Building Dynamic and Energy-Efficient Datacenters with Passive Optical Devices

by

Pei Jin

A thesis submitted to the Faculty of Graduate and Postdoctoral
Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Carleton University

Ottawa, Ontario

© 2016, Pei Jin

Abstract

Most datacenter network designs overwhelmingly use expensive and power-consuming electronic switches or expensive active optical switches with long reconfiguration times. In this thesis, we explore architectural solutions to leverage the design elements of Passive Optical Cross-Connection Networks with Multiple Planes (POXN/MPs) and Passive Optical Cross-Connection Networks with Multiple Planes and Bundled Ports (POXN/MP-BPs), both of which consist primarily of passive optical fabrics and optical transceivers that replace groups of switches in either hierarchical or recursively defined networks. Through simple physical interconnections, our proposed architectures allow datacenters to incrementally scale up in network capacity and scale out in total number of racks. From developed formulas for calculating cost and power consumption, we demonstrate that POXN/MP-BPs can significantly reduce the cost and power consumption of datacenter networks compared to the traditional fat tree topology. To lower overhead and adapt to the types of real datacenter scenarios that are possible with POXN/MP-BPs, we propose the new Multiple Channels with Bundled Ports Distributed Access Protocol (MCBDAP), which outperforms the Multiple Channels Distributed Access Protocol (MCDAP) for POXN/MP in terms of bandwidth efficiency, especially for those applications involving higher proportions of inter-rack traffic than intra-rack traffic.

Acknowledgements

I would like to express my deepest thanks and sincere appreciation to my supervisor, Prof. Changcheng Huang, whose kind guidance, rich expertise, and comprehensive understanding make it possible for me to complete my research on this topic I was interested in. It is a great honor to work under his supervision.

I would like to express my deep gratitude to my parents and my wife for endless encouragement, gracious advice, and unconditional support throughout the study.

It is a great pleasure to acknowledge my thanks to Mr. Wenda Ni, Mr. Weiwei Li, and Mr. Yang An for being so generous to give their precious advice and kind help.

Table of Contents

Abstract.....	2
Acknowledgements	3
Table of Contents	4
List of Tables	8
List of Figures.....	9
List of Variables	13
List of Acronyms	15
Chapter 1 Introduction.....	17
1.1 Motivation.....	17
1.2 Problem Statement and Objective.....	18
1.3 Thesis Contribution.....	19
1.4 Thesis Structure	20
Chapter 2 Background Information	22
2.1 Application and Traffic Patterns in Datacenters	22
2.1.1 One-to-one traffic pattern	23
2.1.2 One-to-several traffic pattern	24
2.1.3 Several-to-one traffic pattern.....	24
2.1.4 All-to-all traffic pattern	24
2.2 Datacenter Network	25
2.2.1 Traditional datacenter network architectures	25
2.2.1.1 Fat tree.....	29

2.2.1.2	BCube.....	34
2.2.1.3	VL2.....	37
2.2.1.4	HyperX.....	38
2.2.1.5	DCell	40
2.2.1.6	Jellyfish	42
2.2.1.7	Summary of traditional datacenter architectures.....	42
2.2.2	Enabling optical interconnect technologies.....	43
2.2.2.1	Optical burst switch-based DCN.....	44
2.2.2.2	AWGR-based TONAK LION switch	49
2.2.2.3	Proteus.....	51
2.2.2.4	Quartz.....	53
2.2.2.5	Performance metrics.....	55
2.3	Passive Optical Device	56
2.3.1	POXN	56
2.3.2	POXN/MP	59
Chapter 3	POXN/MP-BP	62
3.1	Physical Transmission System.....	62
3.2	Power Saving Advantage.....	64
3.3	MCBDAP.....	68
3.3.1	Discovery phase	68
3.3.2	Data transfer phase	71

3.4	New Algorithm for the MCBDAp	73
3.4.1	Initial state	73
3.4.2	An example of transmission scheduling using the MCBDAp	76
3.4.3	Benefits and tradeoffs.....	79
Chapter 4 Integrating POXN/MPs and POXN/MP-BPs into Datacenters.....		83
4.1	POCube	83
4.1.1	POCube structure	84
4.1.2	Routing in POCube	85
4.1.3	Cost and performance.....	91
4.2	Sandwich Tree	93
4.2.1	POXN/MP-BPs as design elements to replace switches	95
4.2.2	Sandwich Tree Structure	98
4.2.3	Routing in Sandwich Tree	99
4.2.4	Benefits in terms of cost and power consumption	104
4.2.4.1	Cost advantages.....	104
4.2.4.2	Power consumption advantages	110
Chapter 5 Numerical Results.....		114
5.1	Algorithm-level Simulation	114
5.1.1	Different proportions of biased traffic.....	117
5.1.2	Different bundled port numbers for biased traffic.....	119
5.1.3	Mean of the unicast queue length for unbiased traffic	120
5.1.4	Variation of the unicast queue length for unbiased traffic	121
5.1.5	Different bundled port number for unbiased traffic	122

5.2	Device-level and Network-level Simulations	124
5.2.1	Device-level simulation.....	124
5.2.2	Network-level simulation	128
5.2.2.1	Two-level POCube modeling.....	129
5.2.2.2	Two-tier Sandwich Tree modeling.....	133
5.2.2.3	Three-tier Sandwich Tree modeling.....	142
5.2.3	Performance comparison	150
Chapter 6 Conclusion and Future Work		152
Appendix.....		154
References		155

List of Tables

Table 2-1. Traditional datacenter architecture comparison, where n represents the switch port count, k represents the recursive structure level, N represents the total number of servers, and Sk stands for the number of switches in a k -dimensional HyperX.....	43
Table 2-2. Optical architectures comparison	55
Table 3-1. Multicast traffic list obtained from CONFIRMATION message.....	70
Table 3-2. Unicast traffic list obtained from CONFIRMATION message	70
Table 4-1. Forwarding table for server 00	89
Table 4-2. The first solution is a $POCube_4$ built with $4*4$ POXN/MPs with five levels, and the second solution is a 2-level $POCube_1$ built with $32*32$ POXN/MPs.	92
Table 4-3. Cost comparison of the two solutions	93
Table 4-4. Power consumption comparison of the two solutions.....	93
Table 4-5. Comparison of main component quantities among three topologies	107
Table 4-6. Unit power consumption of each element.....	111
Table 5-1. Time periods for different kinds of overhead [9]	126
Table 5-2. Different structure with 16 servers	151

List of Figures

Figure 2-1. Traditional network architecture	27
Figure 2-2. An example of a scale-out datacenter architecture	28
Figure 2-3. Binary fat tree.....	30
Figure 2-4. Fat tree architecture with three levels of switches in four pods	31
Figure 2-5. Two-level table for edge switch 10.0.1.1 in pod 0.....	33
Figure 2-6. Two-level table for aggregation switch 10.0.2.1.....	33
Figure 2-7. Routing table for core switch 10.4.1.2	33
Figure 2-8. The BCube's recursive structure.....	36
Figure 2-9. HyperX with $L = 2$, $S_1 = 2$, $S_2 = 4$, $K = 1$, $T = 4$	39
Figure 2-10. A DCell _{<i>l</i>} network ($n = 4$) consisting of five DCell _{<i>0</i>} subnetworks	41
Figure 2-11. C-through architecture	46
Figure 2-12 Helios architecture	48
Figure 2-13. TONAK LION switch architecture.....	50
Figure 2-14. Sending and receiving infrastructure of Proteus	52
Figure 2-15. Quartz architecture	54
Figure 2-16. Three-port POXN physical layer	57
Figure 2-17. Interconnection within a 9*9 coupler fabric using Banyan topology	57
Figure 2-18. Physical interconnections of a four-port POXN/MP	59
Figure 2-19. Transmitters and receivers in two planes using different wavelengths.....	60
Figure 3-1. Physical interconnections of a three-port POXN/MP-BP	64
Figure 3-2. Compare power loss at a coupler output for POXN/MP and POXN/MP-BP with 2-in-	

1 bundled ports and 4-in-1 bundled ports	66
Figure 3-3. Compare power loss at a coupler output for POXN/MP and POXN/MP-BP with 2-in-1 bundled ports and 4-in-1 bundled ports	67
Figure 3-4. POXN/MP-BP control message format	69
Figure 3-5. A simple MCBDAP sample message sequence chart.....	72
Figure 3-6. Initial state of queue size in a three-port POXN/MP-BP	74
Figure 3-7. The transmission scheduling of a three-port POXN/MP without optimization	75
Figure 3-8. Transmission scheduling of a three-port POXN/MP-BP with a 2-in-1 bundled port	78
Figure 3-9. New LQF algorithm for POXN/MP-BPs.....	82
Figure 4-1. POCube structure	85
Figure 4-2. A POCube ₁ consists of four POCube ₀ s with 16 servers.....	86
Figure 4-3. POCube packet processing at the system level	87
Figure 4-4. POCube broadcast.....	90
Figure 4-5. One example of Google's datacenter architecture with five stages of switches	96
Figure 4-6. An example of an eight-pod Sandwich Tree with POXN/MP-BPs and POXN/MPs	99
Figure 4-7. Transponder numbers 0 to 23 for switch 10.0.0.1 in a POXN/MP-BP with eight pods	100
Figure 4-8. Two-level routing table for switch 10.0.0.1. It will deliver an incoming packet to a specific transponder depending on the destination IP address of the packet.	101
Figure 4-9. Sandwich Tree broadcast	103
Figure 4-10. Traditional fat-tree broadcast	104
Figure 4-11. Cost comparison among the traditional fat tree topology, the POXN/MP, and the POXN/MP-BP, with different numbers of bundled ports for each bundled group	108

Figure 4-12. Power consumption comparison of the traditional fat tree topology, the POXN/MP, and the POXN/MP-BP, with different numbers of bundled ports for each bundled group on the switch	112
Figure 5-1. The ratio of outgoing traffic to local traffic in 10 datacenters [50]	115
Figure 5-2. Different inter-rack/intra-rack proportions for biased traffic	118
Figure 5-3. Comparison of LQF and LQF/BP algorithm efficiencies for biased traffic with different numbers of bundled LQF/BP ports	120
Figure 5-4. Different mean queue lengths for unbiased traffic	121
Figure 5-5. Different standard deviations in queue length for unbiased traffic	122
Figure 5-6. Comparison of LQF and LQF/BP algorithm efficiencies for unbiased traffic with different numbers of bundled LQF/BP ports	123
Figure 5-7. Simulation topology of a six-port POXN/MP-BP	125
Figure 5-8. Throughput performance of the MCBDAF with an increase in load	127
Figure 5-9. Mean packet delays for the MCBDAF with different offered loads ρ	128
Figure 5-10. Simulation topology of a two-level POCube with 16 servers	130
Figure 5-11. Delay from source server to destination server going via one POXN/MP; delay from source server to destination server going via two POXN/MPs; and end-to-end delay for a POCube topology with different loads offered to the system	131
Figure 5-12. Simulation topology of POXN/MP-BP with eight servers	135
Figure 5-13. Simulation topology of a POXN/MP with eight servers	135
Figure 5-14. Delay from source server to destination server going via one POXN/MP-BP; delay from source server to destination server going via two POXN/MP-BPs; and end-to-end delay for a two-tier Sandwich Tree topology (shown in Figure 5-12) with different loads offered to the	

system	136
Figure 5-15. Delay from source server to destination server going via two POXN/MP-BPs include three main kinds of delays, which are server queueing delay, switch input port queueing delay, and switch output port queueing delay	138
Figure 5-16. All four kinds of link utilizations with different loads offered to each source server	140
Figure 5-17. Normalized aggregate throughput for a two-tier POXN/MP and a two-tier POXN/MP-BP	141
Figure 5-18. The three-tier simulation topology is a two-pod network composed of four POXN/MP-BPs, two POXN/MPs, and four switches	143
Figure 5-19. The three-tier simulation topology is a two-pod network composed of six POXN/MPs and eight switches	144
Figure 5-20. Delay from source server to destination server going via one first-level POXN/MP and two first-level POXN/MP-BPs; delay from source server to destination server going via two first-level POXN/MP-BPs and one top-level POXN/MP; and end-to-end delay in a three-tier Sandwich Tree topology with different loads offered to the system	145
Figure 5-21. Delay from the source server to the destination server via one first-level POXN/MP-BP and two first-level POXN/MP-BPs; delay from the source server to the destination server via two first-level POXN/MP-BPs and one top-level POXN/MP; and ETE delay with different loads under a biased traffic scenario with a proportion of 10:1	148
Figure 5-22. Comparison of the normalized aggregate throughputs for different proportions of inter-pod and intra-pod traffic for a combination of POXN/MPs and POXN/MP-BPs and POXN/MPs only in a three-tier Sandwich Tree structure	149

List of Variables

S_k – stands for the number of switches in k-dimensional HyperX

T_i^L – the loopback time for transmitter i

T^P – the corresponding one-way propagation delay

T^C – the constant time value for message processing

T^D – the transmission delay of the CONFIRMATION message

T_{ij}^Q – the transmission time for the queue to be sent from transmitter i to receiver j

$T_{ij \rightarrow ij'}^Q$ – the transmission time of the queue that originally designated from transmitter i to receiver j is now sent from transmitter i to receiver j' (j and j' are bundled in a single node)

i – the index of a transmitter

j – the index of a receiver

t_{ik}^{Mx} – the k -th Type x mismatch that transmitter i experiences

T^T – the constant tuning time for a tunable transponder

Q_{ij} – the queue to be sent from transmitter i to receiver j

$Q_{ij_{left}}$ – the queue to be sent from transmitter i to each remaining receiver j

T_i – the normal individual transmitter i

R_j – the normal individual receiver j

T_b – the special bundled transmitter b

R_b – the special bundled receiver b

t_i^{Idle} – the transmitter i 's idle time

t_{ij}^S – the starting time for the transmission from transmitter i to receiver j

t_{ij}^F – the completion time for the transmission from transmitter i to receiver j

B – the number of bundled ports

T_{ij}^{loc} – the completion time for intra-rack transmission from transmitter i to receiver j

T_{ij}^{ext} – the completion time for inter-rack transmission from transmitter i to receiver j

List of Acronyms

AWG	Arrayed-Wavelength Grating
AWGR	Arrayed-Wavelength Grating Router
CAPEX	Capital Expenditure
DCN	Datacenter Network
ECMP	Equal-Cost Multi-Path
HEDAP	High Efficiency Distributed Access Protocol
LQF/BP	Longest Queue First / Bundled Ports
LR	Long-Range
MCDAP	Multiple Channels Distributed Access Protocol
MCBDAP	Multiple Channels with Bundled Ports Distributed Access Protocol
MEMS	Micro Electro Mechanical System
NIC	Network Interface Card
POCube	Passive Optical Cube
POXN	Passive Optical Cross-connection Network
POXN/MP	Passive Optical Cross-connection Network with Multiple Planes
POXN/MP-BP	Passive Optical Cross-connection Network with Multiple Planes and Bundled Ports

RAM	Random-Access Memory
SR	Short-range
TCAM	Ternary Content-Addressable Memory
ToR	Top of Rack
WFFOC	Wavelength Flattened Fiber Optic Coupler
WDM	Wavelength Division Multiplexing
WSS	Wavelength Selective Switching

Chapter 1 Introduction

Datacenter networks (DCNs) are important for delivering web services, online services, social networks, and modern data storage infrastructures. They also play a key role in cloud computing [1]. At present, most DCN designs involve the massive usage of active and expensive devices, such as electronic switches and active optical switches. However, given the increasing demands for cloud infrastructure, the semiconductor industry has reached the physical limits of voltage scaling [2-3]. Bandwidth and power consumption requirements for future systems reach 400 PB/s and 20 MW [4], and the need to reduce these usages has necessitated changes to future datacenter architectures. At the same time, some studies [5-6] have shown that datacenters should be able to handle highly dynamic and unpredictable traffic patterns, which change constantly at a granularity of 15 ms. Furthermore, many applications involving large and distributed computations are spread across many racks, meaning that datacenters must be capable of providing high bandwidth to entire networks [7]. Therefore, current DCNs must satisfy many requirements, including, primarily, ease of deployment, the ability to handle mixed and dynamic traffic patterns, the ability to offer high bandwidth for various applications, low power consumption, and low cost to build and maintain.

The first section of this chapter describes the motivation behind our approach. The second section discusses the specific problems that exist in the design of Passive Optical Cross-Connection Networks with Multiple Planes (POXN/MPs). The third section summarizes our contributions. Finally, the last section outlines the structure of the subsequent thesis.

1.1 Motivation

Current DCNs have adopted a variety of ways of interconnecting electronic switches to handle high-volume traffic [4]. Some designs reduce costs by using commodity switches instead of non-commodity switches; however, the high power consumption of electronic switches has not yet

been considered. Other DCNs have introduced active optical switches as core elements for routing traffic across racks [4]. The drawbacks of this solution include slow switching times, expensive hardware costs, and long circuit visit delays due to pairwise connections. The Passive Optical Cross-Connection Network (POXN) [8] design was proposed to leverage passive coupler fabrics to offer reliable high-efficiency communication based on a broadcast-and-select approach, especially for multicast traffic. The Passive Optical Cross-Connection Network with Multiple Plans (POXN/MP) [9], an upgraded version of the POXN approach, is capable of achieving high throughputs for both multicast and unicast traffic with low facility cost and power consumption. If we can use the POXN/MP approach to replace one or more levels of electronic switches or active optical switches, dynamic and energy-efficient DCNs can be achieved through passive optical devices. However, this potential has not been explored.

Motivated by the advantages of POXNs and POXN/MPs, we must explore new ways to leverage these passive devices and to integrate POXN/MPs and other new devices with additional components, as needed, to build large-scale datacenters while achieving a decent tradeoff among cost, power consumption, and network performance.

