulative delay matrix will be clear when the path assignment algorithms are presented in Chapters 6 and 7.

Switching control (SWC) Matrices: SWC matrices carry boolean values that represent the switch state for all switches in a path. The value (1) represents BAR state and (0) represents CROSS state. Switching controls are kept as simple as Boolean values to reduce the difference in speed between the electronic slow processing speed and the fast optical transmission speed [70]. DPAA and EPAA algorithms are dependent on the cumulative delay matrix and SWC matrices.

SWC matrix is a three dimensional matrix with the size of $[S][k][\ \delta_{max}\ ]$. For any

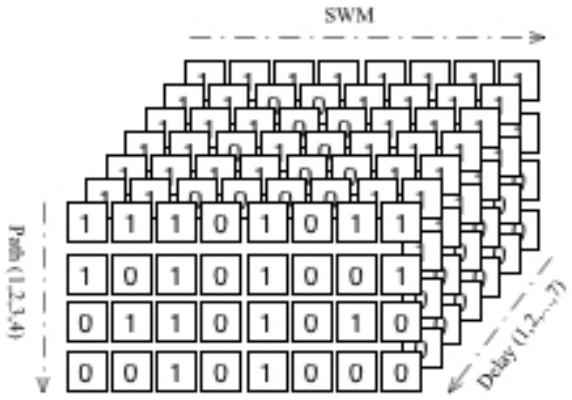


Figure 51: Sample of SWC matrices for S = 4

given delay δ, any timeslot has S alternative paths at which to be interchanged. There are δ_{max} matrices, each row in the matrix represents an alternative path and each column represents a switch in the path. Figure 51 demonstrates a sample of the SWC matrix for $S = 4$; the complete switching matrices are available in Appendix C .

Returning back to GTP, as mentioned earlier, there are two directions that the controller would take to decide the state of the current guard-time period:

- New timeslot direction: When the controller decides that the current guard-time period is a guard-time period prior to a new generated timeslot($guardtimeCounter < S-1$), the controller recalls swa_i and sends the following message to *outputTimeslot* method:

$$S_o^{out} \leftarrow \text{outputTimeslot } (S_i^{in}, swa_i)$$

outputTimeslot method returns S_o^{out} value from SWA matrix. Once S_o^{out} is determined, the controller computes the required δ to perform the switching operation by sending the following message to *computeDelay* method:

$\delta_i^o \leftarrow \text{computeDelay}\ (S_i^{in},\ S_o^{out},\ S)$

computeDelay method returns the value of δ required to switch S_i^{in} to S_o^{out} for S per frame using Equation 4.1. After that, the controller sends the following message to *selectPath* method:

$DE_i \leftarrow \text{selectPath}\ (\delta_i^o,\ S)$

selectPath method returns the delay element index i (DE_i) (every DE represents a different path in the fabric) to the controller. There are two different algorithms for selecting a desired path in the switching fabric: DPAA and EPAA. Both algorithms are presented greater detail in Chapters 6 and 7, respectively. Once the path is selected, the controller extracts the corresponding row of SWC from the SWC matrices using the following message:

for $int\ \ i = 0;\ \ i < k;\ \ i++$
$SWC[i] \leftarrow \text{getSWC}\ \ (\delta, DE_i, i)$

The extracted SWC matrix is a $1 * k$ matrix and extracted as follows: the value of δ determines the matrix that corresponds to the delay required, and the value of DE_i determines the row index that corresponds to the SWC configuration; lastly, i is the switch index in the path and it is the column index in the SWC matrix.
Once the SWC configuration is extracted, the controller starts computing the start holding time (*startHoldingTime*) for every switch, which is the time where S_i^{in} arrives at switch index k. The value of *startHoldingTime* determines whether switch k at a given time is idle or busy. The value of *startHoldingTime* is given by:

$$startHoldingTime = t_{current} + \delta * t_s \tag{5.1}$$

And the controller will run this equation for every SWM in path.

for $int\ \ i = 0;\ \ i < k;\ \ i++$

$startHoldingTime \leftarrow\ t_{sim} + getCumulativeDelay(\delta, i) * t_s$

The cumulative delay is extracted from the matrix using the following message:

$getCumulativeDelay \leftarrow\ commulativeDelay[\delta][i]$

At this point of time, the incoming timeslot has its SWC configuration extracted (*switchingState*) and *startHoldingTime* computed for every switch in a path. The controller adds these value to every switch attribute in order, based on the *startHoldingTime* to the database, such that the earliest *startHoldingTime* is always first. The database that stores SWCs is a FIFO queue such that every SWM has its SWC queue. The following paragraph is an extracted sample from the simulation output for S_0^{in} in f_0 for swa_2.

```
Current Timeslot @ Controller is .  .  .  0
Current Frame @ Controller is .  .  .  0

output switching for S 0 is .  .  .  0
required delay for switching S 0 to 0 is .  .  .  4

Module osplitter0 SwitchingControl [1] Start Holding Time = 1
Module osplitter1 SwitchingControl [1] Start Holding Time = 1
Module osplitter11 SwitchingControl [0] Start Holding Time = 1
Module oswitch10 SwitchingControl [1] Start Holding Time = 1
Module oswitch11 SwitchingControl [0] Start Holding Time = 1
Module ocombiner11 SwitchingControl [1] Start Holding Time = 41
Module ocombiner1 SwitchingControl [1] Start Holding Time = 41
Module ocombiner0 SwitchingControl [1] Start Holding Time = 41
*****************************************
Available Path index is...0
*****************************************
```

```
Module ( osplitter0 )
Length = 1
==============================
Job 1------
Start Holding Time = 1
Switching State = 1
==============================

Module ( osplitter1 )
Length = 1
==============================
Job 1------
Start Holding Time = 1
Switching State = 1
==============================

Module ( osplitter11 )
Length = 1
==============================
Job 1------
Start Holding Time = 1
Switching State = 0
==============================

Module ( oswitch10 )
Length = 1
==============================
Job 1------
Start Holding Time = 1
```

```
Switching State = 1
==============================

Module ( oswitch11 )
Length = 1
==============================
Job 1------
Start Holding Time = 1
Switching State = 0
==============================

Module ( ocombiner11 )
Length = 1
==============================
Job 1------
Start Holding Time = 41
Switching State = 1
==============================

Module ( ocombiner1 )
Length = 1
==============================
Job 1------
Start Holding Time = 41
Switching State = 1
==============================

Module ( ocombiner0 )
Length = 1
==============================
```

```
Job 1------
Start Holding Time = 41
Switching State = 1
============================
```

As the simulation advances in time, the number of jobs per switch increases. The following few lines are an example of the database for a single switch with five jobs scheduled in time:

```
Module ( ocombiner0 )
Length = 5
============================
Job 1------
Start Holding Time = 71
Switching State = 1
Job 2------
Start Holding Time = 81
Switching State = 1
Job 3------
Start Holding Time = 91
Switching State = 1
Job 4------
Start Holding Time = 101
Switching State = 1
Job 5------
Start Holding Time = 111
Switching State = 1
============================
```

Because *startHoldingTime* is inserted in order at every guard-time period, the controller checks the first row in the database for every switch. If the first row is equal

to the current simulation time t_{sim}, the switching control is sent to the desired switch and deleted from the database.

- Existing timeslot direction:

  The second direction occurs during a guard-time period when all timeslots are generated but some timeslots are yet to depart the fabric. This direction exists because frame integrity is considered in this TSI and control algorithm. The controller checks all databases (for every switch), if there is not a SWC scheduled (queue is empty), the controller simply skips this switch and moves to another one. However, if there exists a SWC that is not sent to its SWM, the controller checks the *startHolding-Time* to see if it equals the current simulation time t_{sim} or not. If it is equal to t_{sim}, the controller sends it to the desired switch and deletes it from the database. The process repeats until all SWCs are sent to their switches. The simulation is terminated once all SWCs scheduled at the controller are sent to the desired SWM.

That concludes the control algorithm except *selectPath* algorithms. Figure 53 summarizes the sequence of operations that happens during guard-time period in a flow chart and the pseudocode for this algorithm is presented in Algorithm 2. Both timeslot assignment algorithms are presented in the next two chapters. They share the exact TSI control algorithm.

3. **Timeslot Phase (TSP)**: This is a repetitive phase that starts at time $t = 1 + t_g$ and repeats itself after every t_g for S times per frame as it appears in Figure 46. In every simulation run, there are $S * F$ TSP. During this period, the source generate timeslots and send them to the TSI. Timeslots travel inside TSI while switches in a path are either configured or scheduled to change its state in the future.

Algorithm 1 Summary of the events that take place during guard-time period

Guard-time events

```
if getNumJobs() > getJobCounter() then
    int inTimeslot outTimeslot pathIndex δ,SWA
    inTimeslot = getTimeslotCounter()
    if getFrameCounter()%2==0 then
        SWA = getSWAF0()
    else
        SWA = getSWAF1()
    end
    outTimeslot = getOutTimeslot(SWA, inTimeslot)
    delay = computeDelay(inTimeslot,outTimeslot)
    pathIndex= selectPath(δ)
    reservePath(pathIndex,δ)
    sendSwitchingCont()
    increaseTimeslotCounter()
    jobCounter++
else
    if allSWCQEmpty() == false then
        sendSwitchingCont()
        increaseTimeslotCounter()
    else
        finish()
    end
end
```

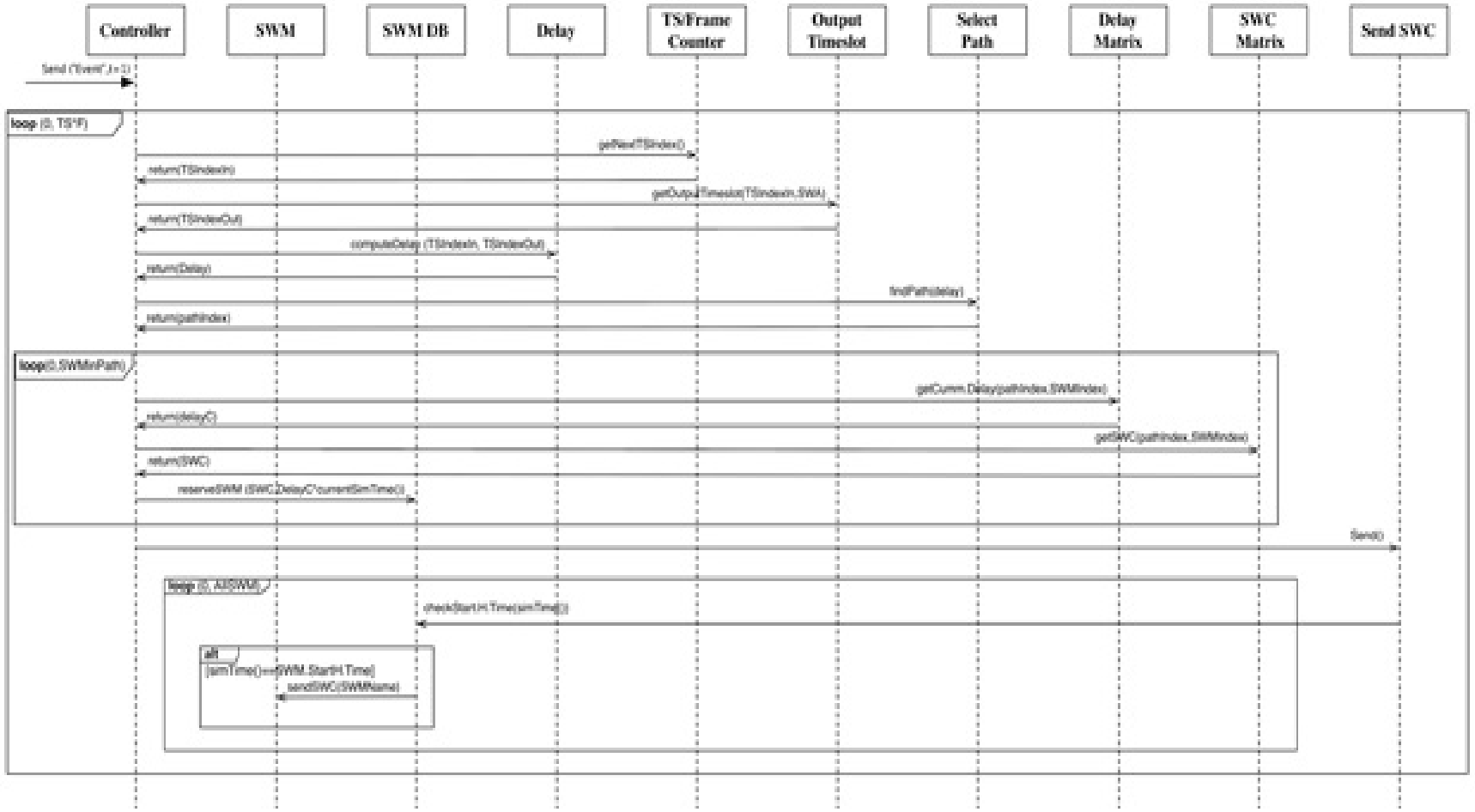


Figure 52: Guard Time Phase Sequence Diagram

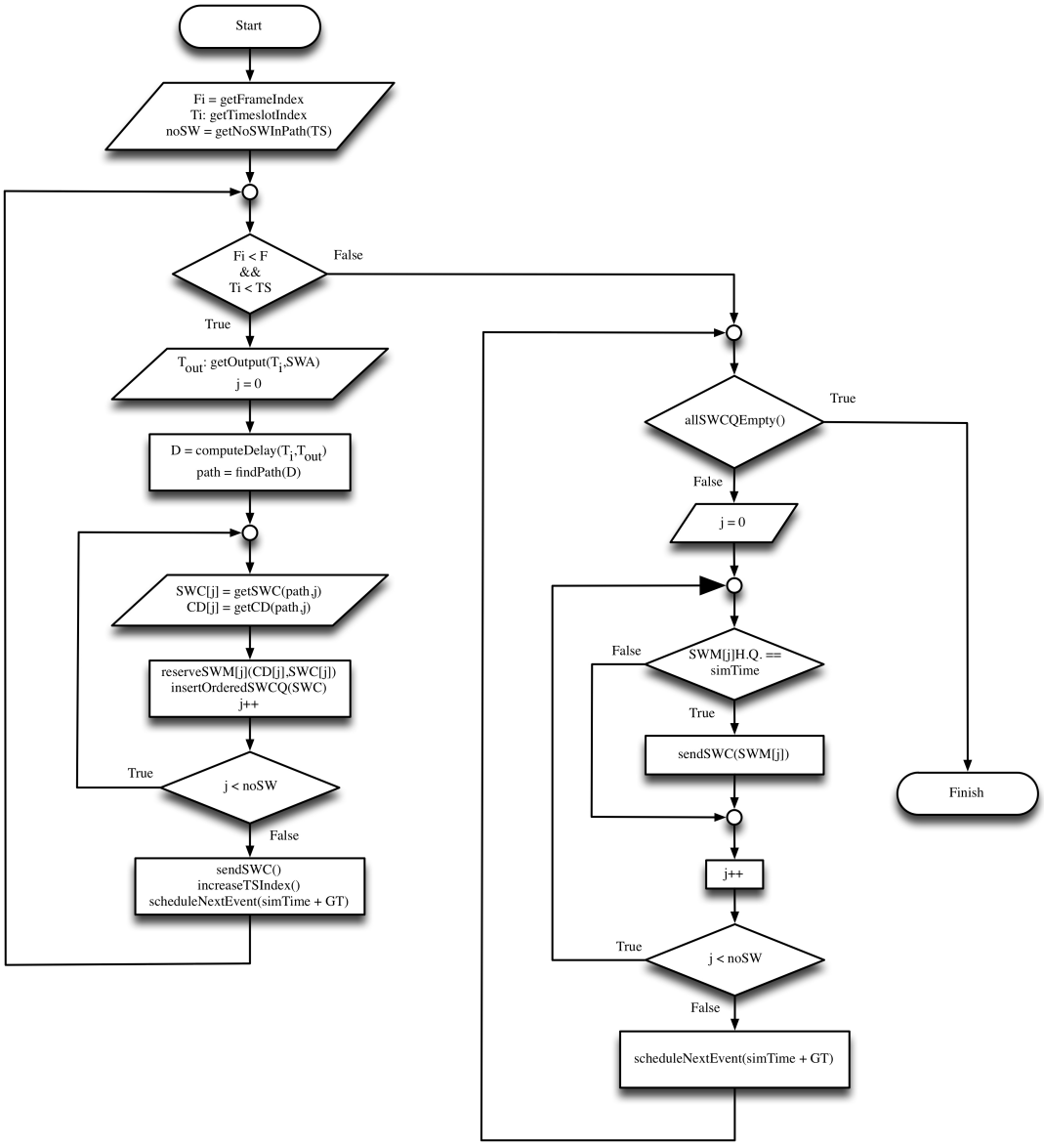


Figure 53: Summary of Guard Time Phase

6.0 DILATED PATH ASSIGNMENT ALGORITHM

The proposed TSI was meant to be simulated first. In addition, there is always a suspicion on the required number of parallel DEs when timeslots share DEs while maintaining $P_b = 0$. A Dilated Path Assignment Algorithm (DPAA) is introduced, (in addition to the fabric control algorithm), to operate the fabric at negligible crosstalk [1]. In switching, dilation is practiced when poor hardware components are used to build the switching fabric. Poor hardware components result in high crosstalk when multiple signals share the hardware component simultaneously. Hence, this assignment algorithm prevent two timeslots from passing though any switch simultaneously. The downside of dilation is the increase on the number of hardware components and manufacturing cost.