1.2 Problem Statement and Objective

In POXNs [8], passive optical fabrics play a core role in enabling multicast data transmission; however, they cannot offer high throughput for unicast traffic patterns. Compared to POXNs, POXN/MPs [9] introduce another plane exclusive to unicast traffic among ports through extra-tunable transceivers, thus supporting full-bandwidth dynamic unicast traffic patterns. However, from the system-level perspective, the maximum available port count for POXN/MPs is 81 [9], which is far from sufficient to meet the requirements of large-scale datacenters. Unlike with interconnections among switches in different network tiers, it is difficult to cascade multiple

POXN/MPs to overcome the impediments of the low port density of passive optical fabric. Power budgets may not be sufficient to transmit signals through two continuous passive coupler fabrics due to insertion losses and power splits in the passive optical fabric. In addition, POXN/MPs must have the same port types. This limits their ability to engage in traffic aggregation, which is typically desired in a DCN, especially at the access level. For example, a typical top-of-rack (ToR) switch has twenty 1 Gbps downlinks and two 10 Gbps uplinks. POXN/MPs do not support this kind of configuration. To address the abovementioned problems, we set the following objectives for this thesis:

- Fit POXN/MPs into a large-scale DCN by cascading POXN/MP interconnections or by integrating POXN/MPs with other devices, such as electronic switches
- Design a new device based on POXN/MPs to engage in traffic aggregation at the access level in datacenter architectures
- Build a large-scale DCN by interconnecting proposed devices
- Design a routing scheme for multicast and unicast traffic within the proposed DCN
- Reduce total cost and power consumption as much as possible within the proposed DCN

1.3 Thesis Contribution

The contributions of this thesis can be summarized as follows:

- We propose to integrate POXN/MPs in a recursively defined structure called the Passive Optical Cube (POCube). This structure is able to achieve a decent tradeoff among total cost, power consumption, and network performance.
- We propose a Passive Optical Cross-connection Network with Multiple Planes and Bundled Ports (POXN/MP-BP) based on the POXN/MP for the mechanism for bundled ports. Compared to the old POXN/MP, the POXN/MP-BP exhibits better performance across many key

indicators, including lower hardware cost, lower power consumption, and higher bandwidth efficiency, especially in cases requiring traffic aggregation.

- We propose to replace ToR switches and core switches with POXN/MP-BPs, thus building a tree topology using POXN/MPs only or POXN/MPs and POXN/MP-BPs called the Sandwich Tree. This will significantly lower the total cost to build and power consumption to maintain large datacenters.
- We propose a new protocol, the Multiple Channels with Bundled Ports Distributed Access Protocol (MCBDAP), based on the Multiple Channels Distributed Access Protocol (MCDAP). With the introduction of bundled ports on a switch, a certain amount of overhead can be avoided during the data transfer phase. As a result, higher bandwidth efficiency can be achieved. When this measure is combined with a modified two-level routing table, packets can be delivered from server to server in a Sandwich Tree structure.
- We propose a new algorithm for collision-free transmissions within POXN/MP-BPs. The simulation results show that, compared to the old algorithm for POXN/MPs, the new algorithm can improve channel efficiency for both biased and unbiased traffic.

1.4 Thesis Structure

Chapter 2 will present four kinds of common traffic patterns, discussing representative instances for each. Next, several datacenter architectures and their advantages and disadvantages will be discussed. Common datacenter networks based on electronic switches will be examined in terms of their architectures, routing solutions, merits, and drawbacks. Then, we will illustrate some representative datacenter networks using optical technologies, including burst-mode optical circuit switches, Arrayed-Wavelength Grating (AWG), and Wavelength Division Multiplexing (WDM). Last, we will discuss two passive optical devices: the POXN and the POXN/MP.

In chapter 3, we will introduce POXN/MP-BPs at the device level. We will discuss the physical interconnections of POXN/MP-BPs, their advantages in terms of cost and power consumption, how the new MCBDAP works, how a packet is delivered from server to server, how the new algorithm for collision-free transmission is scheduled, and why this algorithm performs better than the old POXN/MP algorithm.

Chapter 4 presents our proposed POCube structure. Here, we calculate the POCube's cost and power consumption and describe its routing scheme. We also discuss how the POXN/MP-BP can be used as a design element to replace certain levels of existing switches in current DCNs. Next, we present another proposed structure: the Sandwich Tree. Finally, we compute the cost and power consumption and describe the routing scheme of this structure.

In chapter 5, we present the simulation results for the proposed MCBDAP algorithm in different scenarios. Then, we discuss packet delay and throughput for the POXN/MP-BP at the device level. Finally, we analyze the network performance of the two proposed structures—the POCube and the Sandwich Tree—based on POXN/MP-BPs and/or POXN/MPs at the network level.

Chapter 6 summarizes the thesis.

Chapter 2 Background Information

This chapter provides detailed background information on popular DCNs. First, we discuss traffic patterns, since these drive current DCN designs. In other words, new DCN designs must consider the kinds of dynamic traffic patterns they will need to support and whether they can provide sufficient available bandwidth for these patterns. The provision of bandwidth should not significantly increase hardware costs or other important network performance metrics, such as delays. Achieving a good balance among critical requirements is key for building a DCN. Therefore, following the discussion of traffic patterns, we will discuss how current DCN designs address these issues, which traffic patterns they can efficiently support, and which they cannot. Motivated by these concerns, two popular categories of DCNs will be discussed. In addition, we will discuss two related research streams on POXNs and POXN/MPs, from which our work draws intellectual inspiration, since these design elements have the potential to adapt to dynamic traffic patterns and to reduce cost and power consumption in the building of passive optical DCNs.

The first section will present four popular traffic patterns and the primary applications that frequently display these patterns in current DCNs. The second section will discuss two categories of datacenter architectures with 11 representative examples. The third section will present how POXNs work in terms of physical interconnections, power budgets, and distributed access protocols. Afterwards, the POXN/MP, as an upgraded version of the POXN, will be discussed in terms of physical system changes, protocol modifications, and performance improvements.

2.1 Application and Traffic Patterns in Datacenters

In current datacenters, the growth in Internet traffic is rapidly increasing due to the huge demands of large-scale applications, such as video streaming, web search, and distributed file system applications [10]. Server applications in a datacenter involve many domains, ranging from

banking and finance to e-commerce and social networking [11]. Most of these applications not only increase the demands on the computer hardware, which must offer more processing power, greater computation capabilities, and larger storage spaces, but also require high bandwidth to ensure the proper handling of data and the transmission of data to destination servers.

Different applications use a variety of combinations of traffic patterns. Furthermore, datacenter architectures can be divided into different groups, including campus, private, and commercial DCNs. Different DCN architectures adopt different designs to fully benefit their main services and applications. To identify solutions to accommodate various DCN traffic patterns, we must develop a better understanding of which kinds of applications fall within DCNs and how these affect traffic patterns. For this reason, we must discuss the relationship between DCN traffic patterns and DCN applications before introducing DCN architectures. This discussion serves as the foundation for our claim (in the last section) that current DCNs are driven by traffic patterns. Four main traffic patterns are common in current DCNs. We will cover each of these in detail in the following sections.

2.1.1 One-to-one traffic pattern

One-to-one cluster activities are currently very popular. One representative instance is the case of virtual machine (VM) migration, in which a live VM (including a running operating system) in one host moves to another host in a different network [12].

VM migration is often invoked in datacenters in which the physical machine on a server is overloaded. VM migration and its related applications are hot topics in datacenters, and they have attracted significant research attention. Traditional VM migration methods consider only the hardware limitations of servers. However, Shrivastava et al. [13] proposed application-aware VM migration, which takes measured real-time communication into consideration before determining

how to minimize traffic in the datacenter network.

2.1.2 One-to-several traffic pattern

The most representative application of the one-to-several pattern involves data replication, which has been topic of research interest for more than ten years. It is important that the failure of one or more network components does not negatively impact the performance of the whole network, especially for platforms in charge of vital financial operations, such as large e-commerce companies (e.g., Amazon and Alibaba).

For instance, Dynamo, which is a highly available data storage technology, provides Amazon's e-commerce platform with a simple key/value interface and good control over the tradeoffs among availability, consistency, cost-effectiveness, and performance [14]. Dynamo guarantees consistency, ensuring that updated data will reach all replicas in the end. This process involves a one-to-several traffic pattern.

2.1.3 Several-to-one traffic pattern

In a several-to-one traffic pattern, one server receives aggregated traffic sent by multiple senders. The second phase of the MapReduce [15] function clearly illustrates this kind of traffic pattern. After the first step of the mapping function, the collected result is a word and its associated count of occurrences. At the reducing function stage, for the same word, all counts from many tables or lists are summed together and merged. This process is realized via several-to-one communication.

2.1.4 All-to-all traffic pattern

All-to-all traffic communication is widely used. One of the most representative and significant of its use applications is the first phase of the MapReduce function. MapReduce is a programming

and implementation model for processing and generating large datasets from raw data, such as crawled documents, web request logs, etc. Generally, the amounts of data involved are huge, and computations must be distributed across thousands of machines. To accomplish this, a map function processes a key/value pair to generate a set of intermediate key/value pairs. Then, a reduce function merges all intermediate values associated with the same intermediate key [16].

2.2 Datacenter Network

Current DCN designs focus on building scalable and reliable networks that are capable of adapting highly dynamic traffic patterns, while ensuring modest hardware expenditures and relatively low power consumption. A decent DCN should achieve a good trade-off among these demands. Since our aim is to propose a new DCN architecture, we must first engage in a detailed discussion of current DCN architectures. This discussion will reveal the various disadvantages and design “bottlenecks” of these architectures, allowing us to avoid similar problems and achieve a better tradeoff in our own model.

Current DCN architectures can be divided into two categories: traditional DCNs based on electronic switches and new DCNs enabling optical technology. For each category, we will list the representative DCNs and discuss the architecture in terms of architectural characteristics, main components, routing mechanisms, cost, power consumption, advantages, and disadvantages.

2.2.1 Traditional datacenter network architectures

Traditional DCNs serve as a foundation for giving many designers a general view of what a DCN is and how one can be built by utilizing interconnections among existing hardware. Before introducing our transformed DCN designs, we should develop a basic understanding of the traditional DCN architecture.

Most current DCNs are composed of commodity switches organized in two-tier (a ToR switch and an aggregation switch) or three-tier (a ToR switch, an aggregation switch, and a core switch) architectures [16]. The number of servers can be as many as 48, all of which are connected through ToR switches using 1 Gbps links. In a two-tier topology, aggregation switches exist above ToR switches and connect ToR switches using much faster (10 Gbps) links. In a three-tier topology, an additional tier of core switches is added. At the top level of the topology, content switches (also called application switches) are commonly seen in medium- to large-scale datacenters. These provide many services, such as load balancing, certificate management, and bandwidth management, through the deep inspection of incoming packets [18]. In current datacenter architectures, simple tree topologies are normally applied to the connections between ToR switches and aggregation switches. By contrast, 10 Gbps, 40 Gbps, or 100 Gbps links are used to connect core switches and aggregation switches. For instance, several previous studies [17-19] show that Google is pushing for industry-wide usage of 40/100 GigE.

Figure 2-1 shows a three-tier topology with slight variance. An Ethernet edge block forms the first tier, while an access router and a core router form the second and third tiers, respectively.

Figure 2-1. Traditional network architecture

In small- to medium-size datacenters, network scale-up projects can be cost-effective and easy to set up; however, they require large up-front investments into more expensive and reliable high-capacity hardware. In particular, the maximum network scale is limited by the need for higher-tier switches and routers to handle more traffic [18]. For these reasons, in the past decade, the alternative approach of network scale-out is becoming increasingly dominant.

Figure 2-2 shows a scale-out datacenter architecture using an array of pods, each of which consists of switches interconnected in a Clos [21] topology with an edge block and ToR switches. Full bisection bandwidth can be realized through path diversity within and among the pods.

Figure 2-2. An example of a scale-out datacenter architecture

The scale-out approach offers many advantages for building large-scale datacenters:

- (1) Agility: Different applications can own network bandwidth in a modular fashion;
- (2) Scalability: With a modular scale-out approach, it is possible to add computing and storage capacity on demand. The datacenter fabric can scale while offering constant cost per-port and per-bit/sec of bisection bandwidth; and
- (3) Accessibility: Without bandwidth fragmentation and oversubscription across a huge pool of users, each user can access a wide range of computation power [18].

Up until this point, we have investigated the traditional DCN architecture built by electronic switches through either a simple scale-up approach or a scale-out approach. However, many transformed architectures utilizing Clos topologies have been proposed to build more flexible

DCNs with greater functionality, including extensive path diversity among servers, non-blocking performance, robustness-facing link failures, etc. In addition to Clos topologies, BCube, a recursive defined structure, offers the novel idea that the server plays a role in forwarding packets on behalf of other servers. We will also cover several other structures, such as butterfly and random topologies. In the following subsections, we will first present and then summarize six representative DCNs.

2.2.1.1 Fat tree

The old version of the fat tree is called a binary fat tree. This is shown in Figure 2-3. In this topology, end servers are connected by a tree structure, in which the end servers are the leaves of the tree and the other nodes are switches. Redundant interconnections exist between any two switches. The total number of interconnections increases when the level approaches the root node because higher aggregated bandwidth is required at the root node. The drawback of this structure is very apparent: The core switch is likely to become the bandwidth bottleneck in the face of high global communication demand [20].

Figure 2-3. Binary fat tree

In order to solve the bottleneck problem in the binary fat tree topology, a scalable fat tree topology based on commodity switches was proposed. This topology can be considered a transformed version of a folded non-blocking Clos topology [16]. The main feature of the fat tree topology is that, unlike in the traditional hierarchical architecture, in which all switches have different port counts, all deployed switches are identical. Figure 2-4 plots the architecture of a fat tree topology. An N -ary fat tree consists of n pods, each containing $n/2$ switches (edge switches and aggregation switches) across two levels. Each edge switch connects to $n/2$ downward hosts and $n/2$ aggregation upward switches. Like the edge switches, the aggregation switches connect to $n/2$ edge switches in the lower layer and $n/2$ core switches in the root layer. The number of core switches is $(n/2)^2$, and the numbers of edge and aggregation switches are identical: $n^2/2$ [21].

Figure 2-4. Fat tree architecture with three levels of switches in four pods

Oversubscription is a core factor that can influence network scalability. It is also a metric to quantify the amount of bandwidth shared among all servers. Given the structure of a fat tree topology with 1:1 oversubscription, if a switch port count is n , then the numbers of total switch ports and servers will be $\frac{5}{4}n^3$ and $\frac{1}{4}n^3$, respectively; thus, the number of switch ports per host is five [22]. Path diversity, another important network metric, is used to measure how many available paths exist between any two servers. This metric has the potential to improve network throughput by loading traffic balance and to improve robustness in the face of node failures. In an n -ary fat tree topology, the number of hosts is $\frac{1}{4}n^3$; thus, the number of host pairs under the same access switch is $p = \frac{n^3}{8}(\frac{n}{2} - 1)$, the number of host pairs under the same aggregation switch but different access switches is $q = \frac{n^4}{16}(\frac{n}{2} - 1)$, and the number of total host pairs is $r = \frac{n^3}{8}(\frac{n^3}{4} - 1)$. Therefore,

the average number of shortest paths is $\frac{p*1+q*\frac{n}{2}+(r-p-q)*\frac{n^2}{4}}{r}$ and the maximum path diversity is $\frac{n^2}{4}$

[22]. From this, it is easy to determine that average hop count is $\frac{p*2+q*4+(r-p-q)*6}{r} = \frac{6n^3-2n^2-4n-8}{n^3-4}$

and that the path length converges to six with an increase of n .

To simplify the routing table, the fat tree adopts a new addressing method to assign IP addresses to servers and switches. Taking $n = 4$ as an example (shown in Figure 2-4) and assuming that the 10.0.0.0/8 block is available to be assigned, the format for the switches in a pod region is as follows: 10.pod.switch.1, where *pod* represents the pod number, ranging from 0 to $n - 1$, *switch* denotes the switch position in the pod (from left to right and bottom to top), and the value is between 0 and $n - 1$. For example, in pod 0, the ToR switches (counting from left to right) on the bottom are assigned the Internet protocol (IP) addresses 10.0.0.1 and 10.0.1.1, respectively, while the aggregation switches (counting from left to right) on the top are assigned the IP addresses 10.0.2.1 and 10.0.3.1, respectively. On the other hand, the format of the core switches is as follows: 10. $n.j.i$, where j and i represent the coordinates of the $(n/2)^2$ core switches, each in $[1, n/2]$, from top to left. The end host follows the following address form: 10.pod.switch.ID, where *ID* is the position of the host in its corresponding subnet, ranging from 2 to $n/2 - 1$.

Figure 2-5. Two-level table for edge switch 10.0.1.1 in pod 0

Figure 2-6. Two-level table for aggregation switch 10.0.2.1

Prefix	Output port
10.0.0.0/16	0
10.1.0.0/16	1
10.2.0.0/16	2
10.3.0.0/16	3

Figure 2-7. Routing table for core switch 10.4.1.2

A two-level routing algorithm is applied to the fat tree. In the first routing table, each prefix-port entry has an additional pointer to a secondary table of suffix-port entries. The entries at the first level are matched from left to right (in the form of $1^m 0^{32-m}$), while those at the second level are matched in the opposite direction (in the form of $0^{32-m} 1^m$). If a destination's IP address does not terminate when it is matched in the primary table, the suffix is searched using the secondary table. For the implementation part of this two-level process, Ternary Content-Addressable

Memory (TCAM) is used to store address prefixes and suffices, and Random-Access Memory (RAM) is used to store the IP address of the next hop and its corresponding output port.

For the pod switch, traffic is routed down to servers through terminating first-level prefix lookups and routed up towards core switches through second-level suffix lookups. The core switch is assigned only a first-level prefix, since it is only responsible for routing traffic down to the correct pod using exactly one link.

For example, in Figure 2-4, server 10.0.1.2 wants to send a packet to destination 10.2.0.3. The packet first arrives at connected gateway switch 10.0.1.1. Then, it searches for the two-level lookup table, in which the left-hand entry does not yield a terminating prefix and the right-hand suffix X.X.X.3 is matched. Thus, the packet is forwarded to port 2, as shown in Figure 2-5. Once it arrives at aggregation switch 10.0.2.1, the search process is repeated. Then, the packet is forwarded to port 3, which is connected to core switch 10.4.1.2. According to Figure 2-7, core switch 10.4.1.2 matches 10.2.X.X; thus, the packet is forwarded to switch 10.2.2.1 in pod 2. Then, switch 10.2.2.1 matches the packet with entry 10.2.0.X in the first-level prefix and forwards it to port 0. Finally, switch 10.2.0.1 forwards the packet to destination server 10.2.0.3.

The fat tree is a non-blocking architecture, and it supports such functionality as ECMP, VLB, and fault tolerance. The fat tree has lower power consumption than the BCube and DCell topologies, which will be covered in the following subsections. However, though commodity switches are cheaper than high-end switches in terms of unit price, the massive usage of commodity switches and wires makes fat tree more costly than BCube and DCell topologies.

2.2.1.2 BCube

The BCube was proposed to support modular datacenters (MDCs), which differ from

traditional datacenters. The BCube is a server-centric design, meaning that the responsibility of relaying traffic is assigned to servers. Like fat tree topologies, BCube topologies use only low-end, commercial off-the-shelf (COTS) mini-switches instead of expensive, high-end switches.