DPAA returns to the control algorithm (Chapter 5) an index number that corresponds to a path inside the TSI that an incoming timeslot will follow. The current TSI has S parallel DE, each of which is an alternative path. The algorithm starts searching in sequence from DE_0 to DE_{S-1} and selects first a DE that is "capable" of interchanging a timeslot without blocking. The term "capable", in this context, is defined as switching an incoming timeslot at a specific DE without blocking. Blocking happens when two timeslots arrive at the same switch simultaneously, even if both require the same switching state. Once blocking occurs, the algorithm will skip to the next DE. Since there are S alternative paths in the TSI, there exists a path for any incoming timeslot.

The rest of this chapter is organized as follows: DPAA is discussed in greater detail in Section 6.1. The simulation setup and results are discuses in Section 6.2. Lastly, a discussion on the results is presented in Section 6.3

[1] Dilation does not element crosstalk, however, it significantly reduce crosstalk.

6.1 ASSIGNMENT ALGORITHM

The simulator kernel assigns the assignment algorithm to the controller during the initialization phase of the simulation. Once S_o^{out} is determined, the controller sends the following method to the *selectPath* class:

$DE_i \leftarrow \text{selectPath } (\delta_i^o, S)$

This method returns the DE_i which S_i^{in} will follow. There are two subroutines that *selectPath* will call in order to proceed in the algorithm. These subroutines are *isEmpty* and *isBusy*. The two subroutines are discussed below.

bool *isEmpty(swm$_i$)*: This subroutine returns whether the SWC queue for swm_i is empty or not in boolean, Algorithm 6.1. SWC is stored in the FIFO queue. When the length of the queue is set to zero, the subroutine returns TRUE; otherwise, FALSE is returned.

Algorithm 2 isEmpty algorithm to verify that a switching control queue is empty or not

function isEmpty (swm_i) **Input** : A nonnegative integer swm_i

Output *isEmpty*

bool isEmpty

if $swm_i -> SWCQ.length() == 0$ **then**
| isEmpty = true

else
| isEmpty = false

end

return isEmpty

bool *isBusy(swm$_i$, startHoldingTime)*: This subroutine returns whether the swm_i is scheduled (busy) at *startHoldingTime* or not, Algorithm 6.1. Since jobs are stored in a FIFO queue, a queue iterator is used to traverse throughout the queue as follows:

The algorithm starts by initializing variables and declaring values as follows:

```
bool    pathSWQEmpty=true;
bool    blocking=false;
int     DE=0;
```

Algorithm 3 isBusy algorithm to verify that a switching module is previously reserved or idle

function isBusy ($swm_i, startHoldingTime$) **Input** : A nonnegative integer swm_i and simulation time variable $startHoldingTime$

Output $isBusy$

bool isBusy = false

queue::Iterator it = queue::Iterator (it->SWCQ, 0)

queue::Iterator itEnd = it.end()

SwitchingCont * SWC

do

SWC = (SwitchingCont *) it()

if *SWC.startHoldingTime==startHoldingTime* **then**

isBusy = true

else

it++

end

while *it!=itEnd || isBusy==false*

return isBusy

```
time      startHoldingTime=0;
```

After the initialization and declaration stage, the algorithm starts a loop that tests all S available paths, this loop is terminated if one of two conditions is satisfied: the path is found or $DE >= S$. When searching for a path in the TSI, there are three possible scenarios:

Scenario I: TSI is empty. This scenario is tested using the following code:

```
for (int i= 0; i< k; i++)
        pathSWQEmpty = pathSWQEmpty & isEmpty(swm_i i)
```

If *pathSWQEmpty* is TRUE, the algorithm finds a path, the algorithm is aborted and a DE index is returned. However, if *pathSWQEmpty* is FALSE, the algorithm proceeds to the next scenario.

Scenario II: TSI is not empty and no SWC is scheduled for any switch in the path (no blocking). This scenario is tested by checking every SWC for every switch in path DE_i, using the following code:

```
for (int i= 0; i< k; i++){
   startHoldingTime = simTime() + getCumulativeDelay(δ, i) * t_s
blocking = blocking | isBusy(swm_i, startHoldingTime);
}
```

If *blocking* is TRUE, the algorithm will proceed to the next scenario. However, if *blocking* is FALSE, the algorithm finds a path, the algorithm is aborted and a DE index is returned. This means timeslots are scheduled to use at least one switch but not at the same time.

Scenario III: TSI is not empty and there is a SWC scheduled for at least one switch in the path (blocking exist). In this scenario, the algorithm increases the path index DE_i by one, resets the boolean variables and starts from the beginning (after the initialization and declaration stages), as shown below.

```
pathSWQEmpty=true;
blocking=false;
```

```
DE++;
startHoldingTime=0;
```

Once the desired path is selected, the algorithm returns the selected DE_i to the control algorithm. Figure 54 presents DPAA in a flow chart and the pseudocode is presented in Algorithm 6.1.

6.2 SIMULATION SETUP AND RESULTS

DPAA is tested using different SWAs and a different number of timeslots per frame, as follows:

1. Simulating DPAA using SSWA when $S = 4$.
2. Simulating DPAA using DSWA when $S = 4$.
3. Simulating DPAA using SSWA when $S = 8$.

The simulation output and results is described below.

6.2.1 DPAA using SSWA when $S = 4$

The simulation setup is identical to Figure 45. A screen shot of the simulation setup for $S = 4$ is shown in Figure 55. The total number of simulation runs using SSWA is $SWA = 24$, Equation 2.12. In each simulation run, there were $S * F = 4 * 2 = 8$ timeslots generated by the source, giving a total of $S * F * SWA = 2 * 4 * 24 = 192$ timeslots generated in 24 runs. All of the generated timeslots were received by the Sink, as it appears in Table 8.

In Table 8, 100% of the generated traffic passed through Splitters 0, because it was the ingress switch to all DEs. In addition, 100% of the traffic generated by Source passed through Splitter 1, which means all traffic used DE_0 and/or DE_1. The remaining DEs can be eliminated without affecting the value of P_b of the TSI. In fact, when DE_2 and DE_3 are eliminated, Splitter 0 and Combiner 0 will also be eliminated because there will not be a

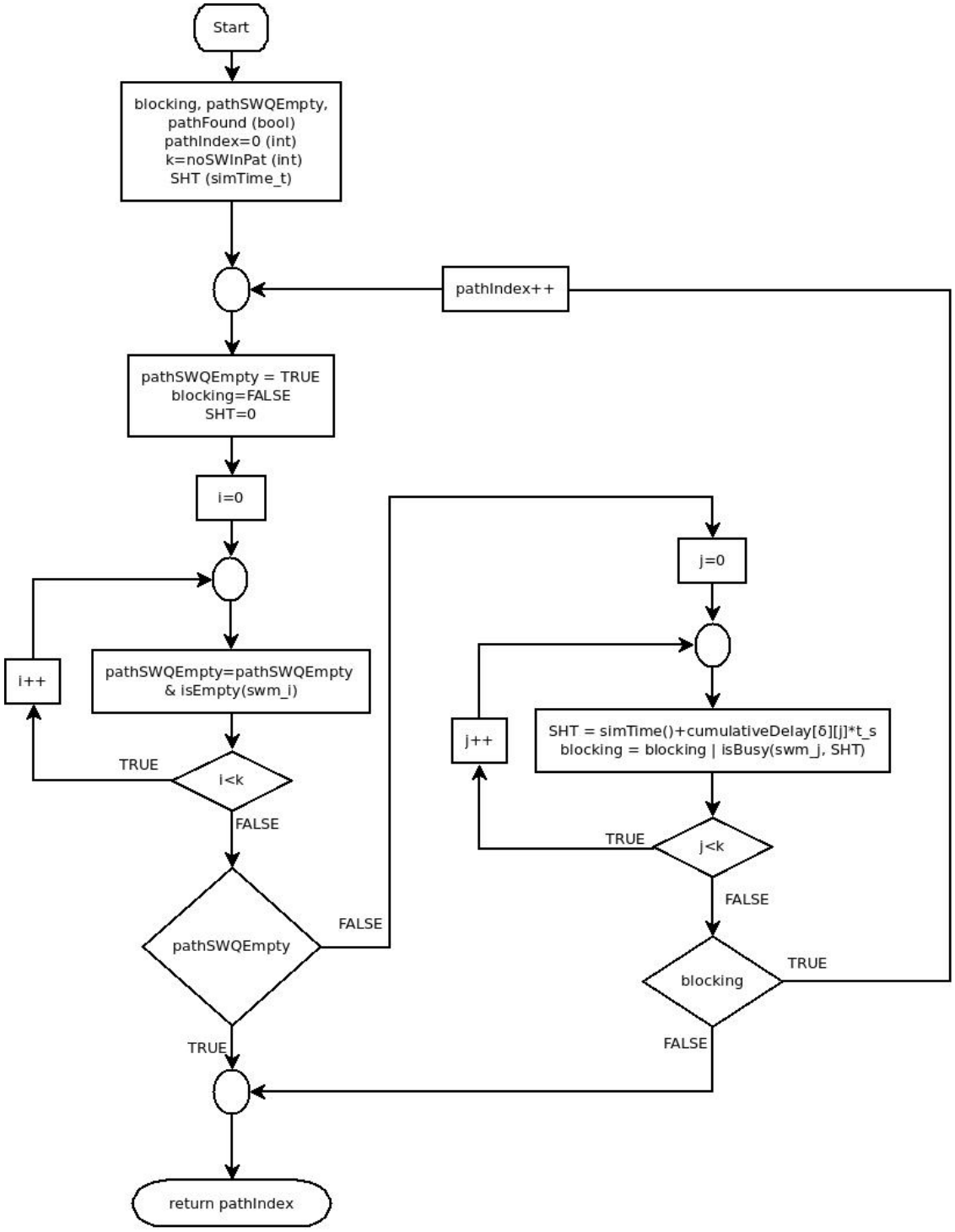


Figure 54: Flow Chart for DPAA

Algorithm 4 DPAA algorithm

selectPath (δ) **Input** : A nonnegative integer δ

Output $pathIndex$

.

bool blocking, pathSWQEmpty, pathFound=false

int pathIndex=0, k=getNoSwitchesInPath()

simtime_t SHT

do

  blocking = false pathQueueEmpty=true SHT=0

  for $i \leftarrow 0$ **to** $i < k$ **do**

    pathSWQEmpty= pathSWQEmpty & isEmpty(getModule(pathIndex,i))

  end

  if *pathSWQEmpty==true* **then**

    pathFound = true

  else

    for $j \leftarrow 0$ **to** $j < k$ **do**

      SHT = simTime()+cumulativeDelay[δ][j]*TimeslotDuration

      blocking= blocking | isBusy(getModule(pathIndex,j),SHT)

    end

    if *blocking==false* **then**

      pathFound = true

    else

      $pathIndex++$

    end

  end

while *pathIndex<S && pathFound==false*;

return pathIndex

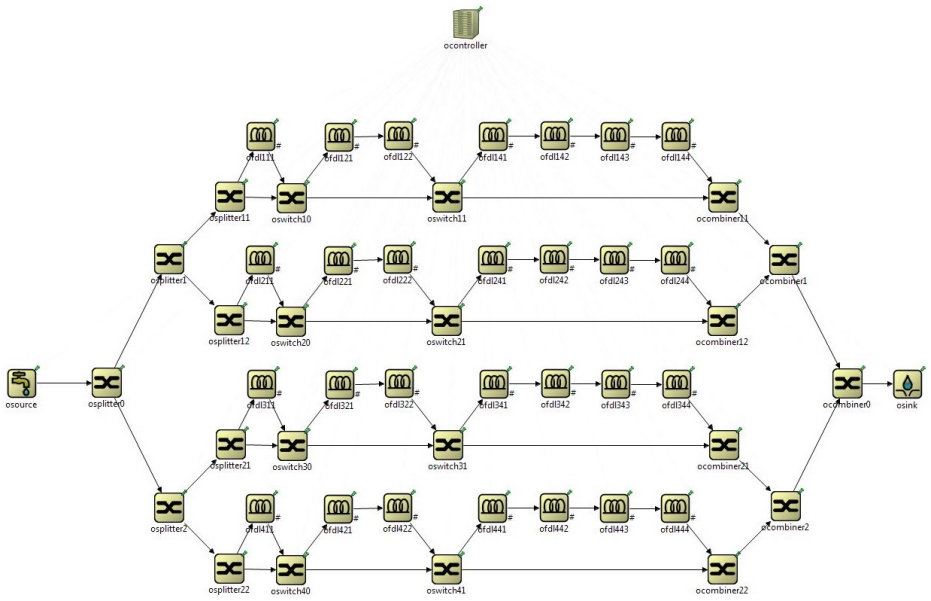


Figure 55: A screen shot of the simulation setup for $S = 4$

need to have them both. The total traffic is divided between DE_0 and DE_1 with 85.4% and 14.6%, respectively.

6.2.2 DPAA using DSWA when $S = 4$

This simulation setup is similar to the setup using SSWA when $S = 4$, with the difference in the switching assignment. In this setup, each frame has an independent SWA. The total number of simulation runs using DSWA is $24^2 = 576$, Equation 2.12. In each simulation

Table 8: Simulation summary of DPAA using SSWA when $S = 4$

						DE 0	**DE 1**	**DE 2**	**DE 3**
	Source (created)	Sink (revieved)	Splitter 0 (arrived)	Splitter 1 (arrived)	Splitter 2 (arrived)	Splitter 11 (arrived)	Splitter 12 (arrived)	Splitter 21 (arrived)	Splitter 22 (arrived)
# Timeslots	192	192	192	192	0	164	28	0	0
% Timeslots	100%	100%	100%	100%	0%	85.4%	14.6%	0%	0%

run, there were $S * F = 4 * 2 = 8$ timeslots generated by the source, giving us a total of $S * F * SWA = 2 * 4 * 576 = 4608$ timeslots generated in 576 runs. All of the generated timeslots are received by the Sink, as it appears in Table 9.

In Table 9, 100% of the generated traffic passed through Splitters 0. Unlike the previous setup, 99.9% of the traffic generated by Source passed through Splitter 1, while 0.1% of the total traffic passed through Splitter 2. The total traffic was divided between DE_0, DE_1 and DE_2 as it appears in Table 9. In this simulation, eliminating DE_3 does not affect the value of P_b. Figure 56 compares the precentage of traffic that passed through each DE using SSWA and DSWA.

6.2.3 DPAA using SSWA when $S = 8$

This simulation setup is identical to the setup using SSWA but $S = 8$ instead of $S = 4$, with the difference in the switching assignment and TSI itself, as shown in Figure 57. The total number of simulation runs using SSWA is $8! = 40,320$. In each simulation run, there were $8 * 2 = 16$ timeslots generated by the source, giving us a total of $2 * 8 * 40,320 = 645,120$ timeslots generated in 40320 runs. All of the generated timeslots are received by the Sink, as it appears in Table 10.

In Table 10, 100% of the generated traffic passed through Splitters 0. The total traffic is divided between 3 DEs (DE_0, DE_1, and DE_2). In this simulation, eliminating DE_3 though DE_7 did not affect the value of P_b, Figure 58.

6.3 DISCUSSION

This algorithm is suitable for a switching fabric with poor hardware components. If hardware components have high crosstalk, sharing switches between timeslots simultaneously will affect the received signal. Both SWA represent real case scenarios. SSWA is tested because in most case the variance between two consecutive frames is small. For instance, a termination of a single connection (timeslot) out of 256 connections can be neglected. On the other hand,

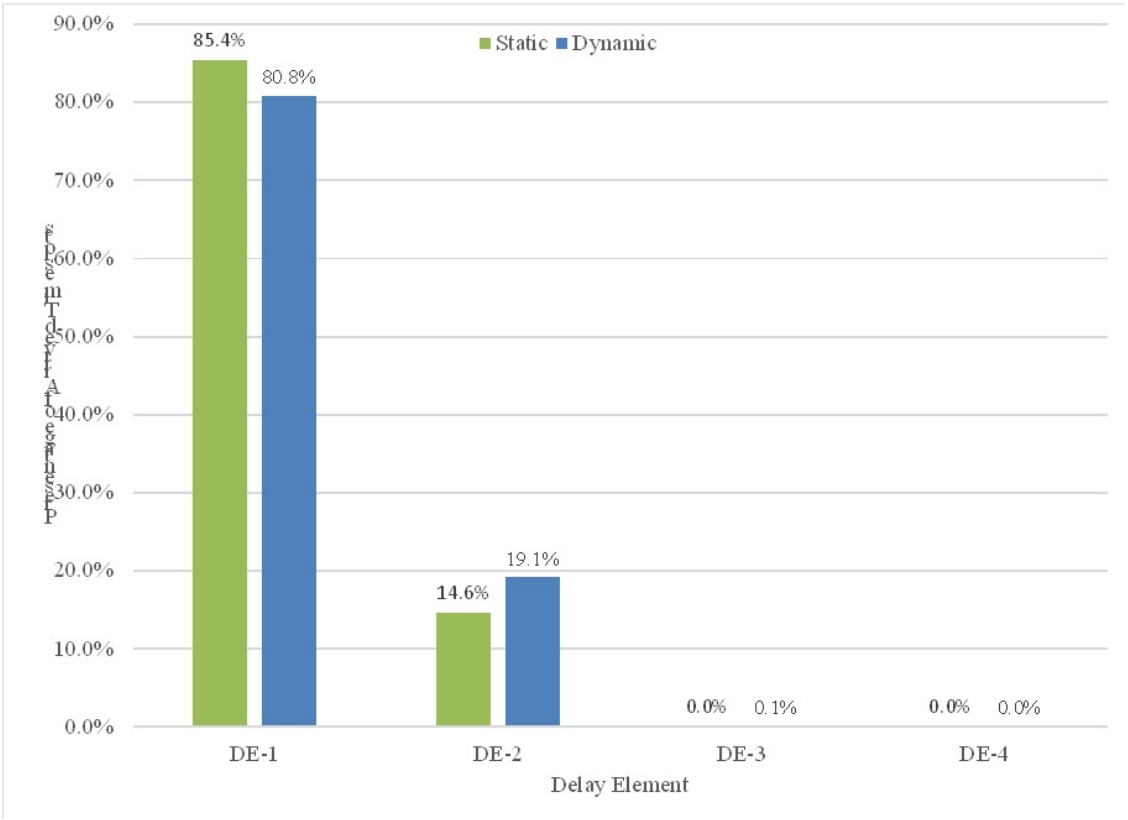


Figure 56: Comparison of the amount of timeslots that passes though every DE between SSWA and DSWA using DPAA when $S = 4$

Table 9: Simulation summary of DPAA using DSWA when $S = 4$

						DE 0	**DE 1**	**DE 2**	**DE 3**
	Source (created)	Sink (revieved)	Splitter 0 (arrived)	Splitter 1 (arrived)	Splitter 2 (arrived)	Splitter 11 (arrived)	Splitter 12 (arrived)	Splitter 21 (arrived)	Splitter 22 (arrived)
# Timeslots	4608	4608	4608	4604	4	3724	880	4	0
% Timeslots	100%	100%	100%	99.9%	0.1%	80.8%	19.1%	0.1%	0.0%

Table 10: Simulation summary of DPAA using SSWA when $S = 8$

										DE 0	**DE 1**	**DE 2**	**DE 3**	**DE 4**	**DE 5**	**DE 6**	**DE 7**
	Source (created)	Sink (revieved)	Splitter0 (arrived)	Splitter1 (arrived)	Splitter2 (arrived)	Splitter11 (arrived)	Splitter12 (arrived)	Splitter21 (arrived)	Splitter22 (arrived)	Splitter111 (arrived)	Splitter112 (arrived)	Splitter121 (arrived)	Splitter122 (arrived)	Splitter211 (arrived)	Splitter212 (arrived)	Splitter221 (arrived)	Splitter222 (arrived)
#Timeslots	645120	645120	645120	645120	0	643872	1248	0	0	466656	177216	1248	0	0	0	0	0
%Timeslots	100%	100%	100%	100%	0.0%	99.8%	0.2%	0.0%	0.0%	72.3%	27.5%	0.2%	0.0%	0.0%	0.0%	0.0%	0.0%

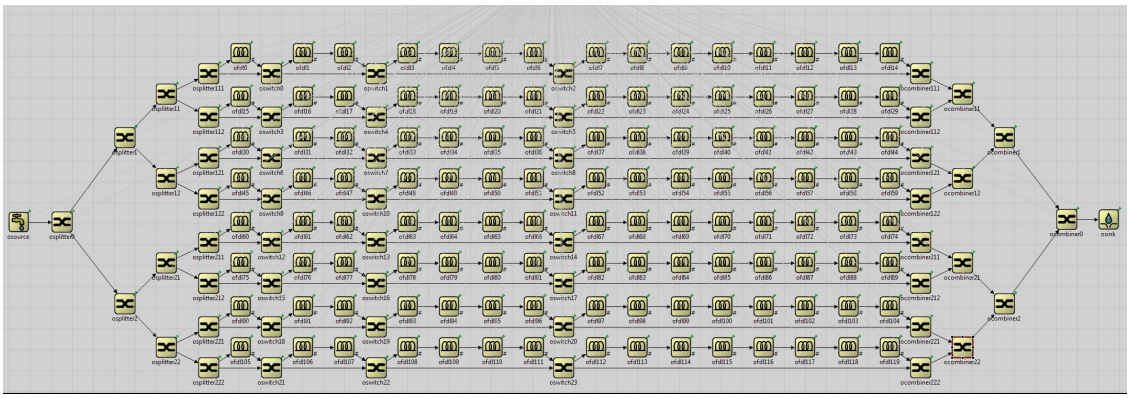


Figure 57: A screen shot of the simulation setup for $S = 8$

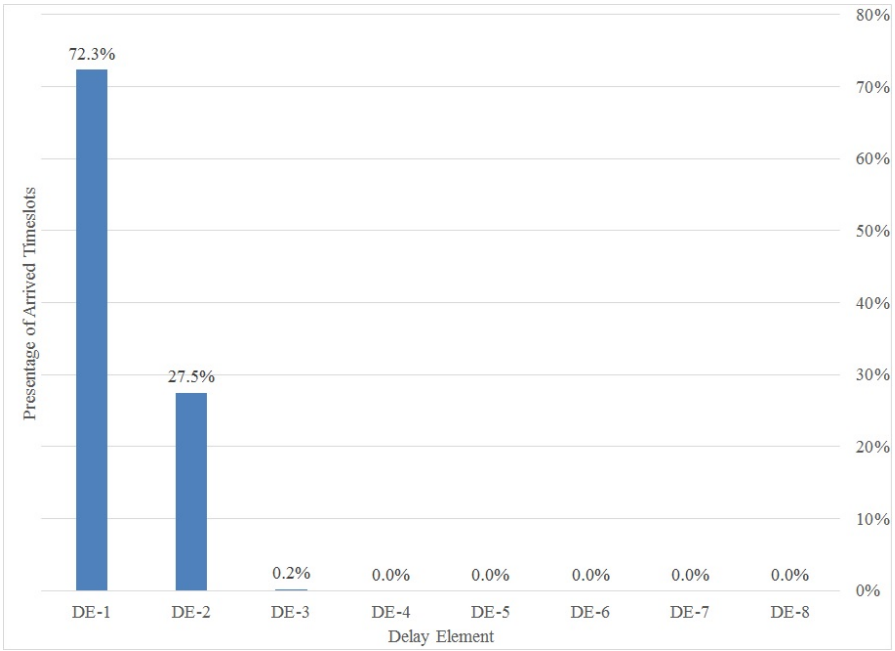


Figure 58: Percentage of traffic that passes through each DE using SSWA when $S = 8$ and DPAA is used

DSWA represents the worst case scenario when all connections are changed or terminated.

In this study, the P_b is set to be zero. However, when the footprint or cost limits the manufacturing process, using a single DE when $S = 4$ gives us P_b equal to 14.6% for SSWA and 19.2% for DSWA. On the other hand, when $S = 8$, $P_b = 0.002$ is achieved using 2 DEs.

7.0 ECONOMIC PATH ASSIGNMENT ALGORITHM

Low crosstalk SWM are available in the market at high cost. Power penalty (e.g. loss, attenuation and crosstalk) can be reduced by using special materials and sophisticated manufacturing process. When a switching fabric is made from low crosstalk, two optical signals may share a switch simultaneously without affecting each other. Using low crosstalk hardware components results in reducing the number of hardware components used to build the switching fabric, which results in increasing the overall availability [67]. On the other hand, adding more hardware to the fabric increases the cost, footprint and control algorithm complexity. Meanwhile, a simple control algorithm reduces the fabric development and manufacturing time [67].

The Economic Path Assignment Algorithm (EPAA) is introduced to be used when low crosstalk hardware is used to build a switching fabric. In EPAA, the dilation constraint is relaxed and two optical signals (represented in timeslots) are allowed to share any switch simultaneously if, and only if, both timeslots require the same switching state. The following sections presents an EPAA algorithm followed by the simulation setup; lastly, a discussion about the output is presented.

7.1 EPAA ASSIGNMENT ALGORITHM

EPAA is similar to DPAA; it uses the same concept and matrices. However, the algorithm itself is a bit complicated compared to DPAA. Once a simulation kernel assigns EPAA to be the algorithm to find a path for all incoming timeslots during the initialization phase, and S_o^{out} is determined, the controller sends the following method to *selectPath* class:

$pathIndex \leftarrow$ selectPath (δ_i^o, S)

This method returns the DE_i that S_i^{in} will follow. In addition to the two subroutines presented in section 6.1, an additional subroutine is used to determine a path for an incoming timeslots. This subroutine is *validateConflect*, and it is discussed below.

bool *validateConflict(startHoldingTime, swm_i, SWC_i)*: The controller calls this subroutine when two timeslots are scheduled to arrive at a given SWM simultaneously at the same *startHoldingTime*. If both timeslots require the same switching state SWC, the subroutine returns FALSE, and returns TRUE otherwise. Since jobs are stored in a FIFO queue, a queue iterator is used to traverse throughout the queue as it appears in Algorithm 7.1.

Algorithm 5 Validate conflict algorithm

validateConflict (SHT, swm_i, swc_i) **Input** : SHT

Output $onflictFound$

```
bool conflictFound=false
queue::Iterator it = queue::Iterator ( it->SWCQ, 0)
queue::Iterator itEnd = it.end()
SwitchingCont * SWC
do
    SWC = ( SwitchingCont *) it()
    if SWC->startHoldingTime==startHoldingTime then
        if SWC->switchingState!= switchingState then
            conflectFound = true
        end
    else
        it++
    end
while it!=itEnd || conflictFound==false
return conflictFound
```

EPAA has one additional variable to DPAA that is declared at the initialization phase, which is:

```
bool    SWConflict=true;
```

The algorithm runs exactly similar to DPAA, except in **Scenario III**, which is defined in EPAA as:

Scenario III: TSI is not empty and there is a SWC scheduled for at least one switch in the path. In this scenario, the algorithm calls the subroutine named $validateConflict$ $(startHoldingTime, swm_i, SWC_i)$ subroutine to determine whether a conflict occurs or not. If a conflict exist with a previously scheduled SWC, the algorithm will increase the path index by one, reset the boolean variables, and start from the beginning (after the initialization and declaration stages), as shown below.

```
pathSWQEmpty=true;
blocking=false;
pathIndex++;
SHT=0;
SWConflict=true;
```

However, if no conflict exists, the algorithm will return the selected DE_i to the control algorithm as it appears in the flow chart in Figure 59 and the pesudocode is shown in Algorithm 6. For additional explanation, a numeric example is presented below.

Numeric example: Assuming that f_0 is assigned to swa_0, such that [(0)(1)(2)(3)], from Figure 73. In addition, assuming that all timeslots but S_3 departed the fabric and S_3 is using DE_0. According to the control algorithm, the arrival time of S_3 is $t = 4$. Meanwhile, assuming that f_1 is assigned to swa_6, such that [(01)(2)(3)] from Figure 73 and the arrival time is $t = 5$. During the guard-time prior to S_0 in f_1, the controller is trying to find a path for S_0 to be switched to S_1, from equation 4.1. The required delay to perform the interchanging process is $\delta_0^1 = 5$.

Now the controller will verify if blocking exist or not, such that two SWC for a given switch require a different switching state. This is achieved by XOR-ing the two SWCs corresponding to $\delta = 4$ for path 0 (DE_0), Figure 77, and $\delta = 5$ for the same path, Figure 78, as shown below.

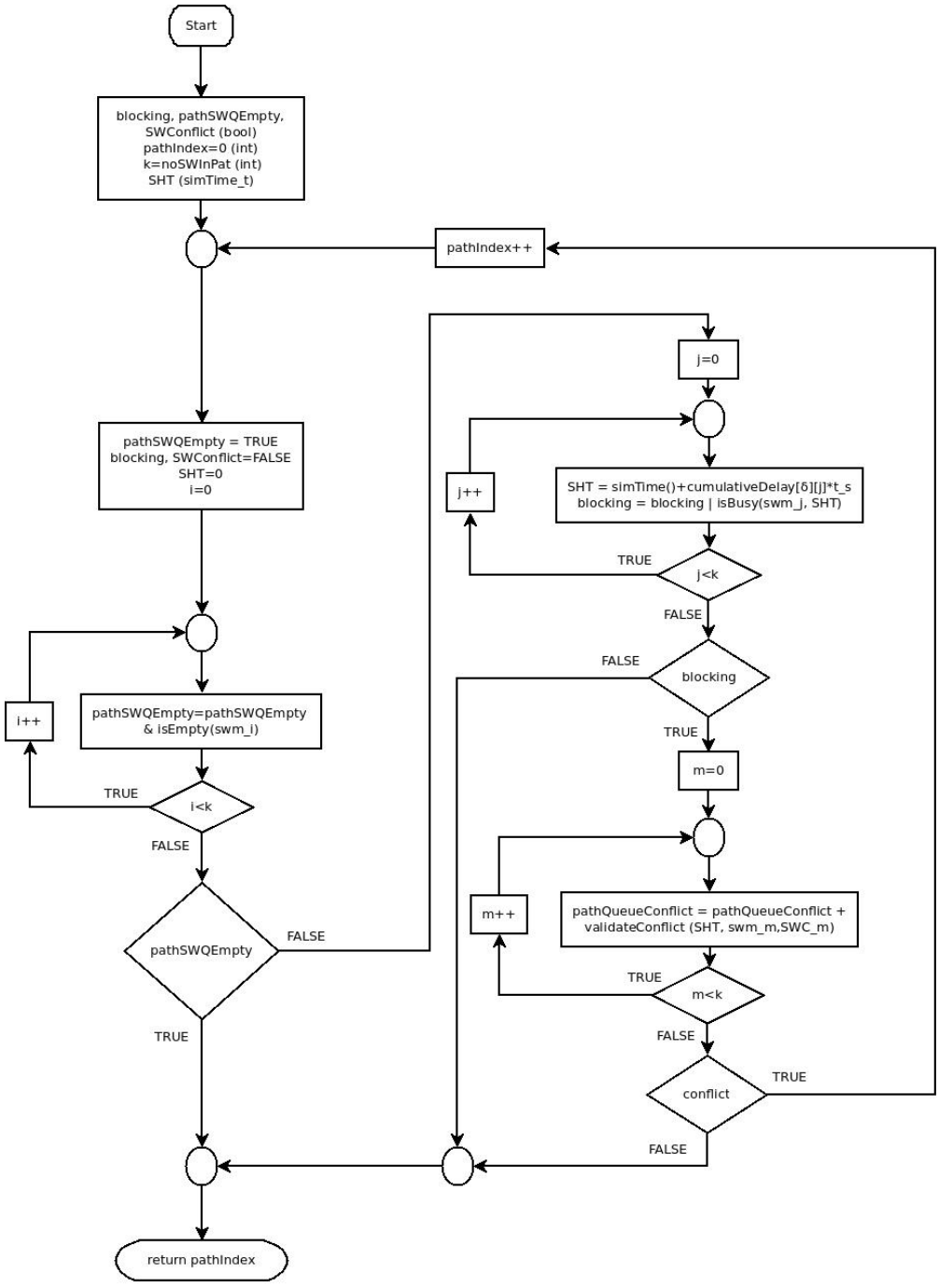


Figure 59: Flow Chart for EPAA

Algorithm 6 EPAA algorithm

selectPath (δ, S) **Input** : A nonnegative integer δ

Output pathIndex

.

bool blocking, pathSWQEmpty, SWConflict, pathFound=false

int pathIndex=0,k=getNoSwitchesInPath()

simtime_t SHT

do

blocking = SWConflict = false pathQueueIsEmpty=true SHT=0

for $i \leftarrow 0$ **to** $i < k$ **do**

pathSWQEmpty= pathSWQEmpty & isEmpty(getModule(pathIndex,i))

end

if *pathSWQEmpty==true* **then**

pathFound = true

else

for $j \leftarrow 0$ **to** $j < k$ **do**

SHT = simTime() + cumulativeDelay[δ][j] * TimeslotDuration

blocking= blocking | isBusy(getModule(pathIndex,j),SHT)

end

if *blocking==false* **then**

pathFound = true

else

for $m \leftarrow 0$ **to** $m < k$ **do**

SWConflict = SWConflict | validateConflicted

(SHT, (getModule (pathIndex, m), getSWC(δ, m));

end

if *SWConflict==true* **then**

pathIndex++

else

pathFound=true

end

end

end

while *pathIndex<S && pathFound==false*;

$$11010111$$

$$11100111 \oplus$$

$$00\boxed{11}0000$$

If the XOR-ing operation results in (1), then blocking exists and an additional process is needed. On the other hand, the (0) value means blocking does not exist, regardless of the arrival time to the switch. In this example, blocking occurs on the third and fourth SWM (Splitter11 and Switch10 in Figure 55). Since blocking is likely to occur, the arrival time of each timeslot to the blocking switch must be investigated. Conflict happens when two timeslots arrive at the same SWM simultaneously. In this example, the arrival time at the third and fourth switch (swm_2 and swm_3) is investigated below using equation 5.1:

Cumulative delay for $\delta = 4$: $00\boxed{00}0444$

Cumulative delay for $\delta = 5$: $00\boxed{01}1555$

Assuming that $t_s = 1$ for all timeslots, then: the arrival time for S_3 and S_0 at swm_2 is:

S_3: $3 + 0 * 1 = \boxed{3}$ $\qquad$ S_0: $4 + 0 * 1 = \boxed{4}$

And the arrival time for S_3 and S_0 at swm_3 is:

S_3: $3 + 0 * 1 = \boxed{3}$ $\qquad$ S_0: $4 + 1 * 1 = \boxed{5}$

Because S_3 and S_0 do not meet at either swm_2 or swm_3 at the same time, the algorithm concludes that no conflict occurs and the path can be reserved.