A $BCube_l$ consists of n $BCube_0$ s and n^k n -port switches [23]. A $BCube_k$ has $k+1$ levels ranging from level 0 to level k and comprises k $BCube_{k-1}$ s and n^k switches with n ports and $N = n^{k+1}$ servers. Each server in a $BCube_k$ has $k+1$ ports. Normally, k is a small integer, typically less than or equal to three. The level- k port of the i^{th} server (i belongs to the range of $[0, n^k - 1]$) in the j -th $BCube_{k-1}$ (with j taking a value between $[0, n-1]$) is connected to the j -th port of the i -th level- k switch.

As can be seen in Figure 2-8, the BCube is a recursively defined structure. We can see very clearly how a $BCube_2$ is derived recursively. First, the innermost rectangle (a dashed red line), represents the most fundamental element: a switch and four servers. Then, one more level is introduced; this is $BCube_1$, and it is denoted by an outer rectangle (also a dashed red line). Each server should be equipped with one additional transceiver to connect to switches at higher levels. In this recursive way, we can depict $BCube_2$, which is denoted by the outmost red dashed line.

In the BCube topology, switches only connect with servers; they never connect with other switches. This is an obvious distinction between BCube topologies and other topologies.

Figure 2-8. The BCube's recursive structure

Next, we will discuss the network performance metrics of the BCube structure.

In terms of scalability, we calculate switch ports per host. Each level has n^k n -port switches; thus, the total number of switch ports in a BCube_k is $(k+1) * n * n^k$. Accordingly, the number of ports per host is $\frac{(k+1) n^k n}{n^{k+1}} = k + 1$.

The BCube [23] also proves six theorems. Since one of our proposed topologies is built on POXN/MPs, leveraging the features of the BCube structure, four related theorems are listed below:

Theorem 1: The longest shortest path among all the server pairs of a BCube_k is $k+1$.

Theorem 2: $k+1$ parallel paths exist between any two servers in a BCube_k .

Theorem 3: The average path length from one server to the rest of the servers is $\frac{(n-1)N}{n(N-1)} (k + 1)$.

Theorem 4: The aggregated bottleneck throughput for a BCube network under the all-to-all traffic model is $\frac{n}{n-1} (N - 1)$, where n represents the switch port number and N is the number of servers.

The BCube topology leverages a small number of network ports on the server and uses different links to connect different levels of mini-switches. A comparison of the performances of the tree, the fat tree, and the DCell topologies shows that the BCube is better than the DCell in terms of MDCs. From the perspective of ABT, the BCube far outperforms the DCell. Compared with the fat-tree topology, the BCube has a shorter shortest average path length. In building same-scale datacenters, BCubes have multiple options: They can use either two-level structures with more ports per switch or three-level structures with fewer ports per switch. By contrast, a fat tree topology has a fixed construction with three levels; however, the number of switch ports can vary. To some degree, BCubes offer network designers a variety of choices.

One main drawback of BCube topologies is that they require more ports on each server to connect with different switch levels. As a result, power consumption will increase as the total number of servers in the structure increases. The other main problem with BCube topologies is that servers, as relaying devices, tend to crash and upgrade more frequently and to require long periods of time for reboot [1]. These issues negatively affect the robustness of the network.

2.2.1.3 VL2

VL2 comprises low-cost switch ASICs, which build networks in a Clos topology [24]. VL2 uses a scale-out method to scale the whole network. Therefore, the network can offer a high aggregated bandwidth to accommodate any traffic pattern, subject to the hose model [25].

The switch in VL2 mimics a router with a routing table, thus enabling multi-path communication. A packet sent from the server will trigger a directory system to obtain the destination address and will then go through a tunnel.

Unlike conventional IP addresses, VL2 IP addresses are not fixed or binding between addresses

and locations. Instead, VL2 IP addresses fall into two categories: location-specific IP addresses (LAs) and application-specific addresses (AAs). LAs are assigned to switches and interfaces, while AAs are assigned to application servers. The mappings between these two types of addresses are stored in the VL2 directory system.

When an AA server generates a packet destined for another AA, a VL2 agent translates the ARP request into a unicast query to the VL2 directory system. This system then responds with the LA of the ToR where the packet should go [24]. This mechanism enables the usage of low-cost switches, reduces the size of the routing table for switches, and reduces overhead in the control plane by preventing it from seeing the churn in the host state, instead tasking this to the more scalable directory system [24].

Compared with the fat tree, VL2 reduces the number of cables needed to build the Clos topology. Like the BCube topology, the VL2 approach also utilizes the server's programmability; however, it uses the superior functionality of switches (rather than servers) to achieve forwarding functionality.

2.2.1.4 HyperX

Clos is the dominant high-radix topology in parallel computers today [22, 26-28]. HyperX can be considered a simple extension of either the flattened butterfly or the hypercube concept. It comprises a symmetric direct network in which all switches connect to both servers and other switches [29].

Each switch in the HyperX connects to a fixed number T of end servers and is viewed as a point in an L -dimensional integer lattice. In addition, each switch can be represented by a vector or multi-index $I \equiv (I_1, \dots, I_L)$, where $0 \leq I_k < S_k$ for each $k = 1 \dots L$ [29]. Thus, each switch connects

to $\sum_{k=1}^L (S_k - 1)$ other switches. In Figure 2-9, there are two dimensions, in which two switches (denoted by circle “R”) exist in the first dimension and four switches exist in the second dimension. Each of these connects to four servers (represented by circle “S”).

A dimensionally adaptive load-balanced (DAL) routing algorithm is proposed to route packets across different dimensions in HyperX. To better adapt to congestion, the DAL algorithm is able to deroute as many times as there are offset dimensions, to route a packet through only the subnetwork of nodes that align with the source server and the destination server, and to deroute on the path.

Figure 2-9. HyperX with $L = 2$, $S_1 = 2$, $S_2 = 4$, $K = 1$, $T = 4$.

Figure 2-9 shows that the HyperX can achieve a good balance among performance, power, wiring complexity, and fault tolerance. The HyperX can also use fewer routers to obtain lower

bandwidth on benign patterns [29].

2.2.1.5 DCell

DCell is a recursively defined structure that connects servers using only mini-switches. Each server is connecting to different levels of DCells using different links [30].

In Figure 2-10, the DCell_l network consists of five DCell_0 subnetworks. In DCell_l , every DCell_0 connects to all other DCell_0 s with different links. Each server has two links: one to connect with a server in another DCell_0 and another to connect with its mini-switch. Additionally, each server can be assigned a unique ID, represented by a 2-tuple $[a_l, a_0]$, where a_l and a_0 denote the unique level IDs. In Figure 2-10, a_l and a_0 take values within $[0, 4]$ and $[0, 3]$, respectively. DCell_k is used to represent a level- k DCell and is built in the same way as shown in the example. At level 0, each DCell_0 is viewed as a virtual node, which is fully connected to other DCell_0 s. At higher levels, each DCell_{k-1} has t_{k-1} servers. Then, DCell_k can be built with $t_{k-1}+1$ DCell_{k-1} . Like DCells at lower levels, each DCell_{k-1} can be viewed as a virtual node in order to create a complete graph.

Figure 2-10. A $DCell_1$ network ($n = 4$) consisting of five $DCell_0$ subnetworks

The DCell routing approach is called the DCell Fault-tolerant Routing (DFR) protocol, and it is very similar to shortest-path routing. There are two $DCell_{k-1}$ s, denoted by src and dst , in a $DCell_k$. The path is computed by adding the intermediate path (n_1 to n_2) between the two $DCell_{k-1}$ s in a level and the two sub-paths (from src to n_1 and n_2 to dst) together [30]. DCell offer a distributed, fault-tolerant routing approach without the need for global information. Furthermore, DCell are less expensive and consume less wire than BCube and fat tree. However, the drawback of DCell is obvious: Their aggregated bottleneck throughput (ABT) is much smaller than those of BCube and fat tree when the traffic is imbalanced across different levels of links [23].

2.2.1.6 Jellyfish

Jellyfish was proposed to offer a random graph topology using ToR switches [31]. It aims to adapt well and at a low cost to the incremental growth of future networks, requiring only 40% of the expense of LEGUP [32].

Within the Jellyfish's structure, each ToR switch i has k_i ports, r_i of which connect to other ToR switches and $k_i - r_i$ of which connect to servers. So, if the network allows N ToR switches, the network will have $N(k-r)$ servers. Under this assumption, a network can be regarded as a random regular graph (RRG) and denoted as $RRG(N, k, r)$, which is a complex problem in graph theory [33].

End-to-end throughput is determined by both the capacity of the network and the average path length through which a packet will travel before reaching its destination server. Jellyfish can take advantage of its random graph feature to lower mean path length, achieve some server numbers in fewer hops, and consume fewer network resources than rigid structures, such as fat tree and DCell.

2.2.1.7 Summary of traditional datacenter architectures

In this section, we summarize the listed traditional DCN architectures and compare them in terms of interconnection approach, routing solution, and network performance metrics.

Table 2-1. Traditional datacenter architecture comparison, where n represents the switch port count, k represents the recursive structure level, N represents the total number of servers, and S_k stands for the number of switches in a k -dimensional HyperX.

	Interconnection approach	Routing solution	No. of switches	No. of servers	Diameter	Bisection width
Fat tree	Extended version of a tree topology	Switch-centric	$\frac{5n^2}{4}$	$\frac{n^3}{4}$	6	$\frac{N}{2}$
VL2	Clos	Switch-centric	$\frac{(n-2)n^2}{4}$	$\frac{(n^2+6n)}{4}$	6	$\frac{N}{2}$
HyperX	Multi-dimensional	Switch-centric	$\prod_1^k S_k$	$n \prod_1^k S_k$	n	$\frac{\prod_1^k S_k}{4} \min(S_1, \dots, S_k)$
DCell	Recursive	Source routing	$\frac{N}{n}$	$((n + \frac{1}{2})^{2^k} - \frac{1}{2}, (n + 1)^{2^k} - 1)$	$2^{k+1} - 1$	$> \frac{N}{4 \log_n N}$
BCube	Recursive	Source routing	$\frac{N}{n} \log_n N$	n^{k+1}	$\log_n N$	$\frac{N}{2}$
Jellyfish	Random	NA	NA	NA	NA	NA

2.2.2 Enabling optical interconnect technologies

Current DCNs use a variety of traffic patterns. This creates a significant burden when the DCNs are in their communication stages. When facing the challenge of massive amounts of transferred

data and rapid changes in traffic patterns, high network performance often cannot be achieved due to oversubscription, which results in a bandwidth bottlenecks that exceed the percentage of the oversubscribed factor when a number of hosts in a pod would like to communicate with hosts in other pods. Consequently, more and more new non-blocking network designs based on electrical packet switches, such as the BCube, VL2, and the fat tree, have been proposed. These were discussed in the last section.

However, these kinds of topologies also have disadvantages: They are expensive, have complex wiring, and consume significant power [34]. Thus, motivated by the advantages of large bandwidth over packet switching, some fully optical and optical burst switch-based DCN designs have been proposed. This field has been attracting more and more attention and, therefore, is critical in any discussion of the current DCN landscape. Furthermore, our proposed architectures leverage optical devices. Thus, in the following sections, we will list several of the DCN designs that enable optical technologies (including OBSs [optical burst switches], AWG, and WDM) and explain how their characteristics benefit DCNs. Furthermore, two all-optical approaches—Proteus, which is based on active optical elements, and TONAK LION, which is based on passive optical elements—will be discussed.

2.2.2.1 Optical burst switch-based DCN

Many papers have discussed the issue of how to use OBSs in a network. In one paper [35], a burst aggregation method was proposed by building a mathematical model considering the size of the burst and the packet waiting time in an OBS network. A variety of contention resolution and avoidance schemes have been presented for bufferless OBS networks [36]. However, an OBS is an active device, which must set up light paths before sending packets. This step occupies network resources. If traffic is not significant, it will take a long time to buffer, thus increasing packet delay.

For this reason, most OBS-based DCNs are used only as bypaths, and electrical networks, such as Helios [37] and C-through [11], are retained. Helios utilizes the MEMS optical switch, and WDM technology provides higher bandwidth with less waste than the electrical packet switch-based topology. C-through is a hybrid packet and circuit-switched datacenter network architecture (called HyPac), which adds optical circuit switches between switches from different racks in order to match their maximum aggregated bandwidth demands following configuration, while maintaining the traditional packet switches of a tree topology. Now, we will discuss two representative OBS-based DCNs: C-through and Helios.

Figure 2-11 shows a C-through configuration. We can see that the DCN is divided into two parts: the electrical network on top and the optical network on the bottom. The electrical network still uses ToR, aggregation, and core switches, which are connected in a tree topology from bottom to top. The key to this approach is the use of optical circuits between pairs of ToR switches: this not only achieves higher bandwidth, but also avoids the traditional bottleneck resulting from the aggregated capacity at the top of the tree topology. In terms of cost, optically connecting ToR switches instead of nodes alleviates the complexity caused by optical links.

Figure 2-11. C-through architecture

To realize HyPaC, certain system requirements for end hosts, ToR switches, and optical configuration managers must be met. First, applications on servers need to estimate traffic demands by increasing the per-connection socket buffer size and observing end-host buffer occupancy at run-time [12]. The server kernel de-multiplexes traffic either to the optical or the electrical circuits, depending on the results of the per-rack output traffic scheduler. In addition, the utilization of optical circuits is maximized by buffering in the TCP socket and sending when bandwidth is available. Second, regarding the design of the ToR switches, C-through assigns ToR switches two different VLANs that logically isolate the optical network from the electrical network. VLAN-s handles packets destined for the packet-switched electrical network, and VLAN-c handles packets that go directly to another rack via the optical path [12]. Third, the optical manager must collect statistics that report on the amount of traffic among different pairs of racks from the management daemon running in the server. It then must optimize this information to achieve

maximum aggregated bandwidth across the whole network. To determine how the optical links should be configured, C-through uses Edmond's algorithm, which is fast enough to compute the configuration of 1000 racks within a few hundred milliseconds.

Helios, another hybrid electrical/optical switch architecture, can significantly reduce the cabling, cost, number of switching elements, and power consumption relative to recently proposed DCN architectures [37]. This architecture is similar to the C-through architecture; however, Helios uses an optical technology based on wavelength division multiplexing (WDM).

The idea for Helios stemmed from the objective of building a non-blocking switch fabric to prevent bandwidth bottlenecks between arbitrary sets of nodes in order to support efficient inter-pod communication and to achieve flexible computations and services in DCNs. The dilemma is full bisection bandwidth, which is considered to be the only method for delivering on-demand bandwidth between end servers; however, this is rarely required by the datacenter. Since the optical port is able to carry multiple 10 Gbps links, it is a waste of resources to apply an optical link to transmit no faster than 10 Gbps. However, when there is a high burstiness in the aggregated traffic demand among pod pairs, the number of circuits required to support the bursty communication in the datacenter would be prohibitive. Such bursty communication is better suited to electrical packet switching. Thus, the strengths of one type of switch compensate for the weaknesses of the other type [37].

The Helios architecture is depicted in Figure 2-12. The Helios model consists of two layers of switches—a pod switch and a core switch—in a multi-rooted tree topology. One thing that should be noticed is that half of these uplinks (called superlinks) are connected to electrical packet switches, while the other half are connected to the multiplexer before they reach the optical circuit switches. The number of wavelengths determines the size (w) of these superlinks.

Figure 2-12 Helios architecture

As an example, in Figure 2-12, there are four hosts and four transceivers in each pod, and the wavelength size w is two. In terms of bandwidth, this example delivers full bisection bandwidth. The bandwidth is divided into two identical parts, one of which is shared for packet switching between pods, and the other one of which is assigned to specific pod pairs. This example indicates that one point of the datacenter, in terms of traffic pattern, has a greater than 50% inter-pod traffic proportion and that the traffic pattern changes on a multi-second timescale. Thus, the Helios works well and is superior to other topologies [37].

The most valuable idea of this proposal is the deployment of currently commercial optical components: Glimmerglass optical circuit switches and WDM SFP+ transceivers in pod switches. In terms of cost, energy consumption, and cabling complexity, Helios can offer more benefits, compared with non-blocking electronic switches, to the datacenter, where inter-pod traffic demands do not change too rapidly.

The drawbacks of a hybrid datacenter with an optical network based on optical burst switches are clear. First, such a datacenter cannot offer decent network performance for latency-sensitive applications and cannot adapt to rapid changes in traffic patterns in the current datacenter network due to the slow reconfiguration of the MEMS optical switch. Secondly, the MEMS optical switch drastically increases the total expenditures required to build and upgrade DCNs. Thirdly, optical light paths can only be set up until the accumulated traffic reaches a threshold, which means the provision of sufficient buffer is critical. Low traffic will lead to longer waiting times for buffered packets, thus drastically increasing packet delay. This is also why optical networks can only handle a portion of traffic as a bypath approach, while retaining electrical networks as a core network.

2.2.2.2 AWGR-based TONAK LION switch

Arrayed waveguide gratings (AWGs) are another popular optical device. Arrayed waveguide grating routers (AWGRs) are arrays of AWGs. The main difference between OBSs and AWGs is that AWGs are passive optical devices. As optical routers, AWG and AWGR are core elements for building DCNs in many designs [38-39]. We will explore the AWGR-based TONAK-LION switch, discussing its architecture, operating mechanism, and main disadvantages.

An AWGR with a large port count requires precise control of channel spacing, and the maximum port count is constrained by size, fabrication constraints, and inter-channel crosstalk [40]. To overcome these limitations, an AWGR with a high port count is constructed using many smaller AWGRs with fewer wavelengths. The AWGR-based TONAK-LION switch combines the distributed all-optical token (AO-TOKEN) and the all-optical NACK (AO-NACK) technologies. The AO-TOKEN technique replaces a centralized control plane by introducing the reflective semiconductor optical amplifier (RSOA).

Figure 2-13. TONAK LION switch architecture

Figure 2-13 shows the TONAK LION switch architecture. The optical control plane is depicted with blue lines. The red part represents the core element: the TONAK line card, which is used to combine AO-NACK and AO-TOKEN. H_x is the host, and OCA is optical channel adapter, which contains an ingress buffer queue used to buffer the denied packet, a tunable laser (TL) used to tune the wavelength to the corresponding receiving port, and a NACK detector used to judge whether the token request for an incoming packet is accepted. The red part represents the TONAK line card, which contains a TOKEN Detector (TD), an edge detector (ED), a FPGA module, a 1:2 TOKEN switch (the blue line), a 1:2 DATA switch (in black line), and two circulators.