7.2 SIMULATION SETUP AND RESULTS

EPAA is tested using different SWAs and a different number of timeslots per frame, as follows:

1. Simulating EPAA using DSWA when $S = 4$.
2. Simulating EPAA using SSWA when $S = 8$.

Note that simulating EPAA using SSWA when $S = 4$ is omitted because it is included in DSWA. In other words, SSWA is a special case of DSWA. The simulation output is described below.

Table 11: Simulation summary of EPAA using DSWA when $S = 4$

						DE 0	DE 1	DE 2	DE 3
	Source (created)	Sink (revieved)	Splitter 0 (arrived)	Splitter 1 (arrived)	Splitter 2 (arrived)	Splitter 11 (arrived)	Splitter 12 (arrived)	Splitter 21 (arrived)	Splitter 22 (arrived)
# Timeslots	4608	4608	4608	4608	0	4536	72	0	0
% Timeslots	100%	100%	100%	100%	0%	98.44%	1.56%	0%	0%

7.2.1 EPAA using DSWA when $S = 4$

The simulation setup is identical to DPAA using DSWA when $S = 4$, subsection 6.2.2. The total number of simulation runs using SSWA is $SWA = 576$. In each simulation run, there are 8 timeslots generated by the source, giving a total of $4,608$ timeslots generated in 576 runs. All of the generated timeslots are received by the Sink, as appears in Table 11.

In Table 11, in the present of DPAA all generated timeslots used two paths instead of three paths . Hence, all timeslots used either DE_0 and DE_1. In fact, if the blocking requirement is less than 2%, a single DE is capable of satisfying this requirement. The small number of timeslots that used DE_1 are investigated in section 7.3.

The results shows that, when EPAA is used, two DEs provide nonblocking TSI, compared to three when using DPAA. Figure 60 compares both algorithms.

7.2.2 EPAA using SSWA when $S = 8$

This simulation setup is identical to DPAA using SSWA when $S = 8$, section 6.2.3. The total number of simulation runs using SSWA is $40,320$. In each simulation run, there are 16 timeslots generated by the source, giving a total of $645,120$ timeslots generated in $40,320$ runs. All of the generated timeslots are received by the Sink, as appears in Table 12.

In Table 12, the total traffic is divided between two DEs (DE_0, and DE_1). In this simulation, eliminating DE_2 though DE_7, Figure 57, does not affect the value of P_b. Note that EPAA uses one less DE compared to DPAA, Figure 61. In fact, using a single DE when $S = 8$ in the presence of EPAA provides $P_b < 7$.

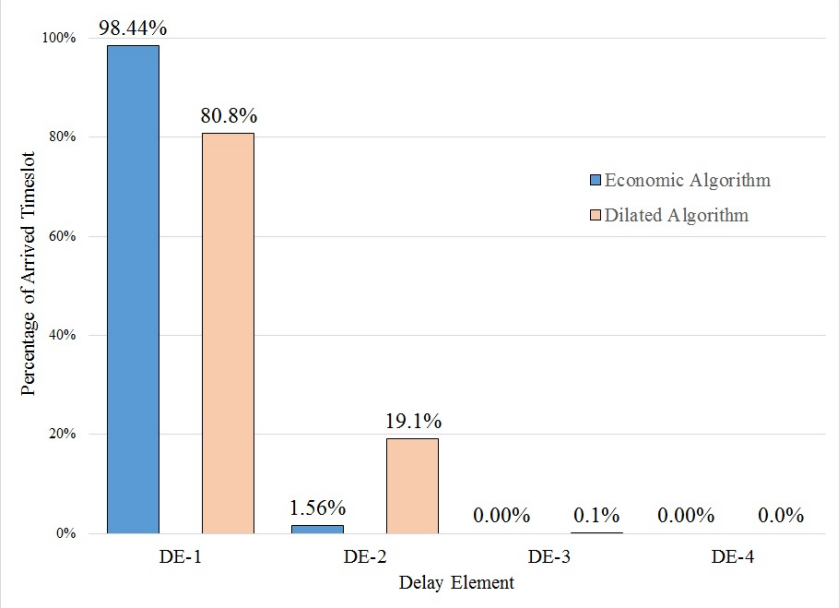


Figure 60: Comparision between DPAA and EPAA for $S = 4$

7.3 DISCUSSION

For S = 4, there were only 72 number of simulation runs out of 576 that required the second delay element, Figure 60. In all of the 72 simulation runs, only 1 timeslot out of 8 (4 S * 2 Frames) timeslots is directed to the second DE. Therefore, a single DE TSI has a probability of blocking equal to .125, while 2 DEs have 0.0 probability of blocking (nonblocking fabric). After investigating the output, blocking occurs in 2 scenarios: blocking that occurs at the switches and blocking that occurs at the FDL. Both scenarios are discussed below.

7.3.1 Switch Blocking

There are 36 simulation runs out of 576 (6.25%) that fall under this category. Blocking happens when 2 timeslots are directed to the same switch each requiring a different switching state. Since the controller is programmed to avoid blocking, the controller will direct the newer timeslot to the next DE. After investigating the output, this scenario happens when

Table 12: Simulation summary of EPAA using SSWA when $S = 8$

										DE 0	**DE 1**	**DE 2**	**DE 3**	**DE 4**	**DE 5**	**DE 6**	**DE 7**
	Source (created)	Sink (revieved)	Splitter0 (arrived)	Splitter1 (arrived)	Splitter2 (arrived)	Splitter11 (arrived)	Splitter12 (arrived)	Splitter21 (arrived)	Splitter22 (arrived)	Splitter111 (arrived)	Splitter112 (arrived)	Splitter121 (arrived)	Splitter122 (arrived)	Splitter211 (arrived)	Splitter212 (arrived)	Splitter221 (arrived)	Splitter222 (arrived)
#Timeslots	645120	645120	645120	645120	0	645120	0	0	0	601920	43200	0	0	0	0	0	0
%Timeslots	100%	100%	100%	100%	0%	100%	0%	0%	0%	93.0%	7.0%	0%	0%	0%	0%	0%	0%

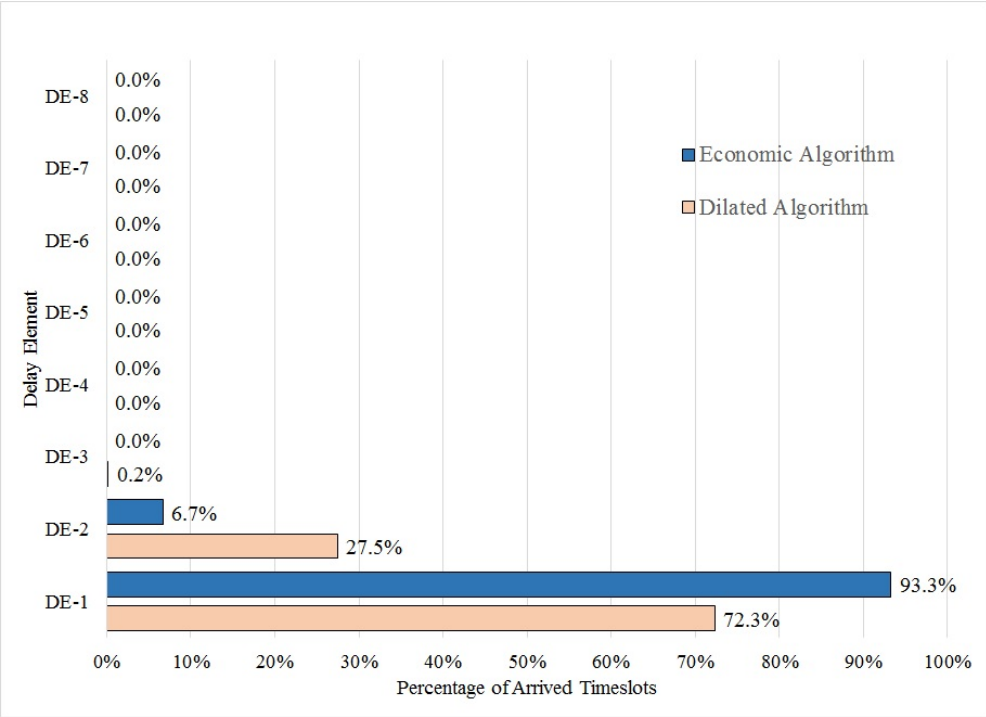


Figure 61: Comparison between DPAA and EPAA for $S = 8$

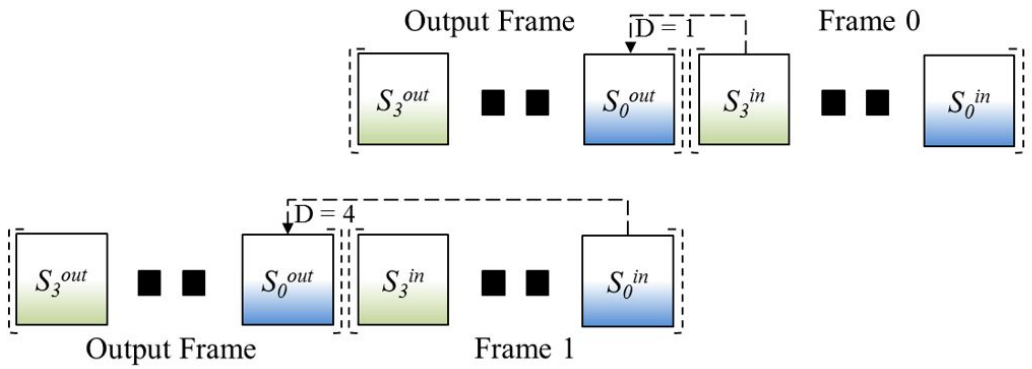


Figure 62: Switch blocking at TSI for S = 4

S3 from F0 is switched to S0 in the same frame (D = 1), while S0 in the next frame (F1) remains in its position (switched to the same index, D= 4), Figure 62. The blocking happens at SW0. To avoid blocking, the controller directs S0 on F1 to the next (second) DE.

To better understand this blocking scenario, it is described starting from the arrival of the last timeslot (S3) from frame 0 (F0) and the arrival of the next timeslot (S0) in frame 1 (F1). At time t=3 (arrival time of S3 on F0), S3 enters DE0 to be switched to S0 at the next frame (delay required to perform this switching is 1* TSdur), Figure 63 (top). At time t=3 + GT, S3 enters FDL0, Figure 63 (bottom).

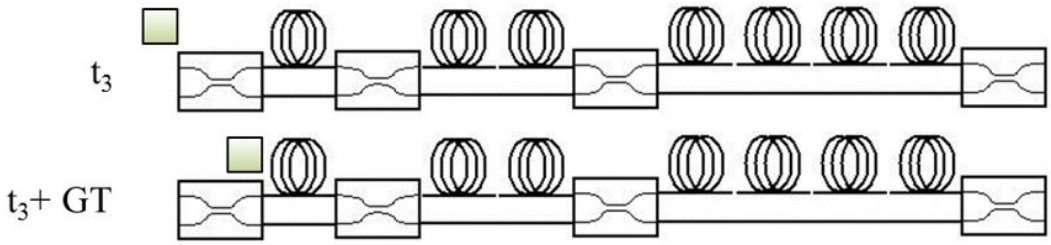


Figure 63: walkthrough for switch blocking scenario for S = 4 - Part 1

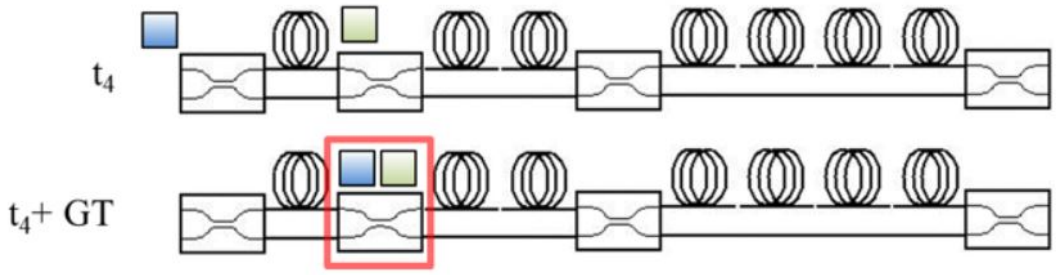


Figure 64: walkthrough for switch blocking scenario for S = 4 - Part 2

At time t=4 (arrival time for S0 on F1), S0 enters DE 0 to be switched to S0 at the next frame (delay required to perform this switching is 4* TSdur), the previous timeslot exits FDL0 after being delayed by a duration of one timeslot, heading to exit DE 0; meanwhile, timeslot S0 on F1 is heading to FDL4 to be delayed by 4 timeslot duration, Figure 64 (Top). At t=4+GT, both timeslots arrive to SW0 at the same time but each requires a different switching state, Figure 64 (bottom). S3 (which is S0 after the delay on F0) requires CROSS switching state, while S0 on F1 requires BAR switching state. Hence, the controller will not allow this scenario to happen and will direct the newer timeslot (S0 on F1) to the next DE 1 to avoid blocking.

7.3.2 FDL Blocking

There are 36 runs out of a total 576 (6.25%) that fall under this category. FDL blocking happens when 2 timeslots are scheduled to enter the same FDL simultaneously. The controller is programmed to avoid blocking by directing the newer timeslot to the next DE.

After investigating the output, this scenario happens when S3 from F0 is switched to S2 in the same frame (D = 3), while S0 in the next frame (F1) is switched to S2 in the same frame (D=6). In this case to avoid collusion (blocking) the controller directs S0 on F1 to the second DE, Figure 65. Switching S0 at F0 requires 2 FDL stages, FDL0 (D=1) + FDL1 (D=2) while S0 at F1 requires 2 FDL stages, FDL1 (D=2) + FDL2 (D=4). To better understand this blocking scenario, it is described starting from the arrival of the last

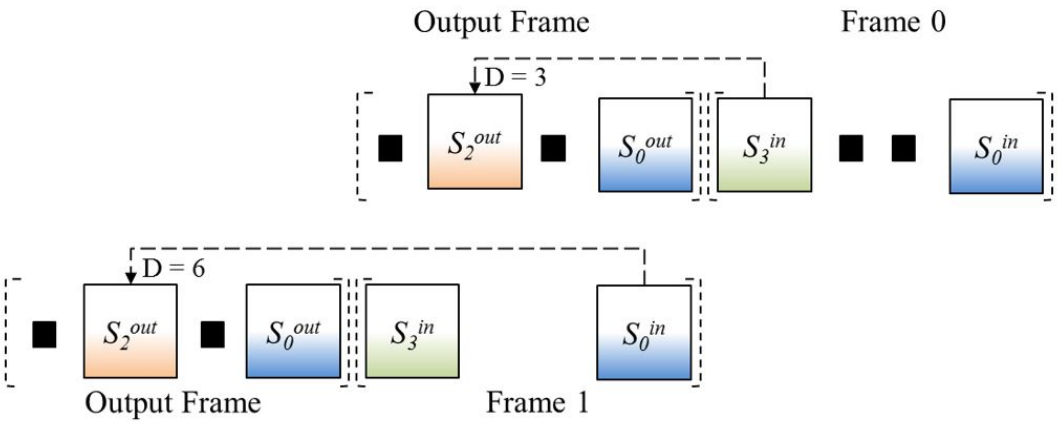


Figure 65: FDL blocking at TSI for S = 4

timeslot (S3) at frame 0 (F0) and the arrival of the next timeslot (S0) at frame 1 (F1).

At time t=3 (arrival time of S3 on F0), S3 enter DE 0 to be interchanged with S2 (delay required to perform this switching is 3*Tdur), Figure 66 (top). At time t=3 + GT, S3 enters FDL0, Figure 66 (bottom). At time t=4 (the arrival time for S0 on F1) , S0 enter DE 0 to be switched with S2 (delay required to perform this switching is 6*Tdur), the previous timeslot exits FDL0 heading to FDL1, meanwhile, Timeslot S0 on F1 is also heading to FDL1 to be delayed by 2, Figure 67 (top). At t=4+GT both timeslots arrive to SW0 at the same time

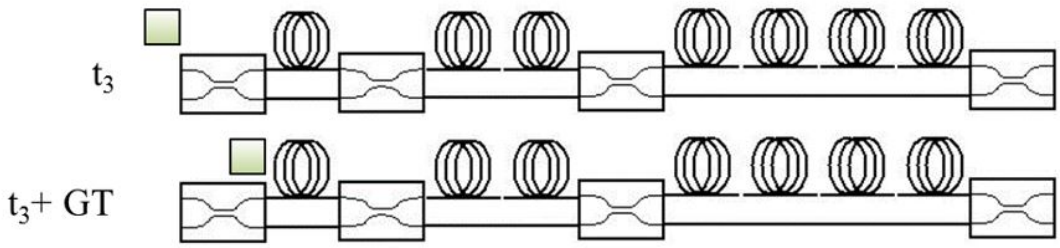


Figure 66: walkthrough for FDL blocking scenario for S = 4 - Part 1

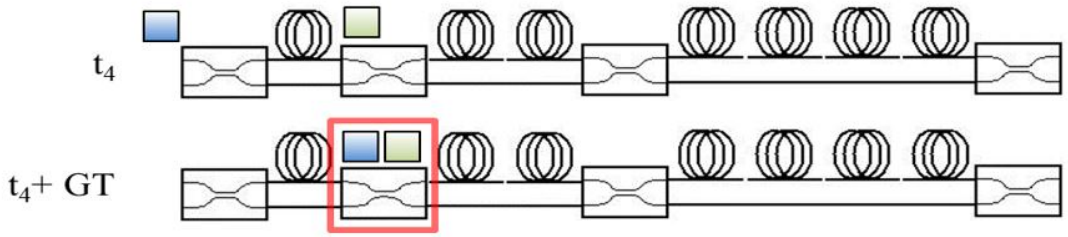


Figure 67: walkthrough for FDL blocking scenario for S = 4 - Part 2

but each require 2 different switching state, Figure 67 (bottom). S3 requires BAR switching state, while S0 on F1 requires CROSS switching state. Hence, the controller will direct the newer timeslot (S0 on F1) to DE 1 to avoid blocking.

7.4 POWER PENALITY FOR DPAA VERSUS EPAA

Poor hardware, including switches and FDLs, are manufactured with low quality materials and basic manufacturing process. The result are hardware with high crosstalk and high insertion loss. On the other hand, when high quality materials are used and sophisticated manufacturing process is practice, switches and FDLs have low crosstalk and insertion loss. Crosstalk can be as high as 50 dB and as low as 30 dB. Meanwhile, insertion loss can be as high as 10 dB and as low as 3 dB. Signal attenuation that results from FDLs is omitted because is a tiny number ($<$ 05 dB when $S = 4$).

Both path assignment algorithms: DPAA and EPAA, are simulated after adding crosstalk and insertion loss to the hardware components for $S = 4$. The purpose is to ensure that DPAA is a real dilated algorithm and the accumulated crosstalk is always minimal. On the other hand, EPAA adds crosstalk to the optical signals.

The simulation is run under these parameters: transmitted power = 0 dBm, insertion loss (A) = 5 dB, crosstalk (X) = 40 dB and the total L. The numbers are not as important as the results in Figure 68.

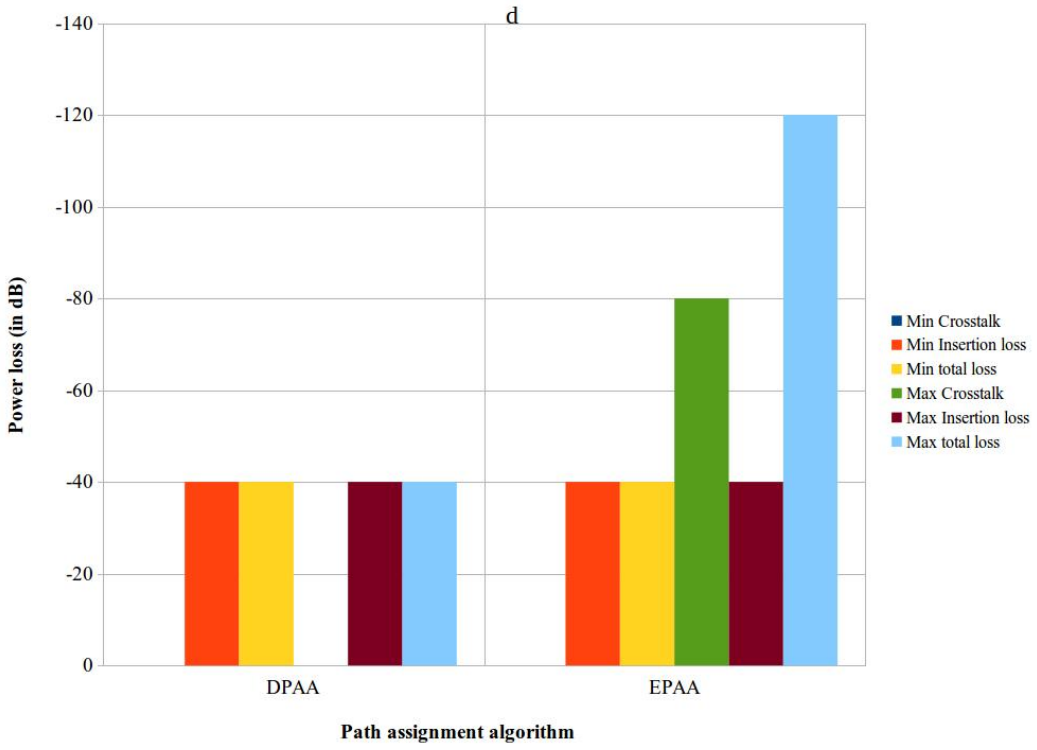


Figure 68: Comparison between the amount of power loss in DPAA and EPAA

Table 13: Algorithm complexity for all algorithms presented in this document

	Algorithm Name				
	isEmpty	**isBusy**	**validateConflict**	**DPAA**	**EPAA**
Complexity	O($DE.k$)	O($DE.k$)	O($DE.k$)	O(($DE.k^2$)^2)	O(($DE^2.k^3$)^2)

From the figure, the minimum crosstalk for both algorithms is 0 dB [1], while the maximum crosstalk is 80 dB when using EPAA. The added crosstalk to the insertion loss results on high total loss that appears in EPAA. Reducing the total loss on EPAA requires more expensive hardware. On the other hand, using DPAA requires extra hardware components.

7.5 ALGORITHM COMPLEXITY FOR DPAA VERSUS EPAA

There are number of algorithms that contribute to this dissertation, six algorithms are presented in this document. Algorithm 2 is simply the sequence of events that take place during the guard-time period. Hence, the complexity is omitted. The two major algorithms are: DPAA and EPAA, the rest of the algorithms are called from these major algorithms.

Assuming that the number of switches that a timeslot will pass though from source to destination is k. In addition, assuming that the number of available alternative paths that a timeslot can be assigned to is DE. Then the algorithms' complexity is presented in table 13.

The major algorithms (DPAA and EPAA) are poorly implemented according to the table. That is acceptable in this case because the initiative was not to implement a perfect algorithm but to operate the switching fabric. Hence, these algorithms requires massive improvement in order to speed the algorithms up.

[1] Crosstalk is stated to be zero for simulation purposes, however, crosstalk always exist. Minimizing crosstalk result on neglectable crosstalk values

8.0 SPACE/TIME SWITCHING WITH FRAME INTEGRITY

This chapter proposes a control algorithm for a space/time nonblocking switching fabric that was proposed in [3]. The proposed switching fabric claimed to be a rearrangably nonblocking fabric with frame integrity. However, as the previous timeslot intechanger in Chapter 5, this claim is not verified as of the date of writing this dissertation. The proposed control algorithm operates a two space input and two timeslots per frame <2,2> switching fabric in Figure 69. The switching fabric supports 4 channels (2 spaces x 2 timeslots/frame). This control algorithm has the ability to accommodate two simultaneous connections into the switching fabric without blocking, in addition to existing connections for prior timeslots. It is important to mention that this fabric is complicated for the following reasons:

1. There are two independent synchronized inputs. The controller must find a path for each timeslot simultaneously without blocking at any switches, keeping in mind that every incoming timeslot for each space has an independent switching assignment.
2. If both inputs are trying to communicate to the same output node, then the controller will terminate both connections. The term "busy" is used to refer to this scenario.
3. In some cases, the controller must change an existing connection's path in order to accommodate new incoming timeslots. Thus, the controller must keep a real-time log for the network state and update it log as new connections are established. The rearrangement scenario is verified manually due to the code complexity and the limited number of such scenarios.

Because it is the first control algorithm for this switching fabric, it starts with a "toy" network to verify the initial claim and that is the main motivation for this work. In the following sections, the switching fabric control algorithm is presented first. Then, the path

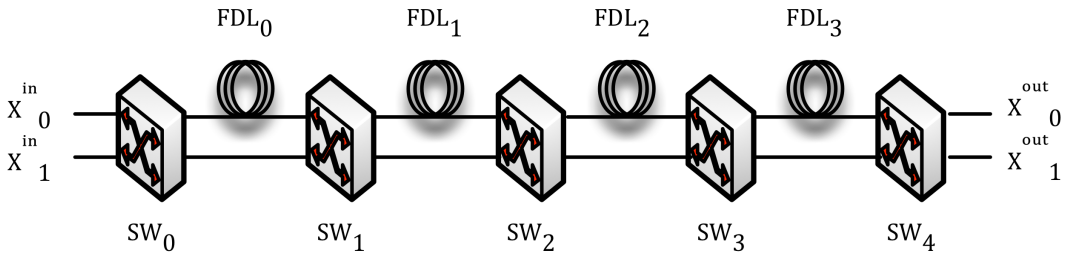


Figure 69: A Rearrangeable nonblocking <2, 2> Switching fabric with frame integrity

assignment for this fabric is also presented. The results and a discussion about these algorithms concludes this chapter.

8.1 SPACE/TIME CONTROL ALGORITHM (STCA)

The control algorithm for the space/time switching fabric in Figure 69 is implemented using the same concept used in the time only switching fabric in chapter 5. The switching fabric consists of two types of hardware components: 2x2 active switches and FDLs. Switches are labeled SW_0 to SW_4, Figure 69, while FDLs are labeled FDL_0 to FDL_3. The two space input ports are labeled X_0^{in} & X_1^{in}, while the space output ports are labeled X_0^{out} & X_1^{out}.

The main concept is to convert the switching fabric into binary matrices and perform ordinary binary operations. The sequence of operations is identical to the sequence of operations for the timeslot interchanger in chapter 5. The difference between both control algorithms is in the following concepts:

- **Output Matrix:** There are two space input ports and two space output ports in this fabric; each space input generates two consecutive two timeslots per frame. Hence, the total number of input channels = 2 x 2 = 4 channels.

  Each frame has an independent switching assignment. All possible output candidates are

tested to ensure the result is valid. One of the following output candidates is assigned to each frame during the initialization of the simulator:

- Timeslots are not switched in either time or space domain [0,0].
- Timeslots are switched in time but not space domain [1,0].
- Timeslots are not switched in time but in space domain [0,1].
- Timeslots are switched in both time and space domain [1,1].

The corresponding SWA matrix for the above output candidates is presented in Figure 70. The index of the first two columns correspond to the input timeslot index, while the values corresponds to the output timeslot index. The third column's value corresponds to the output space index. Row indexes represent the SWA index. For instance, assuming that swa_3 is assigned to f_0 on X_0^{in}, from Figure 70, $S_0^{in} \rightarrow S_1^{out}$ and $S_1^{in} \rightarrow S_0^{out}$ and both are destined to space output X_1^{out}.

Note that to maintain frame integrity, each frame is delayed by a duration of a frame, as discussed earlier in Chapter 4. Hence, when only space switching is required, each timeslot is also delayed by the duration of a frame. In addition, because channels are grouped into frames, when space switching is required, the entire frame exits the network at a single output port. Allowing timeslots to be switched independently adds more complexity to the control algorithm, thus it is left for future work.

- **Paths Switching Control Matrices:**

  There are multiple of paths for each timeslot to follow in the space/time switching fabric. Each path depends mainly on the delay required and the output space for each timeslot. The available paths are grouped into three main categories based on the delay required to perform the time switching operation. Each one of these categories is divided into two subcategories based on whether the timeslot requires space switching or not. All paths are presented below based on the value of δ:

  - $\delta = T_{dur}$**:** A path falls under this category if the required delay for a timeslot is $\delta = T_{dur}$. The total available paths to switch a timeslot from S_1^{in} to S_0^{out} is four paths for each space input. The SWC matrix for each space input depends on whether space switching is required or not, as discussed below:

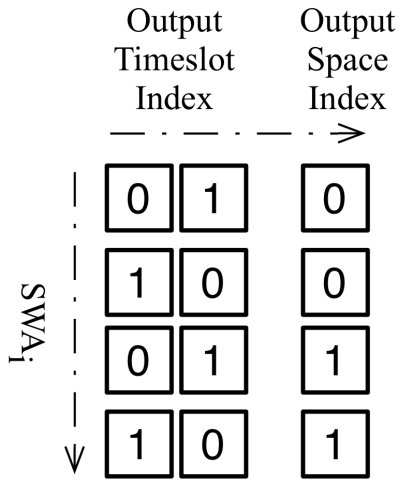


Figure 70: A SWA matrix for <2,2> switching fabric with frame integirity

If no space switching is required: then an incoming timeslot from X_0^{in} can be delayed at one of the four FDL stages in Figure 69, and exits at the same output port X_0^{out}. The SWC matrix that corresponds to this subcategory is found in the top half matrix of Figure 82 in Appendix E. Meanwhile, an incoming timeslot from X_1^{in} can be also delayed at one of the four FDL stages in the same figure and exits the same space output port X_1^{out}. The SWC matrix for that corresponds to this subcategory is found in the top half matrix of Figure 83 in Appendix E. The total number of paths for X_0^{in} without space switching is our, and four paths are available for X_1^{in}.

Space switching is required: then an incoming timeslot from X_0^{in} can be delayed at one of the four FDL stages in Figure 69 and exits at the opposite space output port X_1^{out}. The SWC matrix that corresponds to this subcategory is found in the bottom half matrix of Figure 82. Meanwhile, an incoming timeslot from X_1^{in} can also be delayed at one of the four FDL stages in the same figure, but exits at the opposite space output port X_0^{out}. The SWC matrix that corresponds to this subcategory is

found in the bottom half matrix of Figure 83.

– $\delta = 2 * T_{dur}$**:** A path falls under this category if the required delay for a timeslot is $\delta = 2 * T_{dur}$. The total available paths to switch a timeslot from S_0^{in} to S_0^{out} or S_1^{in} to S_1^{out} is six paths for each space input. The SWC matrix for each space input depends on whether space switching is required or not, as discussed below:

No space switching is required: then an incoming timeslot from X_0^{in} can be delayed at one of the following six combinations of FDL stages in Figure 69 and exits at the same output port X_0^{out}: (i)FDL stage 0 and 1, (ii)FDL stage 0 and 2, (iii)FDL stage 0 and 3, (iv)FDL stage 1 and 2, (v)FDL stage 1 and 3, (vi)FDL stage 2 and 3. The SWC matrix that corresponds to this subcategory is found in the top half matrix of Figure 84 in Appendix E. Meanwhile, an incoming timeslot from X_1^{in} can also be delayed at one of the six combinations of FDL stages in the same figure and exits the same space output port X_1^{out}. The SWC matrix that corresponds to this subcategory is found in the top half matrix of Figure 85 in Appendix E.

Space switching is required: then an incoming timeslot from X_0^{in} can be delayed at one of the six combinations of FDL stages in Figure 69 and exits at the opposite space output port X_1^{out}. The SWC matrix that corresponds to this subcategory is found in the bottom half matrix of Figure 84. Meanwhile, an incoming timeslot from X_1^{in} can also be delayed at one of the six combinations of FDL stages in the same figure, but exits at the opposite space output port X_0^{out}. The SWC matrix for that, corresponds to this subcategory found in the bottom half matrix of Figure 85.

– $\delta = 3 * T_{dur}$**:** A path falls under this category if the required delay for a timeslot is $\delta = 3 * T_{dur}$. The total available paths to switch a timeslot from S_0^{in} to S_1^{out} is four paths for each space input. The SWC matrix for each space input depends on whether space switching is required or not, as discussed below:

No space switching is required: then an incoming timeslot from X_0^{in} can be delayed at one of the following four combinations of FDL stages in Figure 69 and exits at the same output port X_0^{out}: (i)FDL stage 0,1,2, (ii)FDL stage {0,1,3}, (iii)FDL stage {0,2,3}, (iv)FDL stage {1,2,3}. The SWC matrix that corresponds to this subcategory is found in the top half matrix of Figure 86 in Appendix E. Meanwhile,

an incoming timeslot from X_1^{in} can also be delayed at one of the four combinations of FDL stages in the same figure and exits the same space output port X_1^{out}. The SWC matrix that corresponds to this subcategory is found in the top half matrix of Figure 87 in Appendix E.

Space switching is required: then an incoming timeslot from X_0^{in} can be delayed at one of the four combinations of FDL stages in Figure 69 and exits at the opposite space output port X_1^{out}. The SWC matrix that corresponds to this subcategory is found in the bottom half matrix of Figure 86. Meanwhile, an incoming timeslot from X_1^{in} can also be delayed at one of the four combinations of FDL stages in the same figure, but exits at the opposite space output port X_0^{out}. The SWC matrix for that, corresponds to this subcategory found in the bottom half matrix of Figure 87.

- **Cumulative Delay Matrix:** CDM is described in section 5.2.2. This fabric has its own CDM that is used to schedule SWC for the entire path. The complete CDM is presented in Appendix D. The matrix can be divided intro three parts: The first part represents the CDM for the four available paths for $\delta = 1$ (rows 0 to 3). The second part represents the CDM for the six available paths for $\delta = 2$ (rows 4 to 9). Lastly, the third part represents the CDM for the four available paths for $\delta = 3$(rows 10 to 13).

In this study, the assumptions are identical to the onse used in the time only switching fabric in section 5.1. The only difference is in the number of timeslots per frame, where $S = 2$ is used only.