Assuming H_1 wants to send a packet to another host H_N , H_1 will first generate a TOKEN

Request (TR) by tuning TL to λ_{1N} , then go through the designated AWGR input/output port in the optical control plane, and finally enter an RSOA to request the token. The RSOA will amplify the TR and reflect it back to the AWGR input port [40]. This will then be extracted by the circulator in the TONAK line card. Afterwards, the TD will convert this to an electrical signal that is larger than a voltage and proportional to the optical power of the reflected TR, which will be recognized by the controller (FPGA), which will change the 1:2 switch to a cross-state that will allow the packet to be delivered to the designated output port of the AWGR.

The above scenario describes transmission without contention. Assuming the token request is granted, but the packet transmission is not completed, at this time, H_2 generates a token request to the same destination host H_N . Similarly, the token request enters an RSOA. However, the output N is already saturated by the TR sent from H_1 at λ_{1N} , so this RSOA reflects back the TR sent by H_2 to the AWGR input with a lower power. The controller will also recognize this as a denied token request and set the 1:2 switch to a bar state, which will immediately block the incoming packet and send it back to TX.

Due to the elimination of the electrical loopback buffer, to avoid contention, the denied packets will be blocked in the buffer at the host. The resulting packet retransmission will lead to more latency. Another drawback is the high hardware cost caused by the wide usage of expensive active elements, such as FPGA and RSOA.

2.2.2.3 Proteus

In the last subsection, an AWGR-based all-optical solution was discussed. In this subsection, we will examine another all-optical solution using the WDM approach. WDM, a very popular technology, enables the transmission of multiple wavelengths in a network, thus increasing

network capacity. As a result, it is very common to see the wide use of WDM in current DCN designs. Proteus, as a typical all-optical DCN using WDM technology, will be discussed as follows.

Proteus leverages WDM, WSS, MEMS, optical circulators, and optical transceivers to build a flexible topology, as shown in Figure 2-14. It combines the capability of optical fibers with that of WDM to mix multiple wavelengths. At the same time, with the assistance of WSS, a ToR can connect to the MEMS optical switch through mux/demux and WSS. The benefit of this design is that if ToR A wants to communicate with ToR B, ToR A can use w times the maximum speed of a single port, depending on the amount of traffic that needs to be sent. WDM can mix different wavelengths into a single fiber, which WSS receives and splits into corresponding ports in the MEMS optical switch. In this way, Proteus achieves dynamic bandwidth between any two ToRs.

Figure 2-14. Sending and receiving infrastructure of Proteus

Regarding the wavelength assignment problem, Proteus assign each ToR a unique wavelength, meaning that a ToR sends and receives traffic using a fixed wavelength. Only in this way can all

wavelengths at a ToR be multiplexed and demultiplexed without collision. In addition, a topology manager (TM) is used to calculate the configuration of an optical switch. With regard to algorithms running in the TM, a Mixed Integer Linear Program (MILP) is leveraged to localize high traffic volumes and calculate the WSS link capacity, thus achieving high throughput and low latency.

The drawback of Proteus is that its massive usage of expensive optical components, such as WSS and MEMS optical switches, substantially increases the cost of building Proteus-based datacenters. In addition, due to both the limitations in maximum channels of DWDM technology and the port density of the ToR switches, Proteus' low scalability hinders its performance in DCNs.

2.2.2.4 Quartz

In the previous subsections, we discussed how optical networks are added to existing electrical networks (C-through and Helios) and how all-optical DCNs are established from scratch (Proteus and AWGR-based TONAK LION Switch). However, we have not yet discussed how to replace parts of devices in existing electrical networks. To explore this possibility, we will discuss an approach that replaces portions of electronic switches in different structures using WDM technologies and add/drop multiplexers.

Quartz mimics a large port switch in a full-mesh topology, using WDM technology as its core technology. Quartz can replace portions of switches, which can be ToR switches, aggregation switches, or core switch, to obtain DCNs with lower network latency.

The Quartz architecture is illustrated in Figure 2-15. Quartz consists of ToR switches, transceivers, and add/drop multiplexers. In addition to n standard ports connecting to n servers, each ToR switch has k additional transceivers to communicate with other ToR switches [41]. Two adjacent switches have one direct connection using a fixed wavelength. For example, the green

line represents wavelength λ_{12} , which is used exclusively for communication between switches 1 and 2. Non-adjacent switches, such as switches 1 and 3, also have a single dedicated wavelength λ_{13} (the red dotted line in Figure 2-15) to communicate with each other. When the λ_{13} channel passes through the first multiplexer in switch 2, it will be dropped and passed to the second multiplexer. Eventually, the λ_{13} channel will be received by the transceiver in switch 3. This method of connecting switches not only lowers the wiring complexity but also changes a complex $O(n^2)$ mesh to a simple $O(n)$ ring.

Figure 2-15. Quartz architecture

If the left arrow and right arrow in Figure 2-15 are connected, the result is a Quartz ring of size 3. An Integer Linear Program (ILP) formulates the wavelength assignment problem to calculate the minimum number of channels needed to build a Quartz with a certain size. Since current fiber optics can accommodate a maximum 160 channels, the current maximum ring size is 35 [41].

Due to its high volume of additional transceivers and WDMs, Quartz is much more expensive than traditional DCNs [41]. In addition, this approach only lowers the network latency; it does not improve other performance metrics. Thus, Quartz has a high total cost and is difficult to implement.

2.2.2.5 Performance metrics

In this section, we summarize different architectures using optical interconnection technologies and compare them in terms of key device, cost, network performance, and scalability solutions.

Table 2-2. Optical architectures comparison

	Category	Key device or technology	Cost	Power consump tion	Network latency	Scalability	Scalability solution
C-through	Hybrid	MEMS	High	Low	High	Medium	320*320 MEMS in fat tree or butterfly
Helios	Hybrid	Glimmer-glass MEMS	High	Low	High	Medium	320*320 MEMS in fat tree or butterfly
Proteus	All-optical	MEMS, WSS	High	Medium	High	Low	Only 80 ToRs
TONAK LION	All-optical	Optical transceiver, tunable laser, FPGA, RSOA	High	High	High	Medium	N*N switch fabric using 128*128 AWGR
Quartz	NA	Add-drop mux/demux, 160-channel DWDM	Medium	High	Medium	Low	64-port switch in a full mesh to mimic 1056-port switch

2.3 Passive Optical Device

Following our discussion of current DCNs, we find that traditional DCNs based on electrical switches face certain problems, such as high hardware costs and high power consumption in cases of the massive usage of electronic switches. In addition, electronic switches, as core modules, do not, by nature, support various traffic patterns (e.g., multicast and broadcast patterns). On the other hand, DCNs using optical technology also have disadvantages, such as long reconfiguration times for OBS-based DCNs and high costs for active optical modules (e.g., MEMS and WSS). Like electrical networks, they are unable to support dynamic traffic patterns. For instance, if OBS is transmitting broadcast traffic, circuit visit delays will cause longer packet delay.

From this point, two passive optical devices, POXN and POXN/MP, both of which consume little power, cost little, and can provide broadcast transmission channels, attract our attention because of their potential to be leveraged as core elements to build DCNs. In the following subsections, we will discuss these two related devices in detail.

2.3.1 POXN

Ni et al. [44] proposed POXNs for DCNs in which passive optical couplers are core devices. The physical layer construction of a three-port POXN is illustrated in Figure 2-16. Each port facing the optical interconnect has a transmitting port and a receiving port associated with an array of fixed wavelengths, all working at different wavelengths. The wavelengths are combined by a multiplexer before they reach the coupler in the middle. In the reverse direction, all wavelengths are split by a de-multiplexer before reaching the Network Interface Card (NIC). There are no active optical devices in this system; instead, the power budget determines how many ports a passive coupler can have. An interconnection topology within a 9*9 coupler fabric is illustrated in Figure 2-17.

Figure 2-16. Three-port POXN physical layer

Figure 2-17. Interconnection within a 9*9 coupler fabric using Banyan topology

In our case, since the worst-case loopback fiber distance is considered to 20 km, which is

enough to accommodate tens of thousands of servers in a modern datacenter, the port count of a coupler can be determined by calculating the system's power budget and total power loss, which is caused primarily by fiber attenuation losses (4 dB), AWG (3.5 dB), WFFOC (2.5 dB) and the power loss caused by the coupler. For an $N \times N$ coupler fabrics built by 3×3 couplers, the power split loss can be calculated as $5.47 * [\log_3 N] - 0.2\text{dB}$, where 5.47 is calculated as the sum of fundamental power split loss (4.77dB), excess loss (0.5 dB), and fiber splice loss (0.2 dB). With the 24 dB power budget, the port count of a coupler fabric can scale up to 81 [8].

In addition, a high-efficiency distributed access (HEDA) protocol is developed. This protocol is divided into two phases: the discovery phase and the data transfer phase [8]. Each port in this system is able to discover other ports. At the end of this phase, each port knows a common reference clock, the round-trip time, the loopback times of all the joined ports, the MAC addresses and requested traffic amounts for each joined port, the starting time of the impending data transfer phase, the total number of scheduling cycles, and the size of the discovery window for the next discovery phase [8]. In the second phase, the data transfer phase lasts far longer than the discovery phase; this reduces the overhead caused by the discovery phase. Each port discovered in the discovery phase is able to send a data burst, followed by a request message containing the amount of traffic it will send in its next data scheduling cycle. This piggyback approach allows all other ports to learn of its request [8]. If each port follows the same scheduling algorithm individually and locally, the HEDA protocol can realize collision-free transmission in the POXN.

POXN's low facility cost and low power consumption makes them competitive in DCN architectures. However, POXN ports can only send their traffic sequentially, which is good for multicast traffic, but not for addressing the highly dynamic unicast traffic that exists in current DCNs.

2.3.2 POXN/MP

An et al. [9] proposed a POXN/MP, which is an updated version of the POXN that introduces a new plane for transmitting unicast traffic. A four-port POXN/MP transmission system is shown in Figure 2-18.

Figure 2-18. Physical interconnections of a four-port POXN/MP

In this architecture, four ports are connected through a coupler. The difference is that each port is now equipped with two pairs of transmitters and receivers, one of which is used for multicast traffic, and the other of which is used for unicast traffic. The multicast traffic should be sent in a one-by-one manner, while unicast traffic should be sent in parallel. More specifically, all the ports share the same wavelength; thus, they should follow a transmission schedule to route traffic to

avoid collisions.

Figure 2-19. Transmitters and receivers in two planes using different wavelengths

Figure 2-19 illustrates that the white transmitter is responsible for sending multicast traffic using wavelength λ_0 , while the black one is a fixed receiver that filters all other unicast wavelengths $\lambda_1 \sim \lambda_4$ and receives only multicast wavelength λ_0 . In the unicast plane, the red transmitter is able to tune its wavelength to any of the fixed wavelengths $\lambda_1 \sim \lambda_4$. The red transmitter on each port is bundled with its corresponding fixed receiver; these are depicted with four different colors: green, blue, orange, and purple. Unlike in the multicast plane, each tunable transmitter is able to tune to different wavelengths $\lambda_1 \sim \lambda_4$ in order to send unicast traffic to a specific receiver carrying only one wavelength at a time. This approach prevents two planes from interfering with each other, since wavelength contention will never happen as long as all ports follow the same rule. In terms of throughput, the best case is that all unicast transmitters send unicast traffic simultaneously through different wavelengths to all unicast receivers, while one multicast transmitter sends multicast traffic using the other wavelength.

Given an N-port POXN/MP, each wavelength is assigned to each pair of transmitters and receivers using two common rules:

(1) For any pair of transmitter i ($i \in [0, N - 1]$) and receiver j ($j \in [0, N - 1]$), in the unicast plane of POXN/MP-BP, there exists only one reserved wavelength: λ_{ij} ($i \in [0, N - 1], j \in [0, N - 1]$).

(2) For other transmitters and receivers, there is no other channel using the same wavelength λ_{ij} at the same time. Given these constraints, the goal is to minimize $\max_{i,j} \{t_{ij}^F\}$, where t_{ij}^F means the communication completion time between transmitter i and receiver j [9].

Then, the new MCDAP protocol is proposed to guarantee collision-free transmission in two planes. Additional information is added to the control message. Each transmitting port must share how much unicast traffic will be sent to the other receiving ports through control messages. Thus, in addition to the information on the multicast traffic of each port, each port has a global view of the amount of unicast traffic between any pair of ports. Two algorithms, Shortest Queue First (SQF) and Largest Queue First (LQF), are designed to schedule unicast traffic transmission [9].

Compared to POXNs, POXN/MPs can achieve 10 times the bandwidth with only four times the total expenditure [9]. However, thus far, POXN/MPs have only been discussed in terms of cost, power efficiency, and network performance. Until now, the issue of integrating POXN/MPs into large-scale DCNs has been neglected.

Chapter 3 POXN/MP-BP

Thus far, we have reviewed popular datacenter architectures and previous work on POXNs and POXN/MPs, from which our work gains inspiration. We find that POXN/MPs have the potential to be leveraged as core DCN devices, since they are characterized by low power consumption, low cost, and the broadcast transmission of signals. These merits make POXN/MPs more competitive devices for DCNs than electronic and optical switches. However, if a POXN/MP is integrated into a multi-tier DCN, it cannot address traffic aggregation at a certain level of the DCN, since all POXN/MP ports are homogeneous. Thus, a new device is needed that can satisfy the following three requirements: First, this device should retain the characteristics of POXN/MPs. Second, it should provide support functionality for traffic aggregation at a certain level of DCN. Third, it should not degrade performance, such as channel efficiency. Thus, we design a new device, called the POXN/MP-BP, based on the POXN/MP.

First, the physical interconnections in POXN/MP-BPs will be discussed, and an example of a four-port POXN/MP-BP will be provided. We will illustrate how wavelengths are assigned to tunable transmitters and fixed receivers in the unicast plane and to fixed transmitters and fixed receivers in the multicast plane. Then, we will examine how the new algorithm works in the new MCBDAF, with the assistance of pseudo code and a transmission scheduling example. In addition to satisfying the requirements of building a DCN with a passive optical device, POXN/MP-BPs can also bring more flexibility to transmission scheduling through their physical interconnection changes. Thus, we will discuss how POXN/MP-BPs achieve this objective and how their efficiency algorithms can be improved by reducing constant tuning times in certain scenarios.

3.1 Physical Transmission System

To support the traffic aggregation function, POXN/MP-BPs must modify the physical

interconnections of previous POXN/MPs. These modifications should be implemented without reducing channel efficiency, increasing hardware cost, or increasing power consumption.

Like POXN/MPs, and without introducing other devices, POXN/MP-BPs employ a passive cross-connection coupler fabric that acts as their core component, connecting all ports through its interfaces. Additionally, POXN/MP-BPs use two optical planes that enable dynamic traffic patterns in DCNs.

The main difference between POXN/MPs and POXN/MP-BPs is the introduction of bundled ports. In the POXN/MP, both port 3 and port 4 have two transmitters and two receivers. The multicast traffic and the unicast traffic carried on different wavelengths are delivered through a 2×1 wavelength-flattened fiber optical coupler (WFFOC) from the transmitters to the coupler and through a 1×2 AWG optical splitter from the coupler to the receivers. However, unlike the POXN/MP, the POXN/MP-BP bundles designated ports together. For instance, in Figure 3-1, ports 3 and 4 are bundled together and become a new 2-in-1 port. Accordingly, the new 2-in-1 bundled port is equipped with three transmitters and three receivers. As a result, the 2×1 WFFOC is changed to a 3×1 WFFOC and the 1×2 AWG is changed to a 1×3 AWG. The two optical planes can still work effectively, as long as the unicast transmitters use non-overlapping wavelengths to deliver unicast traffic simultaneously and the multicast transmitter uses its exclusive wavelength to broadcast multicast traffic sequentially. More importantly, these changes do not affect the scheduling performance at all and, instead, bring much more scheduling flexibility, thus improving efficiency, which will be discussed in section 3.4.3.

Figure 3-1. Physical interconnections of a three-port POXN/MP-BP

3.2 Power Saving Advantage

Bundled ports change the physical interconnection, which does not only save cost and power consumption by reducing multicast transceivers, but further lower power consumption by reducing the number of input/output ports for the coupler fabric. In this section, we will discuss how bundled ports can save power consumption for transmitters by reducing power loss caused by coupler.

The problem with the mechanism for bundled ports is that more ports are multiplexed/demultiplexed by WFFOC/AWG, which may increase the power loss for WFFOC/AWG. However, consider the existing optical technology, the power loss caused by a multiplexer/de-multiplexer is 2.5 dB for Coarse Wavelength Division Multiplexing (CWDM) based on thin-film filers [8]. Therefore, without causing extra power loss, the number of bundled ports can be multiplexed/de-multiplexed into a single fiber which connects to an $N \times N$ coupler

fabrics.

We now discuss how the POXN/MP-BP saves power budget for transmitters. Compared to the same system capacity of a POXN/MP, we assume that in a POXN/MP-BP, half of the ports are individual ports and the other half are bundled ports (each of which is B -in-1 bundled port). Now we can know that both of B -in-1 bundled port and individual port need one input/output port for coupler fabrics and there are $N/2B$ B -in-1 bundled ports and $N/2$ individual ports. This indicates we only need a $(\frac{N}{2B} + \frac{N}{2})$ -port coupler fabrics. Recall from the section 2.3.1, power loss caused by 3*3 coupler and 2*2 coupler can be reduced from $(5.47 \times \lceil \log_3 N \rceil - 0.2)dB$ to $(5.47 \times \lceil \log_3 (\frac{N}{2B} + \frac{N}{2}) \rceil - 0.2)dB$ and from $(3.71 \times \lceil \log_2 N \rceil - 0.2)dB$ to $(3.71 \times \lceil \log_2 (\frac{N}{2B} + \frac{N}{2}) \rceil - 0.2)dB$. With such decrease of power loss, we can lower the required power budget for transmitters while retaining the same system capacity.