8.2 SPACE/TIME PATH ASSIGNMENT ALGORITHM

The Space/Time Path Assignment Algorithm (STPAA) is implemented using similar concepts to those used in DPAA and EPAA. The difference between finding a path for time switching and space/time switching is stated below:

- Instead of searching for a path for a single connection (timeslot) request, two simultaneous connections are searched for. This implies that both paths must not conflict (block)

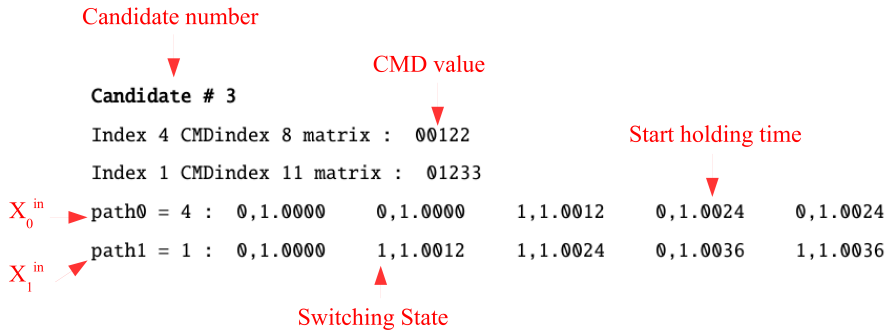


Figure 71: Sample of a candidate path

with each other at any switch.

- Unlike time only switching fabric, this control algorithm is a rearrangeably nonblocking fabric. This implies that if a new connection request cannot find a path in the fabric, the fabric switching table could be updated to accommodate the incoming connection request.

- In space/time switching fabric, space switching is likely to occur. If so, STPAA should ensure that no more than one connection is communicating with the destination at the same time.

In space/time switching fabric, a candidate set of paths is a group of two paths together each path carries incoming timeslot from different space input, Figure 71. As discussed earlier, for $\delta = 1, 2, 3$ there are 4, 6 and 4 possible paths, respectively. Each path in indexed from 1 to the number of available paths. For any two incoming timeslots, the number of candidate set of paths = available path for S_0^{in} from X_0^{in} x available path for S_0^{in} from X_1^{in}. For instance, if two incoming timeslots require $\delta = 3$ and $\delta = 2$, then there will be 4 x 6= 24 set of candidate paths. Each candidate path consist of the following: path index, the corresponding CDM index and its value, switching state, the start holding time for each switch.

The list of candidate paths is shorter than the multiplication of the number of available paths for both incoming timeslots from different space. This due to the elimination of the blocking paths by the controller. The controller eliminates each candidate if paths block with each other. For instance, the following candidate is eliminated because both paths block on four out of five switching stages, such that, the switching state for stage one is different for the same start holding time.

```
   Index 2 CMDindex 6 matrix :  01122
Index 1 CMDindex 5 matrix :  01222
path0 = 2 :  1,1.0000    0,1.0012    0,1.0012    0,1.0024    0,1.0024
path1 = 1 :  0,1.0000    1,1.0012    0,1.0024    1,1.0024    1,1.0024
```

8.3 RESULTS

The results confirm the original claim that the switching fabric is rearrangeably nonblocking. Since each frame has its own SWA, there are 4^2 possible output candidates per input. Thus, the total number of simulation runs is $4^4 = 256$. After simulating all switching assignments, the results can be categorized into three cases:

Case I, Busy case: In this case, both incoming timeslots are assigned to exit the fabric at the same space output port. Depending on the fabric's control algorithm, one of the following actions is performed: (i) reject both connection establishment requests, or (ii) allow one connection to be established and reject the other. Note that, this is not a blocking case, this case is similar to contention in packet networks. Around 75% of the switching assignments fall under this case as it appears in Figure 72.

Case II, Nonblocking case: In this case, both incoming timeslots have independent SWA and both find their way though the fabric to the destination space output port. About 22% of the switching assignments fall under this case as it appears in Figure 72.

Case III, Blocking case: In this case, both incoming timeslots cannot find their way throughout the fabric simultaneously. The network must change its current connections

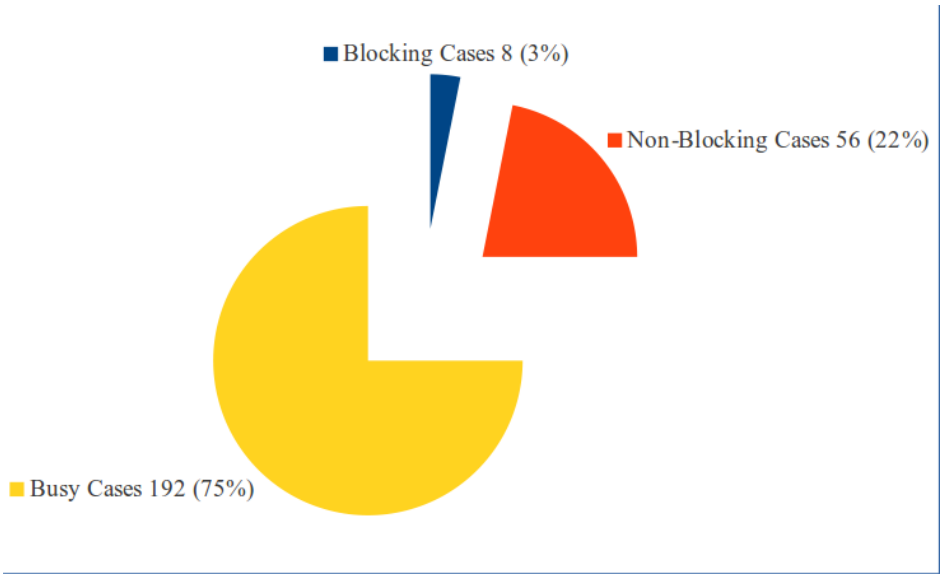


Figure 72: Simulation cases for space/time switching assignment algorithm

in order to accommodate the new connections. About 3% of the switching assignments fall under this case as it appears in Figure 72.

The following example is extracted from the simulation output, it will present the rearrangement process. In a simulation run, frames F_0 & F_1 from X_0 are assigned to swa_0 & swa_1, respectively. Whereas, frames F_0 & F_1 from X_1 are assigned to swa_3 & swa_3, respectively. during the guard-time period prior to timeslots S_0^{in} from both inputs (time t_0), the controller tries to find a path for the incoming timeslots. The controllers reads the output timeslot index for both incoming timeslots S_j^{out} and the output space for both incoming timeslots X_j^{out} from the SWA matrix in Figure 70; the results input/output space, timeslot, frame and delay is shown below.

$swa_0,\ X_0^{in},\ X_0^{out},\ S_0^{in},\ S_0^{out},\ f_0^{in},\ f_1^{out},\ \delta = 2$

$swa_3,\ X_1^{in},\ X_1^{out},\ S_0^{in},\ S_1^{out},\ f_0^{in},\ f_1^{out},\ \delta = 3$

Then, the controller starts finding a path for both incoming timeslots based on the value of δ. The controller starts by listing all candidate set of path for the incoming

timeslots. The candidate paths are all nonblocking paths. The controller insures that both incoming timeslots are not trying to exit the fabric at the same space output port simultaneously. The controller verifies switch blocking by the switching state and the time at which both timeslots arrive to each switch. If both timeslots arrive at any switch at the same time (same start holding time) and each require different switching states, the the controller considers this path a blocking path and starts the process at the next path. This approach is exactly similar to the one explained in chapters 6 and 7.

In this example, the controller start by candidate set of paths # 1 . Because this is the first incoming timeslot to the fabric, The controller reserves candidate set of paths # 1 (the box in the list of candidates means that path is reserved) and timeslots are inserted into the switching fabric at time $t_0 + t_{GT}$. The remaining 9 candidate paths are discarded since a path is found.

```
======== ALL candidates ============
Candidate # 1
Index 1 CMDindex 5 matrix :  01222
Index 4 CMDindex 14 matrix :  00123
path0 = 1 :  1,1.0000    1,1.0012    0,1.0024    1,1.0024    0,1.0024
path1 = 4 :  1,1.0000    0,1.0000    1,1.0012    1,1.0024    0,1.0036

----------------------
Candidate # 2
Index 3 CMDindex 7 matrix :  01112
Index 4 CMDindex 14 matrix :  00123
path0 = 3 :  1,1.0000    0,1.0012    1,1.0012    1,1.0012    1,1.0024
path1 = 4 :  1,1.0000    0,1.0000    1,1.0012    1,1.0024    0,1.0036
----------------------
Candidate # 3
Index 4 CMDindex 8 matrix :  00122
Index 1 CMDindex 11 matrix :  01233
path0 = 4 :  0,1.0000    0,1.0000    1,1.0012    0,1.0024    0,1.0024
path1 = 1 :  0,1.0000    1,1.0012    1,1.0024    0,1.0036    1,1.0036
```

Candidate # 4

Index 4 CMDindex 8 matrix : 00122

Index 2 CMDindex 12 matrix : 01223

path0 = 4 : 0,1.0000 0,1.0000 1,1.0012 0,1.0024 0,1.0024

path1 = 2 : 0,1.0000 1,1.0012 0,1.0024 0,1.0024 0,1.0036

Candidate # 5

Index 5 CMDindex 9 matrix : 00112

Index 1 CMDindex 11 matrix : 01233

path0 = 5 : 0,1.0000 0,1.0000 0,1.0012 1,1.0012 1,1.0024

path1 = 1 : 0,1.0000 1,1.0012 1,1.0024 0,1.0036 1,1.0036

Candidate # 6

Index 5 CMDindex 9 matrix : 00112

Index 2 CMDindex 12 matrix : 01223

path0 = 5 : 0,1.0000 0,1.0000 0,1.0012 1,1.0012 1,1.0024

path1 = 2 : 0,1.0000 1,1.0012 0,1.0024 0,1.0024 0,1.0036

Candidate # 7

Index 5 CMDindex 9 matrix : 00112

Index 3 CMDindex 13 matrix : 01123

path0 = 5 : 0,1.0000 0,1.0000 0,1.0012 1,1.0012 1,1.0024

path1 = 3 : 0,1.0000 0,1.0012 0,1.0012 1,1.0024 0,1.0036

Candidate # 8

Index 6 CMDindex 10 matrix : 00012

Index 1 CMDindex 11 matrix : 01233

path0 = 6 : 0,1.0000 1,1.0000 0,1.0000 1,1.0012 1,1.0024

path1 = 1 : 0,1.0000 1,1.0012 1,1.0024 0,1.0036 1,1.0036

Table 14: Controller's connections list after reserving paths for S_0^{in} in f_0^{in} for both spaces

ID#	X_i^{in}	X_j^{out}	F_i^{in}	F_j^{out}	S_i^{in}	S_J^{out}	δ	path Index
0	0	0	0	1	0	0	2	1
1	1	1	0	1	0	1	3	4

```
----------------------
Candidate # 9
Index 6 CMDindex 10 matrix :  00012
Index 2 CMDindex 12 matrix :  01223
path0 = 6 :  0,1.0000    1,1.0000    0,1.0000    1,1.0012    1,1.0024
path1 = 2 :  0,1.0000    1,1.0012    0,1.0024    0,1.0024    0,1.0036
----------------------
Candidate # 10
Index 6 CMDindex 10 matrix :  00012
Index 3 CMDindex 13 matrix :  01123
path0 = 6 :  0,1.0000    1,1.0000    0,1.0000    1,1.0012    1,1.0024
path1 = 3 :  0,1.0000    0,1.0012    0,1.0012    1,1.0024    0,1.0036
======== END candidates ============
```

After reserving the paths for the new incoming connections, the controller updates the connections list as shown in Table 8.3 below.

During the guard-time period prior to S_1^{in}, the controller repeats the entire process for finding a path for S_0^{in} with additional steps. In this example, the output spaces, timeslots, frames, and delay is read from the SWA matrix and presented below:

swa_0, X_0^{in}, X_0^{out}, S_1^{in}, S_1^{out}, f_0^{in}, f_1^{out}, $\delta = 2$

swa_3, X_1^{in}, X_1^{out}, S_1^{in}, S_0^{out}, f_0^{in}, f_1^{out}, $\delta = 1$

Because there exists a connection in the network for the previous timeslot S_0^{in}, the con-

troller verifies that the candidate paths do not conflict with the existing connection. The first candidate path (below) shows that the third switch must be scheduled to be in BAR state at time $t = 1.0024$, while the same switch is previously scheduled to be in CROSS state. In this case, the controller will skip this candidate's set of paths and move to the next candidate set of paths. This additional step of verifying the switching state with previously scheduled paths did not take place at the first timeslot because the assumption of this simulation is to start at an empty switching fabric.

The following list of path candidates shows that only the third candidate's paths (marked in rectangle box) are nonblocking paths. The candidate's path do not block with each other or with previously scheduled paths.

```
======== Candidate Paths ============
Candidate # 1
Index 1 CMDindex 5 matrix :  01222
Index 2 CMDindex 2 matrix :  00111
path0 = 1 :  1,1.0012    1,1.0024    0,1.0036    1,1.0036    0,1.0036
path1 = 2 :  1,1.0012    0,1.0012    0,1.0024    1,1.0024    1,1.0024
----------------------
Candidate # 2
Index 1 CMDindex 5 matrix :  01222
Index 3 CMDindex 3 matrix :  00011
path0 = 1 :  1,1.0012    1,1.0024    0,1.0036    1,1.0036    0,1.0036
path1 = 3 :  1,1.0012    1,1.0012    0,1.0012    0,1.0024    1,1.0024
----------------------
Candidate # 3
Index 1 CMDindex 5 matrix :  01222
Index 4 CMDindex 4 matrix :  00001
path0 = 1 :  1,1.0012    1,1.0024    0,1.0036    1,1.0036    0,1.0036
path1 = 4 :  1,1.0012    1,1.0012    1,1.0012    0,1.0012    0,1.0024
----------------------
Candidate # 4
```

Index 2 CMDindex 6 matrix : 01122

Index 3 CMDindex 3 matrix : 00011

path0 = 2 : 1,1.0012 0,1.0024 0,1.0024 0,1.0036 0,1.0036

path1 = 3 : 1,1.0012 1,1.0012 0,1.0012 0,1.0024 1,1.0024

Candidate # 5

Index 2 CMDindex 6 matrix : 01122

Index 4 CMDindex 4 matrix : 00001

path0 = 2 : 1,1.0012 0,1.0024 0,1.0024 0,1.0036 0,1.0036

path1 = 4 : 1,1.0012 1,1.0012 1,1.0012 0,1.0012 0,1.0024

Candidate # 6

Index 3 CMDindex 7 matrix : 01112

Index 4 CMDindex 4 matrix : 00001

path0 = 3 : 1,1.0012 0,1.0024 1,1.0024 1,1.0024 1,1.0036

path1 = 4 : 1,1.0012 1,1.0012 1,1.0012 0,1.0012 0,1.0024

Candidate # 7

Index 4 CMDindex 8 matrix : 00122

Index 1 CMDindex 1 matrix : 01111

path0 = 4 : 0,1.0012 0,1.0012 1,1.0024 0,1.0036 0,1.0036

path1 = 1 : 0,1.0012 0,1.0024 1,1.0024 1,1.0024 1,1.0024

Candidate # 8

Index 6 CMDindex 10 matrix : 00012

Index 1 CMDindex 1 matrix : 01111

path0 = 6 : 0,1.0012 1,1.0012 0,1.0012 1,1.0024 1,1.0036

path1 = 1 : 0,1.0012 0,1.0024 1,1.0024 1,1.0024 1,1.0024

======== END of Candidates ============

Table 15: Controller's connections list after reserving paths for S_1^{in} in f_0^{in} for both spaces

ID#	X_i^{in}	X_j^{out}	F_i^{in}	F_j^{out}	S_i^{in}	S_J^{out}	δ	path Index
0	0	0	0	1	0	0	2	1
1	1	1	0	1	0	1	3	4
2	0	0	0	1	1	1	2	1
3	1	1	0	1	1	0	1	4

After reserving the paths for the new incoming connections, the controller updates the connections list as shown in Table 8.3 below.

At time $t = t_0 + 2(t_{GT} + t_{dur})$, it is the arrival time for the first guard-time period on the second frame prior to S_0^{in}. During this guard-time period, the first timeslot on the first frame is either outside the fabric or about to leave the switching fabric. During this guard-time, the controller will try to find a path for the incoming timeslot by repeating the same process that takes place during the guard-time period, as shown below:

swa_1, X_0^{in}, X_0^{out}, S_0^{in}, S_1^{out}, f_1^{in}, f_2^{out}, $\delta = 3$

swa_3, X_1^{in}, X_1^{out}, S_0^{in}, S_1^{out}, f_1^{in}, f_2^{out}, $\delta = 3$

Fortunately, there are only two candidate paths that do not block each other. Unfortunately, they block with the previously scheduled connections, as shown below:

```
======== ALL candidates ============

Candidate # 1
Index 1 CMDindex 11 matrix :  01233
Index 4 CMDindex 14 matrix :  00123
path0 = 1 :  1,1.0024    1,1.0036    1,1.0048    0,1.006    0,1.006
path1 = 4 :  1,1.0024    0,1.0024    1,1.0036    1,1.0048    0,1.006

----------------------

Candidate # 2
```

Table 16: Controller's connections list after reserving paths for S_0^{in} in f_1^{in} for both spaces and rearranging the existing connections ID# 2 and ID# 3

ID#	X_i^{in}	X_j^{out}	F_i^{in}	F_j^{out}	S_i^{in}	S_J^{out}	δ	path Index
0	0	0	0	1	0	0	2	1
1	1	1	0	1	0	1	3	4
2	0	0	0	1	1	1	2	2
3	1	1	0	1	1	0	1	2
4	0	0	1	2	0	1	3	1
5	1	1	1	2	0	1	3	4

```
Index 4 CMDindex 14 matrix :  00123
Index 1 CMDindex 11 matrix :  01233
path0 = 4 :  0,1.0024    0,1.0024    1,1.0036    1,1.0048    1,1.006
path1 = 1 :  0,1.0024    1,1.0036    1,1.0048    0,1.006    1,1.006
======== END ALL candidates ============
```

To fit the new incoming timeslot, existing connections must be rearranged. The first candidate's paths above can be used for the new incoming timeslot, if the existing connection for S_1^{in} in f_0^{in} is rearranged to the new set of paths listed below:

```
Index 2 CMDindex 6 matrix :  01122
Index 2 CMDindex 2 matrix :  00111
path0 = 2 :  1,1.0012    0,1.0024    0,1.0024    0,1.0036    0,1.0036
path1 = 2 :  1,1.0012    0,1.0012    0,1.0024    1,1.0024    1,1.0024
```

Hence, the blocking at the second and third stages does not exist between the incoming timeslots from both space inputs and the existing connections. The connections list after rearranging the existing connections, and adding the new connection, is shown in Table 8.3.

8.4 DISCUSSION

All blocking cases are presented in Table 17. The common factor between all blocking cases is that the new incoming timeslot S_0^{in} in frame F_1^{in} always requires the maximum delay $\delta = 3$ for both input spaces. However, there is not a clear pattern to generalize the blocking cases. All blocking cases are tested (manually), and adding new connections is achievable when rearranging the existing connections in the switching fabric.

Table 17: Blocking cases for space/time switching fabric

	Space 0				**Space 1**			
	Frame 0		Frame1		Frame 0		Frame 1	
S_i^{in}	0	1	0	1	0	1	0	1
swa_{index}	0		1		2		3	
S_i^{out}	0	1	1	0	0	1	1	0
δ	2	2	3	1	2	2	3	1
swa_{index}	0		1		3		3	
S_i^{out}	0	1	1	0	1	0	1	0
δ	2	2	3	1	3	1	3	1
swa_{index}	0		3		2		1	
S_i^{out}	0	1	1	0	0	1	1	0
δ	2	2	3	1	2	2	3	1
swa_{index}	0		3		3		1	
S_i^{out}	0	1	1	0	1	0	1	0
δ	2	2	3	1	3	1	3	1
swa_{index}	2		1		0		3	
S_i^{out}	0	1	1	0	0	1	1	0
δ	2	2	3	1	2	2	3	1
swa_{index}	2		1		1		3	
S_i^{out}	0	1	1	0	1	0	1	0
δ	2	2	3	1	3	1	3	1
swa_{index}	2		3		0		1	
S_i^{out}	0	1	1	0	0	1	1	0
δ	2	2	3	1	2	2	3	1
swa_{index}	2		3		1		1	
S_i^{out}	0	1	1	0	1	0	1	0
δ	2	2	3	1	3	1	3	1

9.0 CONCLUSIONS

In this dissertation, two control algorithms for two switching fabrics are proposed. The first switching fabric is a Timeslot Interchanger (TSI), which is a single space input/output fabric that interchanges timeslots with frame integrity. The control algorithm operates the switching fabric to perform the interchanging process without blocking. In addition to the control algorithm, this dissertation proposes two path assignment algorithms for TSI. The path assignments algorithms are named Dilated Path Assignment Algorithm (DPAA) and Economic Path Assignment Algorithm (EPAA). Each path assignment algorithm is used to satisfy different requirement such that: when high crosstalk hardware is used to build the switching fabric, DPAA is the ideal path assignment algorithm to use in the fabric at the expense of extra hardware devices. On the other hand, when sophisticated low crosstalk hardware is used, EPAA algorithm is the ideal path assignment algorithm to use for the TSI at the expense of the high cost of low crosstalk hardware components.

The second switching fabric is a space/time switching fabric with frame integrity. For this switching fabric, a control algorithm is implemented as well as a path assignment algorithm is also implemented. The control algorithms provide nonblocking space and time switching fabric with frame integrity.

Both fabrics can be used in transport networks or Information and Communication Technology (ICT) network. They provide All-Optical Network (AON) which should satisfy the three motivation stated at the beginning of this dissertation. As the control algorithm became available for both switching fabrics, ICT can now replace their old electronic core network with all-optical circuit switched network. Such network should enhance the speed, increase the bandwidth and reduce the power consumption.

10.0 FUTURE WORK

Because these algorithms are the first of its kind to operate the previously proposed switching fabrics, there is a room for improvement that I have noticed including but not limited to:

- **Improve complexity:** All control algorithms in this dissertation heavily depend on matrices. Most of these matrices are two-dimensional matrices. The number of loops used in all algorithms is high. Because the purpose for this work is to operate the switching fabric, the algorithms complexity was not considered. In the near future, the number of loops must be reduced to improve the complexity.
- **Find a generalized formula:** Although, the number of parallel Delay Element (DE) in the TSI is reduced, the exact (or minimum) number of required DEs is still undetermined. In the near future, a general formula for the required number of DEs will be discovered.
- **Switch timeslots separately in space domain:** In the space/time switching fabric, the control algorithm is implemented in the way that the entire frame must exit a single space output port. In the near future, the algorithm will be modified to allow every timeslot to be switched independently in space domain. This practice will allow more flexibility on switching channels in space and time domain.
- **Examine the blocking cases and identify their characteristic:** section 8.4 states eight cases where the space/time switching fabric requires rearranging the existing connections in order to accommodate the new incoming connection. These cases where manually verified. In the future, these cases can be automatically identified and the switching rearrange itself automatically. In addition, the space/time switching fabric with the proposed control algorithm can be tested on a large scale network.

APPENDIX A

SWA APPENDIX

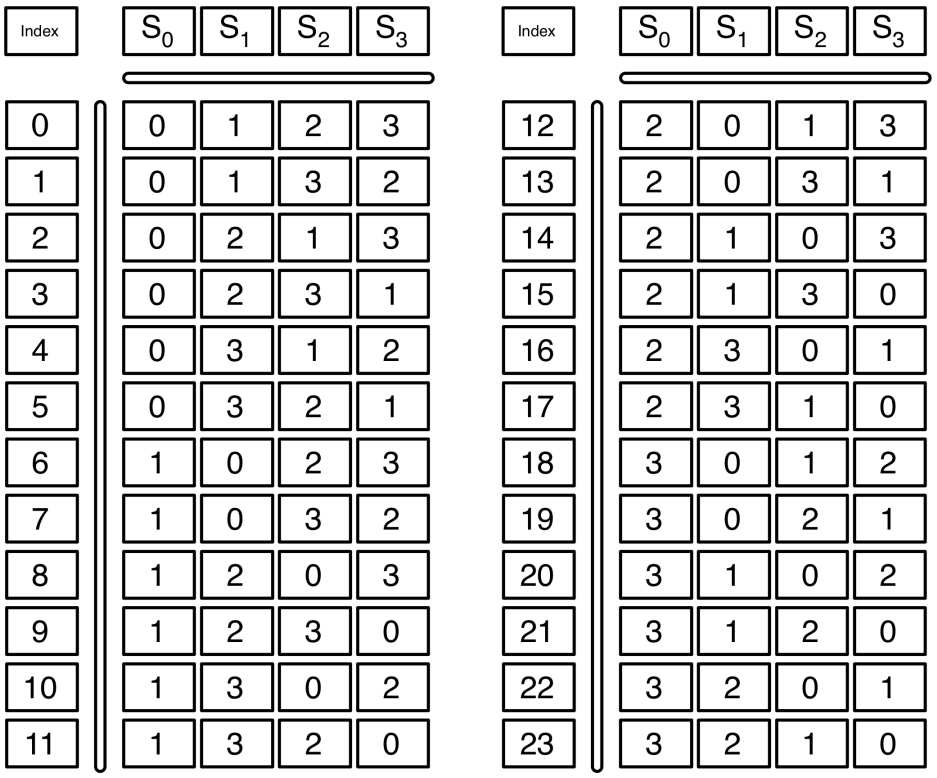


Figure 73: A complete SWA for S = 4

APPENDIX B

DWDM APPENDIX

Table 18: ITU DWDM Grid for C-Band on 100 GHz Spacing

From 1530 to 1570 nm					
No.	f_C (THz)	λ (nm)	No.	f_C (THz)	λ (nm)
1	191.00	1569.59	26	193.50	1549.32
2	191.10	1568.77	27	193.60	1548.51
3	191.20	1567.95	28	193.70	1547.72
4	191.30	1567.13	29	193.80	1546.92
5	191.40	1566.31	30	193.90	1546.12
6	191.50	1565.5	31	194.00	1545.32
7	191.60	1564.68	32	194.10	1544.53
8	191.70	1563.86	33	194.20	1543.73
9	191.80	1563.05	34	194.30	1542.94
10	191.90	1562.23	35	194.40	1542.14
11	192.00	1561.42	36	194.50	1541.35
12	192.10	1560.61	37	194.60	1540.56
13	192.20	1559.79	38	194.70	1539.77
14	192.30	1558.98	39	194.80	1538.98
15	192.40	1558.17	40	194.90	1538.19
16	192.50	1557.36	41	195.00	1537.4
17	192.60	1556.55	42	195.10	1536.61
18	192.70	1555.75	43	195.20	1535.82
19	192.80	1554.94	44	195.30	1535.04
20	192.90	1554.13	45	195.40	1534.25
21	193.00	1553.33	46	195.50	1533.47
22	193.10	1552.52	47	195.60	1532.68
23	193.20	1551.72	48	195.70	1531.9
24	193.30	1550.92	49	195.80	1531.12
25	193.40	1550.12	50	195.90	1530.33

APPENDIX C

SWC APPENDIX

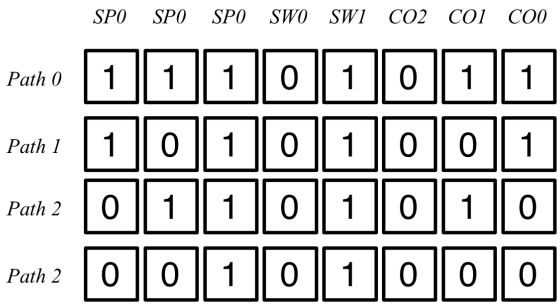


Figure 74: A complete SWC matix for $S = 4$ and $\delta = 1$

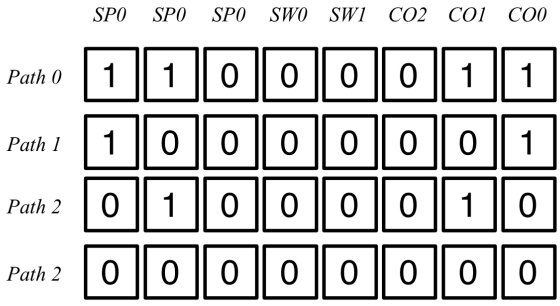


Figure 75: A complete SWC matix for $S = 4$ and $\delta = 2$

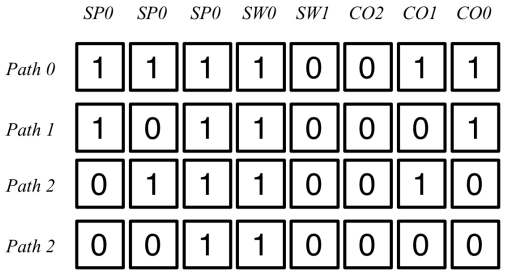


Figure 76: A complete SWC matix for S = 4 and $\delta = 3$

	SP0	SP0	SP0	SW0	SW1	CO2	CO1	CO0
Path 0	1	1	0	1	0	1	1	1
Path 1	1	0	0	1	0	1	0	1
Path 2	0	1	0	1	0	1	1	0
Path 2	0	0	0	1	0	1	0	0

Figure 77: A complete SWC matix for S = 4 and $\delta = 4$

	SP0	SP0	SP0	SW0	SW1	CO2	CO1	CO0
Path 0	1	1	1	0	0	1	1	1
Path 1	1	0	1	0	0	1	0	1
Path 2	0	1	1	0	0	1	1	0
Path 2	0	0	1	0	0	1	0	0

Figure 78: A complete SWC matix for S = 4 and $\delta = 5$

	SP0	SP0	SP0	SW0	SW1	CO2	CO1	CO0
Path 0	1	1	0	0	1	1	1	1
Path 1	1	0	0	0	1	1	0	1
Path 2	0	1	0	0	1	1	1	0
Path 2	0	0	0	0	1	1	0	0

Figure 79: A complete SWC matix for S = 4 and $\delta = 6$

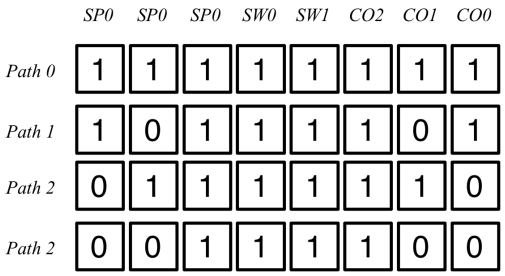


Figure 80: A complete SWC matix for S = 4 and $\delta = 7$

APPENDIX D

SPACE/TIME CUMULATIVE DELAY MATRIX

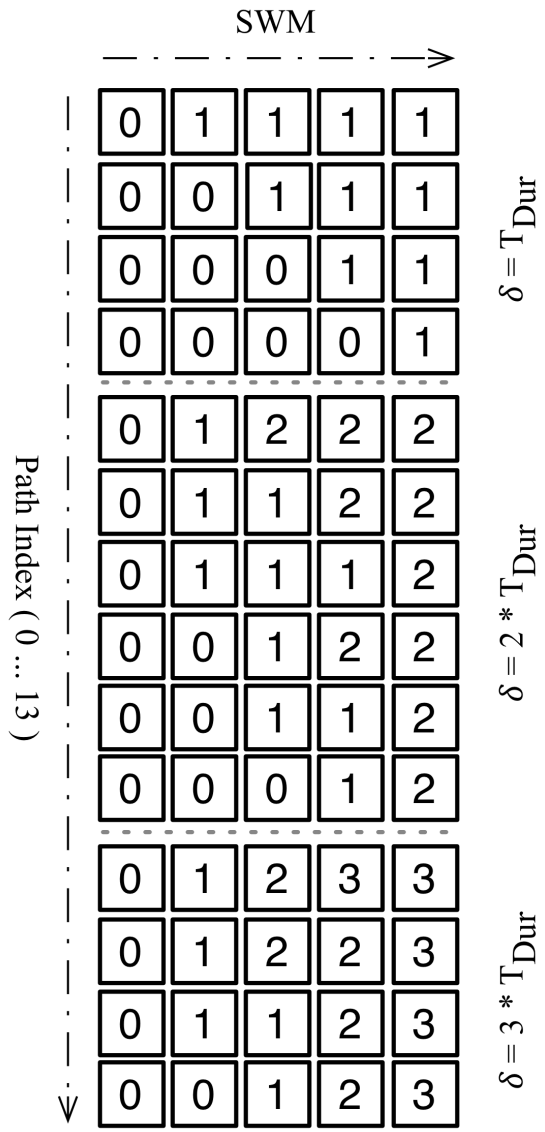


Figure 81: A complete cumulative delay matrix for a <2,2> space/time switching fabric