As can be seen in Figure 3-2, it is clear to see that through saving the number of input/output ports for coupler fabrics, we can use the combination of 2*2 coupler and 3*3 couplers to build a coupler fabrics with a relatively smaller port count and thus reducing the power loss at the coupler fabrics. For example, the POXN/MP needs 5 stages of 2*2 couplers to build a 32*32 coupler fabrics, which leads to $3.71 \times 5 - 0.2 = 18.3dB$. By contrast, the POXN/MP-BP with 2-in-1 bundled ports ($B = 2$) only needs a 24*24 coupler fabrics ($\frac{N}{2B} + \frac{N}{2} = 24$) that can be built by 3 stages of 2*2 couplers and 1 stage of 3*3 coupler, which leads to $3.71 \times 3 + 5.47 \times 1 - 0.2 = 16.4dB$. It is interesting to see that two lines indicating the POXN/MP-BP with 2-in-1 and 4-in-1 bundled ports overlap. The reason is that though the POXN/MP-BP with 4-in-1 bundled ports ($B = 4$) needs a 20*20 coupler fabrics ($\frac{N}{2B} + \frac{N}{2} = 20$), we can only build a 24*24 coupler fabrics with

4 unused ports since manufacturing a unit coupler with more than 3*3 ports is not trivial with the existing technology [8].

Figure 3-2. Compare power loss at a coupler output for POXN/MP and POXN/MP-BP with 2-in-1 bundled ports and 4-in-1 bundled ports

Figure 3-3 shows that through saving the number of input/output ports for coupler fabrics, we can still use the combination of 2*2 and 3*3 couplers to build relatively smaller port count of coupler fabrics and thus reducing the power loss at the coupler fabrics. Different from Figure 3-2, POXN/MP and POXN/MP-BP with 2-in-1 bundled ports overlap. We now explain why this happens. For example, POXN/MP needs 3 stages of 3*3 couplers to build a 27*27 coupler fabrics. POXN/MP-BP with 2-in-1 bundled ports ($B = 2$) needs a 20*20 coupler fabrics ($\frac{N+1}{2B} + \frac{N-1}{2} =$

20) that can be built by 3 stages of 2*2 couplers and 1 stage of 3*3 couplers, which leads to $3.71 \times 3 + 5.47 \times 1 - 0.2 = 16.4dB$. Evidently, using 3 stages of 3*3 couplers is better than this approach. Thus, POXN/MP-BP with 2-in-1 bundled ports will choose 27*27 coupler fabrics. By contrast, POXN/MP-BP with 4-in-1 bundled ports ($B = 4$) needs a 18*18 coupler fabrics ($\frac{N+1}{2B} + \frac{N-1}{2} = 18$) that can be built by 2 stages of 3*3 couplers and 1 stage of 2*2 couplers, which leads to $3.71 \times 1 + 5.47 \times 2 - 0.2 = 14.45dB$. Thus, in this scenario, POXN/MP-BP should have a larger bundled port count to save power consumption.

Figure 3-3. Compare power loss at a coupler output for POXN/MP and POXN/MP-BP with 2-in-1 bundled ports and 4-in-1 bundled ports

3.3 MCBDAP

The introduction of bundled ports changes the unicast plane. The best times to send and receive unicast traffic will differ from those for POXN/MPs. To adapt to these changes in the physical interconnection, a new protocol is proposed.

The MCDAP is divided into two phases: the discovery phase and the data transfer phase. Like the MCDAP, the MCBDAP begins with a discovery phase, which is followed by a data transfer phase. In the following subsections, we will discuss the discovery phase and data transfer phase in general, focusing on the difference between the MCDAP and the MCBDAP.

3.3.1 Discovery phase

For the multicast plane, the first step is to send an ANNOUNCEMENT message, making sure that every port has obtained the other ports' information, including the timestamp for the message's transmission, the MAC address of the transmitters, the multicast traffic amount, and the unicast traffic amount specific to the receivers. The first port to successfully receive the ANNOUNCEMENT message will begin the timer for the discovery window, during which the other points are able to send their ANNOUNCEMENT messages. Once the window expires, all ports begin to wait for a CONFIRMATION message, which will be sent from the first successfully discovered port. The CONFIRMATION contains all necessary information on all successfully discovered ports. Each port that receives a CONFIRMATION message can generate two lists for the upcoming data transfer cycle: one is for the multicast plane and the other for the unicast plane. Since multicast traffic is sent sequentially by each port, bundled ports also follow this rule, which will not affect the data transfer phase for the multicast plane.

The message format of the POXN/MP-BP is shown in Figure 3-4. Ethertype is used to indicate which protocol is attached in the payload. All of the POXN/MP-BP control messages, including

ANNOUNCEMENT, CONFIRMATION, and REQUEST, are indicated by 0x8808. Opcode is used to differentiate the various control messages. Timestamps are used to record when each message is sent, which is used in later calculations. Payload messages store other useful information. For example, the 40 byte CONFIRMATION message contains information on all existing and newly joined ports, the total number of scheduling cycles, the subsequent data transfer phase (e.g., its starting time and two lists for unicast traffic and multicast traffic that each port can send, shown in Table 3-1 and Table 3-2), and the next discovery phase (e.g., the size of the discovery window).

Figure 3-4. POXN/MP-BP control message format

Table 3-1. Multicast traffic list obtained from CONFIRMATION message

Transponder number	Multicast traffic amount
Transponder 0	M_0
Transponder 1	M_1
Transponder 2	M_2
Transponder 3	M_3

Table 3-2. Unicast traffic list obtained from CONFIRMATION message

Transponder number	Transponder number	Unicast traffic amount
Transponder 0	Transponder 0	U_{01}
Transponder 0	Transponder 2	U_{02}
Transponder 0	Transponder 3	U_{03}
Transponder 1	Transponder 0	U_{10}
Transponder 1	Transponder 2	U_{12}
Transponder 1	Transponder 3	U_{13}
Transponder 2	Transponder 0	U_{20}
Transponder 2	Transponder 1	U_{21}
Transponder 2	Transponder 3	U_{23}
Transponder 3	Transponder 0	U_{30}
Transponder 3	Transponder 1	U_{31}
Transponder 3	Transponder 2	U_{32}

3.3.2 Data transfer phase

The POXN/MP-BP comprises several groups of bundled ports that are physically combined in each switch. The changes to the physical interconnections do not make any difference for the multicast plane. However, they do increase the efficiency of the data transfer phase for the unicast plane by eliminating a certain constant tuning time and reducing the variable idle time for waiting for an available receiver (numbers 3 and 5, respectively, in the bottom box of Figure 3-5).

In the MCBDA, each single transmitter in a bundled port has its own tunable LR (1550 nm) transceiver. Retaining this individuality, all transmitters can still use non-overlapped wavelengths to send unicast traffic to other available receivers at the same time. This improvement becomes evident when a transmitter with unicast traffic to a receiver in a 2-in-1 bundled port (e.g., transponder 2 and 3 in Figure 3-5) is allowed to deliver unicast traffic originally designated for delivery from one bundled port to another during an early stage. It is important to note that this never happens in the MCDAP. This occurs, for instance, in Figure 3-5, when transmitters 0 and 1 finish their first data bursts U_{02} and U_{13} to receivers 2 and 3, respectively. Transmitter 0 can continue sending U_{03} to receiver 2, and transmitter 1 can continue sending U_{12} to receiver 3 without tuning the wavelength, since receiver 2 and receiver 3 belong to the same 2-in-1 bundled port and are allowed to receive unicast traffic for one another.

Thus, the introduction of bundled ports provides much more flexibility for the data transfer phase in the unicast plane. A new algorithm designed to develop this potential will be explained next.

3.4 New Algorithm for the MCBDA

As noted in the last section, the introduction of bundled ports creates more flexibility in the data transfer phase for the unicast plane by reducing the tuning time for the tunable transmitter and the variable idle time for available receivers. To see if these changes improve on the old MCDAP algorithm, it is necessary to test the algorithm's efficiency.

In this section, we will explain how the new algorithm works using pseudo code and an example of three-port transmission scheduling. Furthermore, the new algorithm's improvements in efficiency, as compared to the old MCDAP algorithm, will be discussed under different scenarios.

3.4.1 Initial state

In this subsection, we will describe how the algorithm for the POXN/MP-BP works in a real case. Figure 3-6 shows a three-port POXN/MP-BP, in which each transmitter (denoted by different colors of rectangles) has a different-length queue to be sent to each receiver. Transmitter 2 and transmitter 3 belong to the same 2-in-1 bundled port. The rectangle length is proportional to the unicast queue size and is arranged in descending order at each transmitter.

Figure 3-6. Initial state of queue size in a three-port POXN/MP-BP

During the process of transmission, each transmitter can send a unicast queue in a specified wavelength to a dedicated and parallel receiver, as long as no other transmitters are using the same wavelength at the same time. For instance, in the above transmitter 3 $\rightarrow$ receiver 2 may use wavelength 2, transmitter 1 $\rightarrow$ receiver 0 may use wavelength 0, transmitter 2 $\rightarrow$ receiver 1 may use wavelength 1, and transmitter 0 $\rightarrow$ receiver 3 may use wavelength 3. Different wavelengths do not interfere with one another, and each receiver with a fixed wavelength is able to detect the correct signal.

In the illustrated POXN/MP-BP scheduling transmission, the initial state in Figure 3-6 uses different lengths of rectangles to represent different queue sizes between pairs of transmitter and receiver. Figure 3-7 presents an initial transmission schedule for the example without any optimization. Let T_{ij}^Q be the transmission time of the queue from transmitter i ($i \in [0,3]$) to receiver j ($j \in [0,3]$). Let $t_{ik}^{M^x}$ denote transmitter i suffering the k -th ($k \geq 0$) x (x is 1 or 2) type

mismatch. One of the variable idle time happens when a transmitter has finished sending all its unicast traffic but must wait for other transmitters to finish all their unicast traffic before a new cycle can start. This is called Type 1 mismatch (the yellow rectangle bar). The other one of the variable idle time happens when a transmitter has unicast traffic to be sent to receivers, but none of the receivers are available. This is called Type 2 mismatch (the red rectangle bar). Thus, the transmitter has to wait until one of the receivers is available [9]. In addition to these two kinds of mismatch, the transmitter also experiences a constant tuning time T^T that occurs when a receiver is ready to receive traffic, but the transmitter has just finished its last data transfer and must tune its transmitter to the wavelength of the receiver before beginning to send.

Figure 3-7. The transmission scheduling of a three-port POXN/MP without optimization

Our goal is to reduce the channel idle time, including the constant tuning time (the green rectangle bar) and the variable idle time that occurs while waiting for the completion of the last transmission (the yellow rectangle bar), as much as possible. We utilize the proactive scheduling

approach used in the MCDAP [9]. In addition, a transmitter in a bundled port is allowed to pick up a more appropriate queue size, depending on which algorithm is adopted, from all the bundled transmitters' queues. A receiver in a bundled port can receive traffic destined for other receivers in the same bundled port. In this way, a transmitter is allowed to send more than one cycle at the same time, thus eliminating a certain amount of constant tuning time. Variable idle time can be reduced as well; this be discussed in detail in section 3.4.3.

3.4.2 An example of transmission scheduling using the MCBDA

After discussing the initial state, we must determine whether the MCBDA could achieve better performance by reducing a certain amount of overhead. To illustrate the whole process and find our answer, we use an example.

A real transmission scheduling of the new algorithm is depicted in Figure 3-8. In addition to the notations for the transmission time T_{ij}^Q , the type 1 or 2 mismatch $t_{ik}^{M_x}$, and the constant tuning time T^T , $T_{ij \rightarrow ij'}^Q$ is introduced to indicate the queue that should be sent from transmitter i to receiver j , which is now sent from transmitter i to receiver j' (j and j' are bundled in a single node).

Figure 3-8 shows all of the events step by step as they happen over time. Each transmitter experiences nine moments before the end of the cycle. Of these, to avoid redundancy, we will focus on explaining the unique events.

At the beginning of the cycle, t_0 , the largest queue size for each transmitter is arranged at the head. Both transmitter 0 and transmitter 1 want to send their largest traffic to receiver 3. In the old algorithm, to avoid collision, this is not allowed. However, in the new algorithm, transmitter 0 is able to send T_{03}^Q to available receiver 2 instead of receiver 3, while transmitter 1 simultaneously sends T_{13}^Q to available receiver 3. This event is depicted by the notation $T_{03 \rightarrow 02}^Q$.

At time t_1 , transmitter 2 finishes its first data burst and begins to suffer a type 2 mismatch. All other transmitters continue sending packets.

At time t_2 , transmitters 0 and 1 complete at the same time. Transmitter 0 continues sending queue T_{02}^Q to receiver 2, and transmitter 1 picks up the traffic destined for receiver 2 and begins sending it to receiver 3 again. Since the receiver is not alternating at this moment, the transmitters will not experience constant tuning time.

At time t_4 , transmitter 3 is meant to deliver a larger traffic T_{32}^Q to receiver 2, but receiver 2 is occupied with the transmission of T_{02}^Q . To avoid idle time, transmitter 3 chooses to send T_{31}^Q instead.

At time t_8 , transmitter 0 finishes the last queue T_{01}^Q transmission after suffering a constant tuning time. Meanwhile, transmitter 2 will complete its transmission, followed by transmitters 1 and 3. However, transmitter 2 will experience a longer mismatch time than transmitters 1 and 3. Transmitter 1 will not suffer mismatch at all.

Figure 3-8. Transmission scheduling of a three-port POXN/MP-BP with a 2-in-1 bundled port

3.4.3 Benefits and tradeoffs

The difference between the MCBDAF and the MCDAP is that the MCBDAF provides much more flexibility in certain scenarios, thus improving algorithm efficiency by reducing type 2 mismatches and constant tuning time. To explore these benefits, we must analyze which kinds of scenarios trigger them. This is helpful when we simulate the POXN/MP-BP in a more complex environment (e.g., a network-level environment), since we can leverage this flexibility and avoid unnecessary overhead as much as possible.

We explore four scenarios that can trigger the MCBDAF's benefits, as follows:

Scenario 1:

At the beginning of each cycle, transmitter i has a much larger queue $T_{ij'}^Q$ to receiver j' than T_{ij}^Q receiver j . Considering the LQF algorithm for MCDAP, transmitter i is not allowed to send to receiver j , since receiver j is being used by the transmission $T_{ij'}^Q$. The new algorithm also cannot send to receiver j due to the potential for a wavelength collision. However, with the introduction of bundled ports, transmitter i is able to send queue $T_{ij'}^Q$ which originally was designated to be sent from transmitter i to receiver j' to receiver j at the current stage. Based on the largest queue first principle, we know that if a larger queue size can be selected, all transmitters tend to finish at roughly same time, which reduces the potential for type 2 mismatches.

Scenario 2:

Transmitter i has just finished sending its queue to receiver j (which is one of the receivers in a bundled port). Assuming j' is the other receiver in the same bundled port, if $T_{ij'}^Q$ has not been sent yet, transmitter i can keep sending unicast traffic $T_{ij'}^Q$ to receiver j without tuning to another

wavelength. Unlike in the old algorithm for the POXN/MP, in the new algorithm, transmitter i does not need to wait for another available receiver and then tune its wavelength to this receiver. More specifically, the transmitter i will not experience a constant tuning time T^T during the two continuous data bursts.

Scenario 3:

Transmitter i , which is a transmitter in a bundled port, compares its queue size with other transmitters in the same bundled port that is sending to the available receiver j . Then, it picks up the largest traffic to send to receiver j .

Scenario 4:

Transmitter i , which is a transmitter in a bundled port, pairs with receiver j . If the receiver j is also belonging to a bundled port, the queue size to be sent from the transmitter i to the receiver j will be compared with all the combinations of the other transmitters and the other receivers in each same bundled port (including transmitter i and receiver j), making the selection range much wider. As a result, larger traffic volume can be picked up and sent out.

Trade-off:

If these scenarios happen frequently during the data transfer phase, this will lead to a situation in which each transmitter tends to send more traffic to receivers in bundled ports than to normal single ports during the early stages. As a result, receivers in bundling ports may become available, while available transmitters may have no traffic to send. This situation, which is most likely to happen close to the cycle end, may increase the potential for type 2 mismatches.

The following Figure 3-9 presents the pseudo code for the core of the MCBDA algorithm,

where Q_{ij} indicates the queue to be sent from transmitter i to receiver j , $Q_{ij\text{left}}$ refers to the queue to be sent from transmitter i to each remaining receiver j , T_i and R_j refer to the normal individual transmitter and receiver, T_b and R_b refer to a transmitter and a receiver belonging to a bundled port, t_i^{idle} represents transmitter i 's idle time, t^T represents the constant tuning time, t_{current} represents the current time, t_{ij}^F represents the completion time for the transmission from transmitter i to receiver j , $t_{ij'}$ represents the transmission time for $Q_{ij'}$ (i.e., scenario 2), $t_{i'j}$ represents the transmission time for $Q_{i'j}$ (i.e., scenario 3), and $t_{i'j'}$ represents the transmission time for $Q_{i'j'}$ (i.e., scenario 4).

New Largest Queue First for POXN/MP-BP

```

1: for all transmitting port  $i$  do
2:   select  $\max\{Q_{ij}\}$  to send from transmitting port  $i$  to receiving port  $j$ 
3:   while  $j, j \in R_i$  is not available or  $j, j \in R_b$  has 0 bundled ports do
4:     select  $\max\{Q_{ij_{left}}\}$  of the remaining  $j, j \in R_i \cup R_b$ 
5:   end while
6:   disable paired transmitting port  $i$  and receiving port  $j$  and let  $Q_{ij} = 0$ 
7: end for
8: repeat
9:   a pair of  $i$  and  $j$  finishes its data burst first, enable corresponding  $i$  and
10:   $j$  and then traverse
11:  for all receiving port  $i$  do
12:    search for an available receiving port  $j$ 
13:    for all transmitting port  $i$  do
14:      search for an available transmitting port  $i$ 
15:      if an available  $i, i \in T_i$  pairs an available  $j, j \in R_i$  and  $Q_{ij} \neq 0$ 
16:      or an available  $i, i \in T_b$  pairs an available  $j, j \in R_i$  and  $\sum Q_{i'j} \neq$ 
17:       $0, i' \in T_b$  then
18:        pick up  $Q_{ij}$  or  $\max\{Q_{i'j}\}$  and disable just paired  $T_i$  and  $R_j$ 
19:        and let  $Q_{ij} = 0$  or  $Q_{i'j} = 0$ 
20:        if  $t_i^{Idle} > t^T$  then
21:           $t_{ij}^F = t_{current} + t_{ij}$  or  $t_{i'j}$ 
22:        else if  $t_i^{Idle} > 0$  and  $t_i^{Idle} < t^T$  then
23:           $t_{ij}^F = t_{current} + (t^T - t_i^{Idle}) + t_{ij}$  or  $t_{i'j}$ 
24:        else  $t_i^{Idle} = 0$ 
25:           $t_{ij}^F = t_{current} + t^T + t_{ij}$  or  $t_{i'j}$ 
26:        end if
27:      end if
28:      if an available  $i, i \in T_i$  pairs an available  $j, j \in R_b$  and  $\sum Q_{ij'} \neq 0,$ 
29:       $j' \in R_b$  or an available  $i, i \in T_b$  pairs an available  $j, j \in R_b$ 
30:      and  $\sum Q_{i'j'} \neq 0, i' \in T_b$  and  $j' \in R_b$  then
31:        pick up  $\max\{Q_{ij'}\}$  or  $\max\{Q_{i'j'}\}$  and disable just paired  $T_i$ 
32:        and  $R_j$  and let  $Q_{ij'} = 0$  or  $Q_{i'j'} = 0$ 
33:        if  $t_i^{Idle} > t^T$  then
34:           $t_{ij}^F = t_{current} + t_{ij'}$  or  $t_{i'j'}$ 
35:        else if  $t_i^{Idle} > 0$  and  $t_i^{Idle} < t^T$  then
36:          if  $j_{last}$  of  $i, i \in T_i \cup T_b == j$  then
37:             $t_{ij}^F = t_{current} + t_{ij'}$  or  $t_{i'j'}$ 
38:          else
39:             $t_{ij}^F = t_{current} + (t^T - t_i^{Idle}) + t_{ij'}$  or  $t_{i'j'}$ 
40:          end if
41:        else
42:          if  $j_{last}$  of  $i, i \in T_i \cup T_b == j$  then
43:             $t_{ij}^F = t_{current} + t_{ij'}$  or  $t_{i'j'}$ 
44:          else
45:             $t_{ij}^F = t_{current} + t^T + t_{ij'}$  or  $t_{i'j'}$ 
46:          end if
47:        end if
48:      end if
49:    end for
50:  end for
51: until all queues in each transmitting port have been sent

```

Figure 3-9. New LQF algorithm for POXN/MP-BPs

Chapter 4 Integrating POXN/MPs and POXN/MP-BPs into Datacenters

Up until this point, we have discussed the drawbacks of traditional DCNs based on electronic switches, which include high costs, high power consumption and inefficiencies in handling various traffic patterns, and the drawbacks of DCNs enabling optical technology, which include the need for complex control systems for all-optical solutions, long configuration times for OBS, and inefficiencies in handling various traffic patterns. In addition, we have discussed the new POXN/MP-BP passive optical device and its potential to be a core device in DCNs. We have also outlined the benefits of this device, which include low costs, low power consumption, and broadcast transmission channels. Previous research has failed to explore the potential of POXNs and POXN/MPs at the network level. Driven by the existing problems in current DCNs and the potential of POXN/MPs and POXN/MP-BPs, our goal is to build a dynamic and energy-efficient DCN using these two passive optical devices.

As our main contribution, we propose two DCNs, called the POCube and the Sandwich Tree, which are derived from two representative DCNs: the BCube and the Google's multi-tier DCN architecture.

In the first subsection, we will discuss POCube in terms of its structure, its routing scheme, its cost, and its power consumption. In the second subsection, we will discuss how Sandwich Tree is constructed, how to utilize POXN/MP-BPs to replace electronic switches in a multiple-tier network, and how to build a Sandwich Tree structure that leverages POXN/MPs and POXN/MP-BPs alternately. We will also discuss routing schemes, costs, and power consumption.

4.1 POCube

One of the issues hindering the scalability of POXN/MPs is their port density. Previous

calculations [8] have shown that up to 81 ports can be included in a POXN/MP. However, to build a large-scale DCN, tens of thousands of servers must be accommodated. To accomplish this, first, one could consider cascading the POXN/MP to achieve a high port count to connect more servers. Since current DWDM technology can support only 160 channels, the maximum number of ports on all servers is 160 [41]. If more servers join, the overlapping wavelengths will be recycled, resulting in transmission collisions. Furthermore, insertion losses and power splits in the passive optical fabric consume too much power, suggesting that the power budget may not be sufficient to transmit a signal through two POXN/MPs.

Second, one may attempt to increase the port count on the server or the switch, rather than on the POXN/MP itself. In this case, extra POXN/MPs are needed to connect with the increased numbers of ports on the server or switch side. Deep research into the different topologies currently available for DCNs, the recursive BCube structure meets our goal, since the POXN/MP shares some similarities with the mini-switches used in the BCube. These mini-switches do not have routing intelligence, since the BCube, as a server-centric structure, centralizes the routing intelligence in the servers. By contrasts, the POXN/MP shares the task of scheduling transmissions with the ports, not with passive optical couplers, which are only used as transmission mediums.

Therefore, we propose a new topology, called the POCube, to scale POXN/MPs in a recursively defined structure.

4.1.1 POCube structure

An example of a POCube₇ is shown in Figure 4-1. A 32*32 POXN/MP connects to 32 servers (for simplicity, AWGs, WFFOCs, and tunable and non-tunable LR (1550 nm) transceivers are not shown in Figure 4-1).

Figure 4-1. POCube structure

In a POCube structure, POCube_k has $k+1$ levels and consists of n POCube_{k-1} s and n^k n -port POXN/MPs, which connect to $n(k+1)$ -port servers. Each server has $(k+1)$ tunable transceivers and $(k+1)$ non-tunable transceivers, which connect to POXN/MP from level 0 to level k using multiple links. From the properties of the BCube (discussed in 2.2.1.2), we know that the total number of servers is $N = n^{k+1}$. In Figure 4-1, POCube_1 can cover 1024 servers using only two levels. The POCube structure can be effectively scaled by adding more server transceivers to achieve multiple levels, as is done with the BCube (shown in Figure 2-8).

4.1.2 Routing in POCube

Like the BCube, POCube_k adopts the address array $A_k A_{k-1} \dots A_0$ ($A_i \in [0, n-1], i \in [0, k]$) for its servers, such that each POCube has a unique address array. In Figure 4-2, we illustrate a POCube_1 structure with $k = 1$ and $n = 4$.

Figure 4-2. A POCube₁ consists of four POCube₀s with 16 servers

The BCube header is left unmodified to address the issue of routing by utilizing the recursive structure of the BCube. Like the MPLS header, the BCube header is located at layer 2.5, between layer 2 and layer 3. To achieve compatibility between most network applications which are based on TCP/IP, and layer 3, a fixed one-to-one mapping between IP and the BCube address array is implemented. This design will simplify the address solution between IP and the BCube address array [30].

The BCube header includes source and destination BCube addresses, the protocol type, the packet id, the header checksum, and the payload length. The BCube stores the complete path from the server to the destination, which is represented by a next hop array (NHA) in the header of each POCube packet. The NHA is composed of two-bit DPs (different positions) and six-bit DVs (different values) [23]. In our implementation, the POCube level is less than four, and the maximum path length is two, so two bits and six bits are reserved for the DPs and DVs, respectively. The number of NHAs in the header depends on the path length. The server forwards

a packet based on the NHA value indexed by the TTL field in the BCube header. Since the TTL will decrease, the NHAs will be stored in reverse order.

Figure 4-3 describes packet processing at the system level. These modules can be implemented in either software or hardware. On the receiving side, unicast and multicast packets will be delivered to the BCube stack through the unicast plane and the multicast plane, respectively, while control packets will be delivered to the POXN/MP through the multicast plane. On the other hand, the BCube header will be added to the packet before being sent to the data and control queues, and the POXN/MP will provide all control functionalities, such as the selection of a certain wavelength for a unicast packet and the optimal time for sending packets out from the NICs.

Figure 4-3. POCube packet processing at the system level

Each entry in the forwarding table is indexed by its NHA value. The number of entries in Table 4-1 is $(k+1)(n-1) = 6$. The forwarding table also has a corresponding output port for each neighbor's MAC address. When an intermediate server receives a packet, it will extract the NHA value to

search for the MAC address of the next hop and then forward the packet to the corresponding output port. In the POCube, because of the existence of multiple parallel paths between a source server and a destination server, a packet is able to traverse through different paths at different levels. For instance, in Figure 4-2, there are two equal-cost paths from server 00 to server 22 (denoted by red and green dashed lines). Let us assume the upper-layer application is based on the TCP. In a source server, a hash is based on a tuple (src IP, dst IP, src port, dst port). This guarantees that packets in a single flow will always be delivered to the same destination port. After hashing, the selected path indexed by the NHA value will be encoded in the packet header. Packet forwarding is based on a one-table lookup comprising neighbors' MAC addresses and their corresponding output ports. This information is obtained through the MCDAP for the POXN/MP. Each port on a server connecting with other servers is running an independent POXN/MP in layer 2.

Now, we illustrate how a packet traverses from server 00 to server 22 in a POCube_l structure with $n = 4$. We call the left and right ports on servers 00, 02, 20, and 22 ports A and B, ports C and D, ports E and F, and ports G and H, respectively. Only one digit of the POCube addressing form is corrected by one intermediate server. In this example, server 00 wants to send a packet to server 22. From two parallel paths (00->20->22 and 00->02->22), only one path will be selected, based on the result of the tuple hashing function described above. After hashing, the path 00->20->22 is selected, and the result is stored in the NHAs, indicating that the first hop is to 20 ($NHA_2 = 01\ 000010$) and the second (last) hop is to 22 ($NHA_1 = 10\ 000010$).

Based on the knowledge from the discovery phase of the MCDAP, server 00 maintains a forwarding table with six entries indexed by NHA value, as shown in Table 4-1. Different entries with the same output port fall within the same POXN/MP. Once a frame arrives at server 00 from

a source server, the next hop MAC address will be searched in the forwarding table, as shown in Table 4-1. Entry 01 000010, indexed by the TTL header in the POCube header, offers an exact match; thus, the POCube and POXN/MP headers will be modified accordingly. The source and destination of the POCube address will be 00 and 20, respectively. The source and destination of the MAC address in the POXN header will be the MAC addresses for servers 00 and 20, respectively. Then, server 00 will transfer the packet into the queue of output port B.

Table 4-1. Forwarding table for server 00

Entry (NHA= 2-bit DP+6-bit DV)	MAC address	Output port
00 000001	MAC for server 01	B
00 000010	MAC for server 02	B
00 000011	MAC for server 03	B
01 000001	MAC for server 10	A
01 000010	MAC for server 20	A
01 000011	MAC for server 30	A

When server 20 receives the packet, it will extract the NHA from the packet header and then forward the packet to the queue of output port F. The packet will be delivered through the POXN/MP during its data transfer phase. Finally, port G on server 02 will receive the traffic, extract the NHA value from the packet header indexed by TTL field (which is now 2) in the BCube

header, and discover that it is NHA₁, meaning that server 22 is the last hop.

Now, we will discuss broadcast and multicast traffic in the POCube. The POCube has a natural advantage in terms of sending multicast and broadcast traffic. For instance, server 00 has broadcast traffic to send, as shown in Figure 4-4. Server 00 will make a duplicate packet and then deliver the packets from two output ports. In POCube₀0, servers 01, 02, and 03 will receive the packet through a level-0 POXN/MP. The original packet will arrive at servers 10, 20, and 30 through a level-1 POXN/MP. Then, servers 10, 20, and 30 will make duplicate packets and forward them to other servers within each POCube₀. In this way, only four duplicate packets in four servers will be created. Regarding multicast traffic, we can assign a server to collect the IGMP message, maintain the multicast group, and forward the state to each server. Each server will then be able to judge whether a received multicast packet corresponds with its joined multicast group.

Figure 4-4. POCube broadcast

In the BCube, both broadcast and multicast traffic are delivered using a unicast mechanism.

Therefore, in the same scenario, server 00 requires 14 duplicate copies, all of which are sent one-by-one to the other servers. Compared with the BCube, the POCube is more efficient in terms of saving bandwidth in multicast and broadcast transmissions.

4.1.3 Cost and performance

Considering the characteristics of this recursively defined structure and the POXN/MP, it is more desirable, in terms of the tradeoffs among cost, power consumption, and network performance, to cover the same number of servers using larger POXN/MP ports and fewer levels than to use fewer ports of coupler fabric and more levels.

For instance, let us compare two methods of building a DCN covering 1024 servers.

The first solution is to connect four servers with a 4*4 POXN/MP. Since POCube_k consists of n POCube_{k-1} with n^k n -port POXN/MPs and $N = n^{k+1}$ servers, the level k will become four and the number of servers will be $4^5 = 1024$ when $n = 4$. However, when the number of levels increases by one, one more group of tunable and non-tunable LR transceivers will need to be added to each server. The resulting 10 transceivers per server times 1024 servers yields 10240 transceivers, which significantly increases the total cost. The average path length is the same as that achieved in a BCube.

Another solution is to connect 32 servers using a 32*32 POXN/MP. As can be seen in Figure 4-1, the POCube_1 is made up of 32 POCube_0 s. Within each POCube_0 , 32 servers are connected by a 32*32 POXN/MP. From the perspective of POCube_1 , there are another 32 32*32 POXN/MPs, each of which connects with an additional port on each server.

Table 4-2 compares these two methods. Since the POXN/MPs replace the mini-switches, the original packet sent from the server will enter the POXN/MP first and then head to the server. If

this server is the destination switch for the destination server, the packet will be forwarded from the switch to the server. Based on the formula for the path length calculation, the average lengths of POCube₁ with two levels and POCube₄ with five levels are 1.94 and 3.75, respectively. Furthermore, as can be seen from Table 4-3 and Table 4-4, only two pairs of tunable LR (1550nm) transceivers and non-tunable LR (1550nm) transceivers can significantly achieve higher power efficiency and lower hardware cost. In terms of cost and power consumption, POCube₁ offers a desirable solution for a datacenter covering 1024 ToRs. If the ABT is the first priority, POCube₄ would be adopted at the expense of cost and power.

Table 4-2. The first solution is a POCube₄ built with 4*4 POXN/MPs with five levels, and the second solution is a 2-level POCube₁ built with 32*32 POXN/MPs.

No. of elements No. of levels	Server number	Coupler no. / input * output	SFP+ no. / SFP no.	Average path length	ABT (Gbps)
5	1024	596/4*4	5120/5120	3.75	1365
2	1024	64/32*32	2048/2048	1.94	1056

Table 4-3. Cost comparison of the two solutions

Topology	Total cost of each element (Total no. * unit price (USD))					Total cost (million USD)
	Non- tunable LR (1550nm)	Tunable LR (1550nm)	NIC & switch fabric	AWG/ WFFOC	Coupler	
POCube ₄	5120*350 [42]	5120*525 [17]	5120*450	5120*40	596*40	7.01
POCube ₁	2048*350	2048*525	2048*450	2048*40	64*320	2.82

Table 4-4. Power consumption comparison of the two solutions

Topology	Total power consumption of each element (Total no. * unit power consumption (W))					Total power consumption (KW)
	Non- tunable LR (1550nm)	Tunable LR (1550nm)	NIC & switch fabric	AWG/ WFFOC	Coupler	
POCube ₄	5120*1.5 [43]	5120*1.5 [44]	5120*12.5 [45]	5120*0	596*0	79.36
POCube ₁	2048*1.5	2048*1.5	2048*12.5	2048*0	64*0	31.74

4.2 Sandwich Tree

In the last subsection, we explored a POCube architecture recursively built with servers and

POXN/MPs. However, this structure has not yet been implemented in the field. Through the above calculation for the POXN (section 2.3.1), we can ensure that the POXN/MP is able to interconnect with servers or switches in a large-scale datacenter as long as its port count is below the maximum value of 81. Based on current technology, the power budgets for tunable and non-tunable transceivers do not allow the cascading interconnections that are possible in POXN/MPs. However, though it is not possible to cascade POXN/MP or POXN/MPs, it is both possible and practical to use POXN/MPs or POXN/MP-BPs alternatively in a multi-tier structure. Based on Google's DCN architecture, we propose the Sandwich Tree structure, in which we can alternately place switches and POXN/MP-BPs or POXN/MPs. The motivation behind this approach is to increase uplink bandwidth from servers to switches, reduce the number of network tiers, and lower power consumption and total expenditure.

In this section, we will discuss how to build a scale-out DCN using POXN/MPs and POXN/MPs. Since this structure is derived from the Google's DCN architecture, with POXN/MPs and switches that alternate in the network layers, we call it the Sandwich Tree. It is similar to fat tree topology; however, compared with the traditional fat tree topology, the Sandwich Tree introduces POXN/MP-BPs and POXN/MPs to replace portions of switches and we assume that there is traffic aggregation from servers to switches through first-level POXN/MP-BPs.

The first subsection explores how POXN/MP-BPs replace different layers of switches in detail. The second subsection investigates how the Sandwich Tree is constructed. The third subsection discusses how routing mechanisms work for unicast, multicast, and broadcast traffic. The last subsection illustrates the advantages of the Sandwich Tree over the traditional fat tree structure in terms of cost and power savings through detailed calculation.

4.2.1 POXN/MP-BPs as design elements to replace switches

To build a Sandwich Tree, we just address the issue of how to replace certain levels of switches in a multi-tier tree structure. For instance, the ToR switch aggregates traffic from source servers and is asymmetric in structure, with uplinks and downlinks that differ in speed and number. The homogeneous structure of the POXN/MP is not capable of addressing this kind of traffic aggregation; thus, in this case, the POXN/MP-BP becomes the only choice. So, in this section, we will discuss how to replace certain levels of switches in a multi-tier structure with POXN/MPs or POXN/MP-BPs.

Multi-stage Clos topologies built from massive silicon commodity switches can support the building of large-scale DCNs [1]. For example, Google's DCN has been developed and upgraded to new generations every one to two years since 2004. The foundation of all of such architectures is a multi-stage structure, which can be divided into three aspects: ToR switches, aggregation blocks, and spine blocks. One instance of such a structure is shown in Figure 4-5. The extensive use of switches in a multi-stage network will significantly increase the network latency and the total cost. Thus, our goal is to deliver a network with a low hop count and a low hardware cost by replacing groups of switches using POXN/MPs or POXN/MP-BPs.

Figure 4-5. One example of Google's datacenter architecture with five stages of switches

Now, we explain how POXN/MP-BPs or POXN/MPs can be leveraged to replace groups of switches. First, we will discuss the replacement of individual switch layers; then, we will explore the simultaneous replacement of multiple layers of switches.