APPENDIX E

SPACE/TIME SWITCHING CONTROL MATRICES

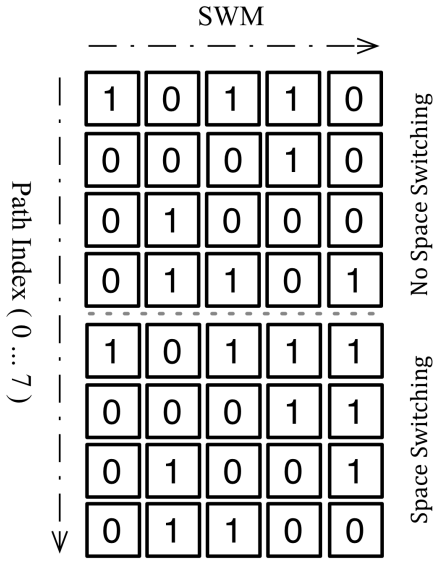


Figure 82: SWC matrix for X_0 and $\delta = 1$

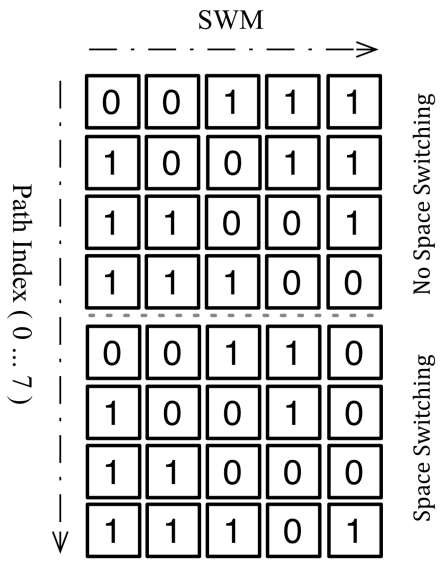


Figure 83: SWC matrix for X_1 and $\delta = 1$

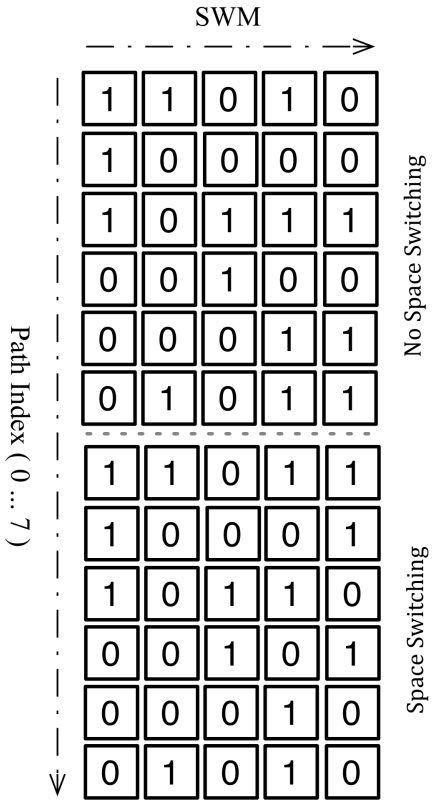


Figure 84: SWC matrix for X_0 and $\delta = 2$

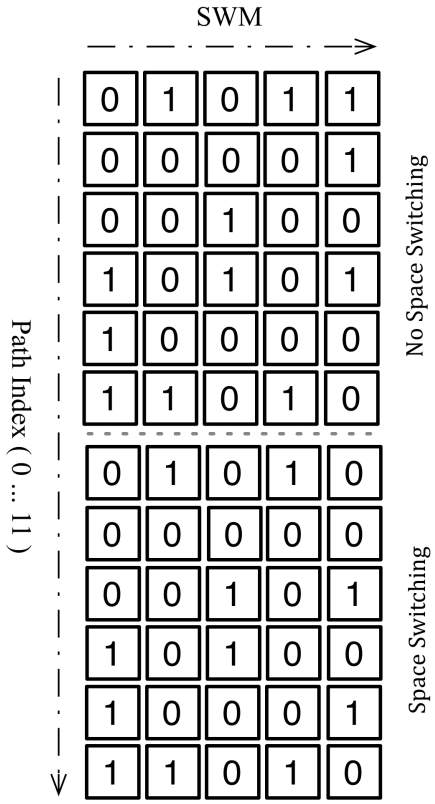


Figure 85: SWC matrix for X_1 and $\delta = 2$

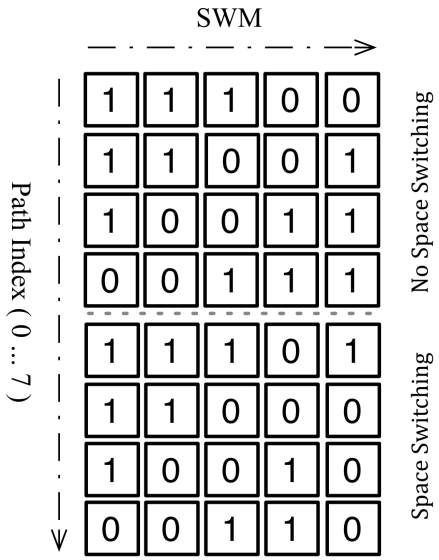


Figure 86: SWC matrix for X_0 and $\delta = 3$

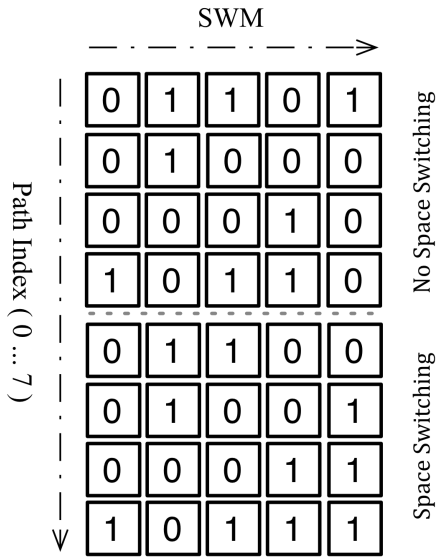


Figure 87: SWC matrix for X_1 and $\delta = 3$

BIBLIOGRAPHY

[1] "Sustaining the cloud with a faster, greener and uptime optimised data center," Corning, NY, Tech. Rep., 2012.

[2] Z. Pan, J. Cao, Y. Bansal, V. K. Tsui, S. K. Fong, Y. Zhang, J. Taylor, H. Lee, M. Jeon, V. Akella *et al.*, "All-optical programmable time-slot-interchanger using optical-label switching with tunable wavelength conversion and n by n arrayed waveguide grating routers," in *Optical Fiber Communication Conference (OFC)*, vol. 70, 2002, pp. 267–268.

[3] D. K. Hunter and D. G. Smith, "An architecture for frame integrity optical tdm switching," *Lightwave Technology, Journal of*, vol. 11, no. 5, pp. 914–924, 1993.

[4] H. J. Chao and S. Y. Liew, "A new optical cell switching paradigm," in *Int. Workshop Optical Burst Switching, Dallas, TX*, 2003.

[5] S. V. Ramanan, H. F. Jordan, and J. R. Sauer, "A new time domain, multistage permutation algorithm [switching systems]," *Information Theory, IEEE Transactions on*, vol. 36, no. 1, pp. 171–173, 1990.

[6] C. V. Networking, "white paper: The zettabyte era–trends and analysis," Tech. Rep., 2014.

[7] E. Adler, "Here's why 'the internet of things' will be huge, and drive tremendous value for people and businesses," December 2013, visited on November 18,2015. [Online]. Available: http://www.businessinsider.com/growth-in-the-internet-of-things-2013-10

[8] I. T. U. (ITU), "The state of broadband 2014: Broadband for all," Tech. Rep., 2014.

[9] I. Qualcomm Technologies, "white paper: 1000x: Higher efficiency," Tech. Rep., 2014.

[10] T. Team, "Growth in data traffic will positively drive corning's optical communications segment," March 2014, visited on November 11,2015. [Online]. Available: http://www.forbes.com/sites/greatspeculations/2014/03/31/growth-in-data-traffic-will-positively-drive-cornings-optical-communications-segment/

[11] B. Labs, "White paper : Video shakes up the ip edge," Tech. Rep., 2012.

[12] C. Qiao and R. Melhem, "Reducing communication latency with path multiplexing in optically interconnected multiprocessor systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 8, no. 2, pp. 97–108, 1997.

[13] D. Reisinger, "Roku still tops as sales of streaming-media players rise," December 2014, visited on November 18,2015. [Online]. Available: http://www.cnet.com/news/more-households-buying-streaming-media-players-roku-still-tops/

[14] M. Hoelzel, "4k tv prices are falling fast, fueling rapid adoption of the new video standard," July 2014, visited on November 18,2015. [Online]. Available: http://www.businessinsider.com/4k-tv-prices-are-falling-fast-fueling-rapid-adoption-of-the-new-video-standard-2014-6

[15] K. Inagaki and J. Cheng, "Inexpensive ultra high-definition tvs on horizon," January 2014, visited on November 18,2015. [Online]. Available: http://www.wsj.com/articles/SB10001424052702304361604579292630876600834

[16] M. Hoelzel, "4k tv shipments are ramping up much faster than hd tv did in the past," October 2014, visited on November 18,2015. [Online]. Available: http://www.businessinsider.com/4k-tv-shipments-growth-2014-9

[17] "Report and order: Protecting and promoting the open internet," Federal Communications Commission, Tech. Rep., 2015, order number: FCC 15-24.

[18] S. Ramachandran, "Netflix to pay comcast for smoother streaming," February 2014, visited on November 18,2015. [Online]. Available: http://www.wsj.com/articles/SB10001424052702304834704579401071892041790

[19] C. F. Lam, H. Liu, B. Koley, X. Zhao, V. Kamalov, and V. Gill, "Fiber optic communication technologies: What's needed for datacenter network operations," *IEEE Communications Magazine*, vol. 48, no. 7, pp. 32–39, 2010.

[20] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[21] "Vision and road map: Routing telecom and data centers towards efficient energy use," Tech. Rep., 2009.

[22] Y. Zhang, P. Chowdhury, M. Tornatore, and B. Mukherjee, "Energy efficiency in telecom optical networks," *Communications Surveys & Tutorials, IEEE*, vol. 12, no. 4, pp. 441–458, 2010.

[23] R. S. Tucker, "Green optical communicationsâĂŤpart ii: Energy limitations in networks," *Selected Topics in Quantum Electronics, IEEE Journal of*, vol. 17, no. 2, pp. 261–274, 2011.

[24] M. I. Green, "Cloud computing and its contribution to climate change," *Greenpeace International*, 2010.

[25] C. Lange, D. Kosiankowski, R. Weidmann, and A. Gladisch, "Energy consumption of telecommunication networks and related improvement options," *Selected Topics in Quantum Electronics, IEEE Journal of*, vol. 17, no. 2, pp. 285–295, 2011.

[26] M. Jimeno, K. Christensen, and B. Nordman, "A network connection proxy to enable hosts to sleep and save energy," in *Performance, Computing and Communications Conference, 2008. IPCCC 2008. IEEE International*. IEEE, 2008, pp. 101–110.

[27] R. S. Tucker, "Green optical communicationsâĂŤpart i: Energy limitations in transport," *Selected Topics in Quantum Electronics, IEEE Journal of*, vol. 17, no. 2, pp. 245–260, 2011.

[28] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures," *Communications Surveys & Tutorials, IEEE*, vol. 13, no. 2, pp. 223–244, 2011.

[29] S. Ramamurthy and B. Mukherjee, "Survivable wdm mesh networks. part i-protection," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 1999, pp. 744–751.

[30] D. Richardson, J. Fini, and L. Nelson, "Space-division multiplexing in optical fibres," *Nature Photonics*, vol. 7, no. 5, pp. 354–362, 2013.

[31] B. Wen, R. Shenai, and K. Sivalingam, "Routing, wavelength and time-slot-assignment algorithms for wavelength-routed optical wdm/tdm networks," *Journal of Lightwave Technology*, vol. 23, no. 9, p. 2598, 2005.

[32] D. K. Hunter, D. Cotter, R. B. Ahmad, W. D. Cornwell, T. H. Gilfedder, P. J. Legg, and I. Andonovic, "2× 2 buffered switch fabrics for traffic routing, merging, and shaping in photonic cell networks," *Lightwave Technology, Journal of*, vol. 15, no. 1, pp. 86–101, 1997.

[33] C. Qiao and M. Yoo, "Optical burst switching (obs)-a new paradigm for an optical internet," *Journal of high speed networks*, vol. 8, no. 1, p. 69, 1999.

[34] X. Chen, B. Guo, S. Ma, P. Zhang, J. Li, Z. Chen, and Y. He, "A novel centralized scheduling for time-slotted optical burst switching in elastic optical network," in *Photonic Networks and Devices*. Optical Society of America, 2013, pp. NW1C–3.

[35] T. S. El-Bawab, *Optical switching*. Springer Science & Business Media, 2008.

[36] G. I. Papadimitriou, C. Papazoglou, A. S. Pomportsis *et al.*, "Optical switching: switch fabrics, techniques, and architectures," *Journal of lightwave technology*, vol. 21, no. 2, p. 384, 2003.

[37] R. Yadav and R. R. Aggarwal, "Survey and comparison of optical switch fabrication techniques and architectures," *arXiv preprint arXiv:1004.4481*, 2010.

[38] S. Yao, B. Mukherjee, and S. Dixit, "Advances in photonic packet switching: an overview," *Communications Magazine, IEEE*, vol. 38, no. 2, pp. 84–94, 2000.

[39] D. K. Hunter, W. D. Cornwell, T. H. Gilfedder, A. Franzen, and I. Andonovic, "Slob: A switch with large optical buffers for packet switching," *Journal of Lightwave Technology*, vol. 16, no. 10, p. 1725, 1998.

[40] M. Chia, D. Hunter, I. Andonovic, P. Ball, I. Wright, S. Ferguson, K. Guild, and M. O'Mahony, "Packet loss and delay performance of feedback and feed-forward arrayed-waveguide gratings-based optical packet switches with wdm inputs-outputs," *Journal of lightwave technology*, vol. 19, no. 9, p. 1241, 2001.

[41] R. Thompson *et al.*, "Architectures with improved signal-to-noise ratio in photonic systems with fiber-loop delay lines," *Selected Areas in Communications, IEEE Journal on*, vol. 6, no. 7, pp. 1096–1106, 1988.

[42] R. Ramaswami, K. Sivarajan, and G. Sasaki, *Optical networks: a practical perspective*. Morgan Kaufmann, 2009.

[43] K. Venugopal, M. S. Kumar, and P. S. Kumar, "A heuristic for placement of limited range wavelength converters in all-optical networks," *Computer Networks*, vol. 35, no. 2âĂŞ3, pp. 143 – 163, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128600001456

[44] M. Sivakumar and S. Subramahiam, "Performance evaluation of time switching in tdm wavelength routing networks," in *Broadband Networks, 2004. BroadNets 2004. Proceedings. First International Conference on*. IEEE, 2004, pp. 212–221.

[45] W. Kabacinski, *Nonblocking electronic and photonic switching fabrics*. Springer Science & Business Media, 2005.

[46] R. A. Thompson, *Telephone switching systems*. Artech House, 2000.

[47] C. Clos, "A study of non-blocking switching networks," *Bell System Technical Journal*, vol. 32, no. 2, pp. 406–424, 1953.

[48] V. E. Benes, *Mathematical theory of connecting networks and telephone traffic*. Elsevier, 1965.

[49] R. Thompson, P. P. Giordano *et al.*, "An experimental photonic time-slot interchanger using optical fibers as reentrant delay-line memories," *Lightwave Technology, Journal of*, vol. 5, no. 1, pp. 154–162, 1987.

[50] H. F. Jordan, D. Lee, K. Y. Lee, and S. V. Ramanan, "Serial array time slot interchangers and optical implementations," *Computers, IEEE Transactions on*, vol. 43, no. 11, pp. 1309–1318, 1994.

[51] R. Srinivasan and A. K. Somani, "A generalized framework for analyzing time-space switched optical networks," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 1, pp. 202–215, 2002.

[52] J. Yates, J. Lacey, and D. Everitt, "Blocking in multiwavelength tdm networks," *Telecommunication Systems*, vol. 12, no. 1, pp. 1–19, 1999.

[53] L. Li, S. D. Scott, and J. S. Deogun, "A novel fiber delay line buffering architecture for optical packet switching," in *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, vol. 5. IEEE, 2003, pp. 2809–2813.

[54] M. Kondo, K. Komatsu, Y. Ohta, S. Suzuki, K. Nagashima, and H. Goto, "High-speed optical time switch with integrated optical 1× 4 switches and single-polarization fiber delay lines," in *4th international conference on integrated optics and optical fiber communication*, 1983, p. 04967.

[55] S. Y. Liew, G. Hu, and H. J. Chao, "Scheduling algorithms for shared fiber-delay-line optical packet switches-part i: The single-stage case," *Journal of lightwave technology*, vol. 23, no. 4, p. 1586, 2005.

[56] O. Zouraraki, K. Yiannopoulos, P. Zakynthinos, D. Petrantonakis, E. Varvarigos, A. Poustie, G. Maxwell, and H. Avramopoulos, "Implementation of an all-optical time-slot-interchanger architecture," *IEEE Photonics Technology Letters*, vol. 19, no. 17/20, p. 1307, 2007.

[57] L. E. Hasnawi and R. A. Thompson, "An improved control algorithm for a class of photonics timeslot interchanger," *ICNS 2015*, p. 64, 2015.

[58] R. Barry, P. Humblet *et al.*, "Models of blocking probability in all-optical networks with and without wavelength changers," in *INFOCOM'95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People. Proceedings. IEEE.* IEEE, 1995, pp. 402–412.

[59] H. Chao, L. Wu, Z. Zhang, S. Yang, L. Wang, Y. Chai, J. Fan, and F. Choa, "A photonic front-end processor in a wdm atm multicast switch," *Journal of lightwave technology*, vol. 18, no. 3, p. 273, 2000.

[60] K. Onohara, H. Sotobayashi, K.-i. Kitayama, and W. Chujo, "Photonic time-slot and wavelength-grid interchange for 10-gb/s packet switching," *IEEE Photonics Technology Letters*, vol. 13, no. 10, pp. 1121–1123, 2001.

[61] C. Qiao, "Analysis of space-time tradeoffs in photonic switching networks," in *INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, vol. 2. IEEE, 1996, pp. 822–829.

[62] J. Späth, J. Charzinski, S. Hörz, and M. N. Huber, "Performance analysis of a combined wdm/tdm network based on fixed wavelength assignment," in *Optical Network Design and Modelling*. Springer, 1998, pp. 147–159.

[63] D. Opferman and N. Tsao-Wu, "On a class of rearrangeable switching networks part i: Control algorithm," *Bell System Technical Journal*, vol. 50, no. 5, pp. 1579–1600, 1971.

[64] A. Chakrabarty and M. Collier, "routing algorithm for (2logn- 1)-stage switching networks and beyond," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 3045–3055, 2014.

[65] L. E. Hasnawi and R. A. Thompson, "Photonic timeslot interchangers with a reduced number of feed-forward fiber delay lines," *Procedia Computer Science*, vol. 34, pp. 47–54, 2014.

[66] D. Hunter and D. Smith, "Optical tdm switching architectures with reduced control complexity," *IEE Proceedings I (Communications, Speech and Vision)*, vol. 140, no. 3, pp. 220–228, 1993.

[67] M. ZDEBLLCK, "Design variables prevent a single industry standard," *Laser focus world*, vol. 37, no. 3, pp. 139–144, 2001.

[68] B. Ramamurthy and B. Mukherjee, "Wavelength conversion in wdm networking," 1998.

[69] R. Thompson, "Optimizing photonic variable-integer-delay circuits," *Springer Series in Electronics and Photonics 25*, p. 158, 1987.

[70] X. Yuan, R. Gupta, and R. Melhem, "Does time-division multiplexing close the gap between memory and optical communication speeds?" in *Parallel Computer Routing and Communication*. Springer, 1998, pp. 261–271.