First, we examine the ToR switch at stage 1. Assuming that 20 servers are connecting with ToR switches using twenty 1 Gbps uplinks and that the top ports of the ToR switch are connecting with stage 2 switches using two 10 Gbps uplinks, the old physical interconnections of POXN/MPs will not work to replace the ToR switches, since all POXN/MP ports are homogeneous. Therefore, a POXN/MP-BP will perform its function here to replace the ToR switch with two groups of 10

bundled ports on the two switches and 20 individual ports on each server. More specifically, the POXN/MP-BP will consist of a 48×48 coupler; 20 individual ports on the server side, each of which will be equipped with a 1×2 AWG, a 2×1 WFFOC, and a pair of tunable and non-tunable transceivers; and two groups of 10 ports on each switch side, each of which will be equipped with a 1×16 AWG, a 16×1 WFFOC, 10 tunable transceivers, and one non-tunable transceiver. Though a 40×40 coupler is sufficient, unit couplers can be only 2×2 or/and 3×3 ; thus, input/output numbers can be any integer numbers composed solely of prime factors of 3 and/or 2 [8]. The case is the same for the AWG and the WFFOC.

Second, switches at stages 2, 3, and 4 have same numbers of uplinks and downlinks. Either POXN/MP-BPs or POXN/MPs can be utilized to individually replace all switches in one of these stages. For instance, a stage-2 switch has four 10Gbps uplinks and four 10Gbps downlinks, which can be easily replaced by an 8×8 POXN/MP.

Third, like switches at stages 2, 3, and 4, switches at stage 5 can also be replaced by POXN/MPs because of their symmetric characteristics. The port counts for the POXN/MPs can differ, and transceivers in this stage are normally equipped with faster speed (e.g., 40/100 Gbps).

In addition to individual replacements within certain stages, POXN/MPs and POXN/MP-BPs can realize multiple replacements at different stages simultaneously; however, no two replacement stages can be adjacent. For instance, stage 1, stage 3, and stage 5 can be replaced simultaneously by POXN/MPs or POXN/MP-BPs. However, stage 1 and stage 2 or stage 4 and stage 5 cannot be replaced at the same time because of problems caused by POXN/MP interconnections, as already explained in section 4.1.

Therefore, we can conclude that POXN/MPs or POXN/MP-BPs can be leveraged to replace,

at most, $\frac{k}{2}$ and $\frac{k+1}{2}$ stages, respectively, of a Clos datacenter network with k even and odd stages.

Furthermore, since it enables the bundled ports with random numbers, the introduction of POXN/MP-BPs can even benefit random topology networks, as long as the total number of ports does not exceed the port count limitation of the POXN/MP-BPs.

4.2.2 Sandwich Tree Structure

There are three main reasons for proposing Sandwich Tree structure with POXN/MP-BPs. First, POXN/MP-BPs can provide greater uplink bandwidth on certain switch ports because of their mechanism for bundled ports. Second, the introduction of POXN/MP-BPs will lead to lower hardware costs and higher energy efficiency. Third, POXN/MP-BPs can further save power budget for transmitters by reducing the power loss for the coupler fabric.

Figure 4-6 shows the construction of an eight-pod Sandwich Tree structure. The Sandwich Tree structure replaces two levels of electronic switches—the ToR switch and the core switch—with POXN/MP-BPs and POXN/MPs, respectively.

It is important to note that we apply the fat tree addressing methodology. In a k -pod network, there are two forms of addressing, which correspond to two types of devices: servers and switches. The address form of the server can be denoted as follows: 10.pod.coupler.ID, where pod refers to the pod number ($\text{pod} \in [0, k-1]$), coupler represents the position of the coupler to which servers are connecting ($\text{coupler} \in [0, \frac{k}{2b}-1]$ (where b is the number of bundled ports), and ID represents the host position ($\text{ID} \in [2, \frac{k}{2} + 1]$). For each switch, the address can be represented by 10.pod.switch.1, where pod has the same meaning as in the server address, switch refers to the position of the switch (from left to right) in a pod, and ID number is replaced by a constant 1, which can be used to differentiate switch from server.

Figure 4-6. An example of an eight-pod Sandwich Tree with POXN/MP-BPs and POXN/MPs

4.2.3 Routing in Sandwich Tree

In this section, we will explain how unicast and multicast traffic are delivered from the source server to the destination server(s) in a Sandwich Tree topology.

First, we will discuss how unicast traffic is sent from server to server. Before this protocol can work, two questions must be solved. First, how can a switch determine the packet destination and insert the packet into the correct transponder when it receives a packet from a server? Second, how do the transponders on the upper level of switches, which are connected by the top level of POXN/MPs, know which transponder (s) to send their packets to?

Our solution to the first question leverages a new two-level routing table, which can be

implemented in hardware using CAM [46]. This offers a fast approach to finding a match. The approach draws on the fat tree topology, which also implements a two-level lookup table to forward packets. Before the routing table is introduced, the transponder number of switch 10.0.0.1 is clearly presented in Figure 4-7. The lower level of the switch comprises four groups of bundled ports, each of which has one transponder (i.e., 0, 3, 6 and 9) for multicast traffic and four 2-in-1 bundled ports, each of which has two transponders (i.e., 1, 2, 4, 5, 7, 8, 10 and 11) for unicast traffic. On the top side, there are eight normal ports, each of which has one transponder (i.e., 12, 14, 16, 18, 20, 22, 24, and 26) for multicast traffic and the other transponder (i.e., 13, 15, 17, 19, 21, 23, 25, and 27) for unicast traffic.

Figure 4-7. Transponder numbers 0 to 23 for switch 10.0.0.1 in a POXN/MP-BP with eight pods

Prefix	Transponder	Suffix	Transponder
10.0.0.4/30	1	0.0.0.2/16	13
10.0.0.0/30	2	0.0.0.2/17	15
10.0.1.4/30	4	0.0.0.3/16	17
10.0.1.0/30	5	0.0.0.3/17	19
10.0.2.4/30	7	0.0.0.4/16	21
10.0.2.0/30	8	0.0.0.4/17	23
10.0.3.4/30	10	0.0.0.5/16	25
10.0.3.0/31	11	0.0.0.5/17	27
0.0.0.0/0			

Figure 4-8. Two-level routing table for switch 10.0.0.1. It will deliver an incoming packet to a specific transponder depending on the destination IP address of the packet.

Figure 4-8 illustrates the two-level routing table for switch 10.0.0.1. First, packets generated at each server can be classified into intra-rack traffic and inter-rack traffic. Since first-level ToR switches are replaced by POXN/MP-BPs in the Sandwich Tree, servers must make a decision regarding which switch to send their packets to. Source servers can solve this problem using many approaches, as long as the switch that receives the packet can forward the packet to the correct transponder. Here, we consider a simple approach. Let us assume the upper layer application is based on TCP. A hash function based on a tuple (src IP, dst IP, src port, dst port) is implemented in each source server. Subsequent packets will follow the same path, thus avoiding packet reordering. The same approach is also used by servers in the POCube. For example, if server 16

10.0.3.5 sends a packet to server 1 10.0.0.2, according to the hash function, the packet will first be forwarded to transponder 10 on switch 10.0.0.1. Once the switch receives the incoming packet, it determines the next hop based on the two-level routing table. Each entry in the first-level prefix will be searched first, until the second entry, 10.0.0.0, is found. Then, the packet will be forwarded to transponder 2. From there, the packet will be delivered to destination 10.0.0.2.

For the inter-rack traffic, the only difference is that the packet will traverse one more level of POXN/MPs. For instance, let us assume that server 128 10.7.3.5 in pod 7 sends a packet to the server 1 10.0.0.2 in pod 0. In the first stage, according to the hash function, the packet will be forwarded to transponder 10 on switch 10.7.1.1. Then, the longest-matching prefix search does not yield a terminating prefix, so the second-level suffixes will be searched. Based on this search, transponder 13 is found, and the packet is forwarded to it. Through the upper-level POXN/MPs, the corresponding transponder 13 on switch 10.0.1.1 will receive the packet and forward it to transponder 2. From there, the packet will be delivered to destination 10.0.0.2.

Second, we will discuss how multicast and broadcast traffic is transferred in the Sandwich Tree.

As we can see from Figure 4-9, server 1 wishes to send broadcast traffic to all other servers in the Sandwich Tree. In order to fully utilize the characteristics of the multicast plane for POXN/MPs or POXN/MP-BPs, we must use other switches to relay broadcast traffic to local servers within a pod because this will reduce the number of duplicate packets on the switch side. For instance, in Figure 4-9, both switch 1 and switch 2 receive the packet. Then, switch 1 forwards the packet through its upper ports, while switch 2 forwards the packet through its lower ports. If switch 1 is selected to forward the packet to servers 3 and 4, it must duplicate one more packet and then forward it through its lower port. The original packet will continue traversing another POXN/MP on the top and arrive at switches 3, 5, and 7. Here, each switch will make a duplicate packet, which

will be sent through the switch's lower port. In this way, only three duplicate packets are generated in the switches. Regarding multicast traffic, we use the same approach proposed by Portland [47]. A centralized manager is added to collect IGMP join requests forwarded by each switch and then to assign a forwarding state to each switch, which finally delivers the multicast traffic to the interested servers.

Figure 4-9. Sandwich Tree broadcast

Compared to the same scale of a traditional fat tree, server 1 sending a broadcast packet requires 14 duplicate packets, since each switch requires $n-1$ duplicate packets to send a packet through n ports. In Figure 4-10, we can see all 14 packet duplicates are denoted by yellow circles. In terms of broadcast and multicast traffic, the Sandwich Tree saves more bandwidth than the traditional fat tree structure by reducing the number of duplicate packets. The bigger the scale of

the DCN, the more bandwidth will be saved.

Figure 4-10. Traditional fat-tree broadcast

4.2.4 Benefits in terms of cost and power consumption

In building current DCNs, minimizing cost and power consumption is critical. Thus, in this section, we will discuss the cost and power consumption of the Sandwich Tree. Six general formulas will be developed to represent the cost and power consumption of all of three topologies: the Sandwich Tree using POXN/MPs only, the Sandwich Tree using POXN/MPs and POXN/MP-BPs, and the traditional fat tree topology. Finally, the advantages of the Sandwich Tree with POXN/MP-BPs over the other two topologies are clearly shown in tables and graphs.

4.2.4.1 Cost advantages

In order to illustrate the huge cost savings made possible by the Sandwich Tree topology, we will develop a general formula for all three topologies. In this model, the Capital Expenditure (CAPEX) per link is not adopted because of the asymmetry of the aggregation switch. Instead, the

CAPEX is calculated for the whole system worth being investigated for the same number of servers. First, we examine the main components in the fat tree topology, which comprises $(k/4)^3$ servers and $(k/2)^2$ k-port core switches in total, $k/2$ k-port aggregation switches, and $k/2$ k-port access switches existing in k pods [24]. Therefore, the total number of k-port switches is: $\left(\frac{k}{2}\right)^2 + \left(\frac{k}{2} + \frac{k}{2}\right) \times k$. Based on this, it is easy to determine the total number of line cards and switch fabrics as: $\left[\left(\frac{k}{2}\right)^2 + \left(\frac{k}{2} + \frac{k}{2}\right) \times k\right] \times k$. Many studies [17-19] show that industry-wide usage of 40 Gbps and 100 Gbps optical transponders is commercially available within datacenter. Compared to electrical transceivers, optical transponder has many merits, such as longer transmission distance, low power consumption, faster speed with cheaper price, and etc. Driven by these big advantages, optical transponder has been replacing electrical transceivers in the past few years. Therefore, we will select 1310 nm LR optical transponder for comparison, rather than 10 Gigabit Ethernet. Since each server port is equipped with one 1310 nm LR transponder to communicate with the ToR switch, the total number of 1310 nm LRs can be calculated as: $\left[\left(\frac{k}{2}\right)^2 + \left(\frac{k}{2} + \frac{k}{2}\right) \times k\right] \times k + \frac{k^3}{4}$.

Regarding the POXN/MP, each server is equipped with one tunable LR (1550 nm) transceiver and one non-tunable LR (1550 nm) transceiver. In a pod, each switch is deployed with k tunable LR (1550 nm) transceivers and k non-tunable LR (1550nm) transceivers, half of which are connected to the lower-level POXN/MPs and the other half of which are connected to the upper-level POXN/MPs. Therefore, the total number of tunable LR (1550nm) and non-tunable LR (1550nm) transceiver can be calculated as: $\left(\frac{k}{2}\right) \times k \times k + \frac{k^3}{4}$. In addition, the number of line cards and switch fabrics depends only on the number of switches deployed; thus, it is: $\left(\frac{k}{2}\right) \times k \times k$.

In contrast, POXN/MP-BP provides flexible bundled port count techniques with multiple

choices. Furthermore, it largely reduces the usage of fixed wavelength LR (1550nm) transceivers. The reason is that the multicast traffic sent by the fix-tuned transceiver follows a one-by-one behavior pattern, meaning that only one fix-tuned transceiver should be active at a time.

Assuming the number of ports bundled together is b (the same as in the POXN/MP), each server equips itself with one tunable LR (1550 nm) transceiver and one non-tunable LR (1550 nm) transceiver. Unlike the deployment on the server port, switch ports connected to the first-level POXN/MPs are equipped with bundled ports, consisting of b tunable LR (1550 nm) transceivers and one non-tunable LR (1550 nm) transceiver. Each of the remaining ports of the switch connected to the top-level POXN/MP contain one tunable LR (1550 nm) transceiver and one non-tunable LR (1550 nm) transceiver. Since there are $k/2$ subnets in one pod, the total numbers of tunable LR (1550 nm) transceivers and non-tunable LR (1550nm) transceivers on each switch are $b \times (\frac{k}{2} + \frac{k}{2})$ and $1 \times (\frac{k}{2} + \frac{k}{2})$, respectively. Then, the total numbers of tunable LR (1550 nm) transceivers and non-tunable LR (1550 nm) transceivers in the k -pod datacenter are: $(b \times k) \times \frac{k}{2b} \times k + \frac{k^3}{4}$ and $k \times \frac{k}{2b} \times k + \frac{k^3}{4}$, respectively. It is obvious that the number of fixed 1550 nm LR transceivers is reduced dramatically with the deployment of POXN/MP-BP. On the other side, the number of line cards and switch fabrics is $\left[\left(\frac{kb}{2} + \frac{k}{2b}\right) \times \left(\frac{k}{2b}\right) \times k\right]$, where $\frac{k}{2b}$ represents the number of switches in one pod and k is the number of pods. The quantity of couplers is easy to calculate: $\frac{k}{2} \times k$ (first level) + $\frac{kb}{2} \times \frac{k}{2b}$ (top level). The AWGs and WFFOCs used for bundled ports require combining and splitting more channels than normal ports; thus, their expenditures will increase. However, thanks to the maturity of the technology, this increase is linear and marginal. The quantity of both AWGs and WFFOCs for the bundled ports on the switch is $\frac{k}{2b} \times \frac{k}{2} \times k$, while the

remaining quantity of both AWGs and WFFOCs for normal ports on the server is $\frac{k^3}{4}$.

Table 4-5. Comparison of main component quantities among three topologies

Topology	Quantity of Components					
	Line-card & switch fabric	10G LR (1310nm)	10G LR (1550nm)	10G tunable LR (1550nm)	Coupler	AWG and WFFOC
Traditional fat tree	$\frac{5k^3}{4}$	$\frac{3k^3}{2}$	0	0	0	0
POXN/MP	$\frac{k^3}{2}$	0	$\frac{3k^3}{4}$	$\frac{3k^3}{4}$	k^2	$\frac{3k^3}{4}$
POXN/MP-BP	$\frac{k^3}{4} + \frac{k^3}{4b}$	0	$\frac{k^3}{2} + \frac{k^3}{4b}$	$\frac{3k^3}{4}$	$\frac{3k^2}{4}$	$\frac{k^3}{4b} + \frac{k^3}{2}$

Based on the main component quantities outlined in Table 4-5, plus their unit costs, we can develop two CAPEX formulas for two kinds of topologies. For the fat tree topology, the total cost can be represented by:

$$\left(\frac{5k^3}{4}\right) \times 450 + \left(\frac{3k^3}{2}\right) \times 200 = 862.5k^3$$

For the POXN/MP topology, the total cost is as follows:

$$\frac{k^3}{2} \times 450 + \left(\frac{3k^3}{4}\right) \times 350 + \left(\frac{3k^3}{4}\right) \times 525 + k^2 \times 480 + \left(\frac{3k^3}{4}\right) \times 40 = 911.25k^3 + 480k^2$$

For the POXN/MP-BP replacement in the fat tree topology, the total cost is as follows:

$$\left(\frac{k^3}{4} + \frac{k^3}{4b}\right) \times 450 + \left(\frac{k^3}{2} + \frac{k^3}{4b}\right) \times 350 + \left(\frac{3k^3}{4}\right) \times 525 + \frac{3k^2}{4} \times 480 +$$

$$\left(\frac{k^3}{4b} + \frac{k^3}{2}\right) \times 40 = 701.25k^3 + 210\frac{k^3}{b} + 360k^2$$

Figure 4-11. Cost comparison among the traditional fat tree topology, the POXN/MP, and the POXN/MP-BP, with different numbers of bundled ports for each bundled group

Based on the developed formulas, we develop the above Figure 4-11. We choose different pod values ($k = 4, 8, 16, 24, 32, 40$, and 48) and then calculate the corresponding number of servers which is $\frac{k^3}{4}$. The x-axis represents the number of servers in the datacenter, which ranges from only 16 to 27,648, while y-axis represents the total expenditure (in million USD) for building DCNs of different scales. POXN/MP-BP b is used to denote different cases of the number of bundled ports, and b here represents the same thing as b in the above formulas. For example, if k is 48 and b is 4, this means that four ports will be bundled together on each switch's NIC, the number of switches in each pod will be $\frac{k}{2b} = 6$, and each switch will have $4 \times \frac{k}{2} = 96$ ports connected to the 24 first-level couplers through 24 different links.

The red line representing the POXN/MP increases sharply with the increasing number of servers, reaching approximate 102 million USD. The fat-tree topology, denoted by the blue line, illustrates a total cost that increases slightly less than that of the POXN/MP, reaching around 95 million USD (thus saving about 6.5 million USD) when the datacenter accommodates 27,648 servers (48 pods). In accordance with expectations, the POXN/MP-BP2 achieves significant cost savings, costing as much as 5.1 million USD less than the fat tree topology. With the increasing number of bundled ports on the switch side, the POXN/MP-BP performs better and better. More specifically, expenditure savings can reach 12.9 million USD and 13.8 million USD for POXN/MP-BP4 and POXN/MP-BP6, respectively. Furthermore, when b increases from 2 to 6, total expenditures decrease from 90.3 million USD to 81.6 million USD. These results match expectations, since, when the number of bundled ports increases, fewer total elements are required (except for the cases of tunable LR (1550 nm) transceivers and couplers).

If we take operational expense (OPEX) into consideration, Sandwich Tree will save more cost

because of two reasons. The first one is that both POXN/MP-BP and POXN/MP are based on passive optical fabrics. In contrast, electronic switches are active devices and extra cooling equipment is required to lower the temperature to ensure it works properly. The second reason is that both POXN/MP-BP and POXN/MP are plug-and-play devices. In contrast, electronic switch is also plug-and-play but it needs to be adjusted and configured in some cases. Therefore, Sandwich Tree can save much more by reducing labor cost.

4.2.4.2 Power consumption advantages

In terms of the power consumption, the benefits of the Sandwich Tree are also clear. One big advantage of using a passive optical coupler is that it consumes zero power by nature. Furthermore, neither the AWG not the WFFOC uses any power. The only element that increases the power usage is the long-range LR (1550 nm), which uses more power than the long-range LR (1310 nm). However, the unit difference between the two is very minor, at only 0.5 W. By contrast, the fat tree topology, with its wide deployment of electronic switches, consumes far more power than the POXN/MP-BP topology.

We develop three general formulas to determine the power consumption of all three topologies based on the number of elements in the last section and the unit power consumption of each component in Table 4-6. One thing should be noticed is that the POXN/MP-BP topology can consume less power by reducing the power budget for coupler fabrics, which is demonstrated in the section 3.2. This is why the formula of the POXN/MP-BP topology contains an extra coefficient 0.67.

Table 4-6. Unit power consumption of each element

Device	Unit power consumption (W)
Switch port	12.5
LR (1310 nm) transceiver (SFP+)	1
LR (1550 nm) transceiver (SFP+)	1.5
LR (1550 nm) transceiver (tunable SFP+)	1.5
Coupler, AWG, or WFFOC	0

The power consumption of the fat tree topology can be calculated using the following formula:

$$\left(\frac{5k^3}{4}\right) \times 12.5 + \left(\frac{3k^3}{2}\right) \times 1 = 17.125k^3$$

For the POXN/MP, the power consumption can be calculated as follows:

$$\frac{k^3}{2} \times 12.5 + \left(\frac{3k^3}{4}\right) \times 1.5 + \left(\frac{3k^3}{4}\right) \times 1.5 = 8.5k^3$$

For the POXN/MP-BP, the power consumption can be calculated as follows:

$$\left(\frac{k^3}{4} + \frac{k^3}{4b}\right) \times 12.5 + \left(\frac{k^3}{2} + \frac{k^3}{4b}\right) \times 1.5 \times 0.67 + \left(\frac{3k^3}{4}\right) \times 1.5 \times 0.67 = 4.381k^3 + 3.376\frac{k^3}{b}$$

Figure 4-12. Power consumption comparison of the traditional fat tree topology, the POXN/MP, and the POXN/MP-BP, with different numbers of bundled ports for each bundled group on the switch

As can be seen from Figure 4-12, the results show that the POXN/MP-BP has a significant advantage over the traditional fat tree topology and the POXN/MP with respect to power consumption. For the fat tree topology, power consumption increases dramatically with the increase in servers, reaching approximately 1.89×10^3 KW when the datacenter has 27,648 servers. The POXN/MP uses significantly less power, consuming only 0.94×10^3 KW for a datacenter with 27,648 servers. Undoubtedly, the POXN/MP-BP has the best performance of all three topologies. It exhibits a much slower increase, when the number of servers increases by four orders of magnitude (from 16 servers to 27,648 servers). More specifically, for a large-scale datacenter containing 27,648 servers, the POXN/MP-BP6 replacement topology results in power

savings up to 1.34×10^3 KW , representing a 71% reduction in the total power consumption of the fat tree topology.

Compared with the savings in the expenditure, it is clear that the power savings of the POXN/MP-BP topology over the POXN/MP and fat tree topologies are more substantial. Thus, the POXN/MP-BP replacement methodology not only saves money, but also relieves the burden of power usage within the datacenter. Furthermore, since less power is used, more funds will be saved. At the price of 12.73 cents/KW hour [48], the power usage difference between the fat tree topology and the POXN/MP-BP6 topology results in a savings of 1.42 million USD per year, under the condition of 27,648 servers. Finally, cooling costs represent another major factor in total electricity usage, representing nearly 30% of this usage in large-scale datacenters [49]. Thus, it should be noted that, once cooling costs are taken into consideration, total saving will be greater than 1.46 million USD.

Chapter 5 Numerical Results

Thus far, we have explored the general concepts of upgrading POXN/MPs to the new POXN/MP-BP design element, building POCubes using POXN/MPs, and building Sandwich Trees using POXN/MPs and POXN/MP-BPs in a fair amount of detail. We have also investigated the routing schemes, hardware costs, and power consumptions of proposed architectures.

In this chapter, we will set up experiments at the algorithm, device, and network levels to theoretically assess our simulation models. We will first examine the algorithm efficiency of the unicast planes of POXN/MP-BPs under different scenarios. Following this simulation, we will test whether a single POXN/MP-BP works at the device level in a real network scenario using an OPNET modeler. We will simulate the complete protocol, including the discovery phase and the data transfer phase, in order to demonstrate the practicability and correctness of the POXN/MP-BP. Finally, we will simulate two new proposed topologies—the POCube and the Sandwich Tree—at the network level. All the results are shown with a 95% confidence interval.

5.1 Algorithm-level Simulation

The algorithm efficiency is defined as the ratio of the mean transmission time of the unicast traffic for a transmitter to the period of one unicast data transfer cycle. This simulation is implemented in C. For simplicity, since the goal of the simulation is to test whether the new algorithm exhibits better performance for the unicast plane of the POXN/MP-BP, we simulate only the data transfer phase, which lasts from the packet's arrival at the transmitter to its arrival at the receiver. In cloud datacenters, 40 to 90% of traffic is destined to leave the rack and traverse the interconnections of the DCN [50].

Figure 5-1. The ratio of outgoing traffic to local traffic in 10 datacenters [50]

In one previous study [50], 10 different categories of datacenters were investigated in terms of the ratio between the traffic spread across other racks and the traffic traversing within the rack. To accomplish this, the study examined 10 datacenters, including education DCNs (EDU1 and EDU2), private DCNs (PRV1 and PRV2), and commercial DCNs (CLD1, CLD 2, CLD3, CLD4 and CLD5).

In Figure 5-1, the x-axis represents the 10 different datacenters, and the y-axis denotes the traffic percentages generated by the servers. The red rectangle illustrates the traffic spread across the rack, while the blue one illustrates the traffic staying within the rack. It is clear that the traffic generated by the servers in the commercial DCNs tends to remain within the rack; however, at least 60% of the traffic in the campus and enterprise DCNs leaves the rack. The main reason for this is that the commercial datacenters (CLDs) must address several dependent services and co-located applications, such as MapReduce jobs, that significantly change traffic patterns [51].

Compared to CLDs, university and enterprise datacenters handle more web services, video application downloads [52], and file server access attempts. Therefore, it is necessary to test different proportions of these two kinds of traffic in our simulation.

The setup parameters are based on the existing EPON technology and are consistent with the HEDAP and the MCDAP. We will simulate a 48-port POXN/MP-BP, since it is common to see 48-port electronic switches in current DCNs. In the setup environment, each port works at a 10 Gb/s line rate. Because this experiment is simulated to demonstrate the efficiency of the algorithm, for simplicity, we can assume that the number of packets in the dedicated unicast queue follows a uniform distribution. Real network environment simulations will be discussed in the next subsection. Constant tuning equals 5 μ s of transmission time, which equals the transmission time of six packets. Every packet has an equal size of 1024 bytes. With respect to accuracy, each simulation result is the average value of 2000 cycles. Each transmission cycle is run on a burst-by-burst basis.

Before the simulation results are discussed, the upper bound for algorithm efficiency is generalized. The ideal result is calculated as the upper bound for algorithm efficiency under the following assumptions: Each transmitter has the same amount of unicast traffic to all receivers during a data transfer cycle; between any two consecutive data bursts, there is only constant tuning time; and each transmitter starts and finishes its data burst at the same time. Thus, for the unbiased traffic scenario, the upper bound efficiency can be represented as follows [9]:

$$E = \frac{(N - 1) \times T_{ij}^Q}{(N - 1) \times T_{ij}^Q + (N - 2) \times T^T}$$

For the biased traffic scenario, the simulation setup parameters are much closer to those found

in a real datacenter, in which bundled ports in the same group do not communicate with one another. Unlike in the MCDAP, in this scenario, not all transmitters can complete at the same time, since variations in queue length exist for intra-rack and inter-rack traffic. Therefore, for the ideal situation, all normal transmitters finish at the same time, followed by the simultaneous completion of all the bundled transmitters. Assuming that half of the ports are individual ports and the other half are bundled ports (further divided into M groups, each of which has B bundled ports), the upper bound efficiency can be obtained as follows:

$$E = \frac{\frac{N}{2} \times \left(\frac{N}{2} - 1\right) \times T_{ij}^{loc} + \left(\frac{N}{2}\right)^2 \times T_{ij}^{ext} + \frac{N}{2} \times (N - B) \times T_{ij}^{ext}}{N \times \max\{[(N - B) \times T_{ij}^{ext} + (N - B - 1) \times T^T], [(\frac{N}{2} - 1) \times T_{ij}^{loc} + \frac{N}{2} \times T_{ij}^{ext} + (N - 2) \times T^T]\}}$$

All of the simulation results in this section are shown with a 95% confidence interval. Since we simulate 2000 data transfer cycles per observation, this yields extremely narrow confidence interval for all simulation results in this section.

5.1.1 Different proportions of biased traffic

In this scenario, the traffic generated by the end nodes either leaves the rack or stays within the rack according to certain proportions. Accordingly, for simplicity, the unicast queue lengths of both the local traffic and the external traffic follow uniform distributions. A set of proportions with the same standard deviation for mean queue length is used to simulate a set of proportions for intra-rack traffic and inter-rack traffic. The queue lengths for the external and the local traffic are selected as follows: [40, 80] and [0, 40], [40, 80] and [5, 45], [40, 80] and [10, 50], [40, 80] and [15, 55], and [40, 80] and [20, 60], respectively. The corresponding proportions of inter-rack traffic to intra-rack traffic are: 3:1, 2.4:1, 2:1, 1.71:1, and 1.5:1.

From Figure 5-2, we can clearly see that the new algorithm is significantly superior to the old

LQF algorithm. The x-axis represents the proportion of inter-rack traffic to intra-rack traffic, while the y-axis shows the algorithm's efficiency. The green line is the upper bound of the algorithm's efficiency. The red line illustrates the new algorithm, while the blue line below represents the old LQF algorithm. As the proportion varies from 3:1 to 1.5:1, all of the relative increases in algorithm efficiency are 4.7% or greater, ranging as high as 6.03% for the 2:1 ratio. Because of the flexibility provided by the introduction of the bundled ports, the transmitters prefer to send packets to the bundled ports first. In cases of lower proportions of intra-rack traffic, type 2 mismatches are substantially reduced. This is why the new LQF/BP algorithm exhibits better performance in terms of efficiency.

Figure 5-2. Different inter-rack/intra-rack proportions for biased traffic

5.1.2 Different bundled port numbers for biased traffic

In this scenario, the mean and standard deviations of the queue length are 60 and 11.54, respectively. The results in terms of efficiency are collected when the number of bundled ports is 2, 3, 4, 6, 8, and 12. We use the same three colors to represent the upper bound of efficiency, the LQF/BP, and the LQF. Figure 5-3 shows that the new algorithm exhibits a steadily increasing advantage over the old algorithm as the bundled port number increases from two to six. When the bundled port number reaches six, the difference between the two is at its greatest: about 4.7%. This difference stems from the more frequent occurrence of scenarios 1, 2, and 3, as described in section 3.4.3, which gradually magnifies the benefits. After bundled port number 6, the difference starts to shrink; this is in accordance with expectations. Since the preference for traffic involving bundled ports leads to a global imbalances, other individual ports tend to dominate the last transmission completions, thus increasing type 1 mismatches.

Figure 5-3. Comparison of LQF and LQF/BP algorithm efficiencies for biased traffic with different numbers of bundled LQF/BP ports

5.1.3 Mean of the unicast queue length for unbiased traffic

Figure 5-4 shows how the algorithm efficiency varies under the conditions of unbiased traffic and a fixed standard deviation as the mean queue length increases from 45 to 65. For unbiased traffic, all queue lengths in the transmitter follow the same distribution, meaning that unbiased traffic has a 1:1 proportion of local traffic and external traffic. This change means that the LQF/BP has a slight advantage over the LQF for different sets of mean queue lengths, achieving as much as a 2.5% relative difference at a value of 45. This is because, even though type 2 mismatches and

certain constant tuning times can be avoided, the last transmission is very likely to be dominated by local traffic. If local traffic has the same distribution as external traffic, type 1 mismatches will certainly increase. In spite of this, the new algorithm is still closer to the upper bound than the old algorithm.

Figure 5-4. Different mean queue lengths for unbiased traffic

5.1.4 Variation of the unicast queue length for unbiased traffic

Figure 5-5 illustrates a mean queue length of 65 and a standard deviation ranging from 8.66 to 20.20. As in the last scenario, the LQF/BP slightly outperforms the LQF. The larger the variation in the unicast queue length is, the lower the algorithm efficiency is. It is important to note that

when the standard deviation is 8.66, the efficiency is very close to the upper bound, with only an approximately 5% difference.

Figure 5-5. Different standard deviations in queue length for unbiased traffic

5.1.5 Different bundled port number for unbiased traffic

In this scenario, the mean queue length is 55 and the deviation is 14.43. The total number of bundled ports is fixed, but the number of bundled ports in a group varies from 2 to 12 in increments of two. Figure 5-6 shows that the LQF follows a straight line, unaffected by the bundled port number. The LQF/BP exhibits better performance than the LQF, with the relative difference increasing from 1.6 to 2.3% as the bundled port count increases from 2 to 12. Under the condition

of unbiased traffic, the larger the number of bundled ports is in a group, the more frequently the longer queue length will be transmitted by the bundled port transmitters and received by the bundled port receiver. However, the normal transmitter will experience a longer transmission time not only when it sends traffic to the bundled ports, but also when it sends traffic to the other normal ports. Even though a certain amount of constant tuning time is avoided, the end of the transmission is still dominated by the normal transmitter. This explains why the variations in efficiency are not particularly outstanding, given the increase in bundled port numbers.

Figure 5-6. Comparison of LQF and LQF/BP algorithm efficiencies for unbiased traffic with different numbers of bundled LQF/BP ports

5.2 Device-level and Network-level Simulations

In order to prove the practicability and validity of the POXN/MP-BP, an six-port POXN/MP-BP is first simulated at the device level. Then, we evaluate the network performance of a Sandwich Tree with a combination of POXN/MP-BPs and POXN/MPs and the network performance of a Sandwich Tree with POXN/MPs only at the network level over two steps. The first step is to build a two-level network model with one level of POXN/MPs and one level of switches. The second step is to build a three-level network model with one level of POXN/MP-BPs, one level of switches, and one level of POXN/MPs. All simulations are simulated in the OPNET modeler.

5.2.1 Device-level simulation

As with the simulation setup parameters in MCDAP [9], we assume that all transmitters (including the fixed transmitter and the tunable transmitter) work at the speed of 10 Gbps. All control messages are 128 bytes, packet sizes follow an exponential distribution with a mean size of 1024 bytes, the data transfer phase lasts $144.96 \mu s$, and a discovery phase is triggered every 20000 cycles. The simulation topology is depicted in Figure 5-7.

Figure 5-7. Simulation topology of a six-port POXN/MP-BP

The overhead consists of control message overhead and hardware limitations, which are shown in Table 5-1. The upper bounds of the MCBDAp can be calculated as follows: $1-12/144.96 = 0.917$.

Table 5-1. Time periods for different kinds of overhead [9]

Operation	Time period
Inter-port guard interval (including laser off and on, automatic gain control [AGC], clock data recovery [CDR], and code-group alignment intervals)	$2\ \mu s$
Transmission time for control messages T^D	$0.1014\ \mu s$
Inter-port processing time	$10\ ns$
Inter-frame gap time	$9.6\ ns$
Worst-case propagation delay from port to coupler	$5\ \mu s$
Tunable transmitter tuning time	$4.916\ \mu s$
Per-frame service time	$0.5092\ \mu s$

As we can see from Figure 5-8, the MCBDAp exhibits better performance than the MCDAP. The system's maximum bandwidth efficiency is about 0.81, which is very close to the value of the upper bound for the MCBDAp. For instance, when the load ρ is 0.81, the throughput for the MCBDAp can reach 8.09 Gbps, compared with 7.56 Gbps for the MCDAP, representing a relative improvement of approximately 7%. As a result, an eight-port MCBDAp system can achieve 64.72 Gbps aggregated throughput, compared with 60.48 Gbps throughput for the MCDAP. This benefit will increase as more ports become involved in the system. This result also matches expectations, since the efficiency improvement algorithm in section 3.4.3 suggested that, at some point, bundled ports transmitting and receiving packets for one another greatly reduces the mean packet queueing delay, allowing more packets to be delivered during the data transfer phase.

Figure 5-8. Throughput performance of the MCBDAP with an increase in load

In addition, Figure 5-9 illustrates how the increase in load to a transmitter affects the mean packet delay in a single POXN/MP-BP. The mean packet delay shows a very steady and slow increase when the value of the load varies between 0 and 0.7, after which it experiences a substantial increase, especially when it reaches the load limit of 0.81. Compared to the POXN/MP, the POXN/MP-BP exhibits a flatter increasing trend regarding the mean packet delay. This is because a certain amount of overhead can be avoided through the benefits of the new protocol.

Figure 5-9. Mean packet delays for the MCBDAP with different offered loads ρ

5.2.2 Network-level simulation

First, a two-level POCube network model with POXN/MPs is simulated. Second, a two-tier network model with POXN/MP-BPs is simulated in order to prove the network performance (e.g., aggregate throughput) of the MCBDAP at the network level. This performance is compared to that of the POXN/MP structure. Second, a three-tier network model with a combination of POXN/MP-BPs and POXN/MPs is simulated to prove that POXN/MP-BPs and POXN/MPs are able to operate well in a multiple-tier network and to cooperate efficiently with traditional electronic switches.

In the simulation, only 16 servers are involved for both of two kinds of architectures for simplicity. In theory, if we use 81-port POXN/MP to build a three-level POCube, POCube can be scaled to 531441 servers ($N = n^{k+1}$). If we use 64-port (41 port connect 41 servers and the other B -in-1 bundled ports connect higher level switches) POXN/MP-BP and 81-port POXN/MP to build

a three-level Sandwich Tree, it can support 136161 servers ($41 \times 41 \times 81$). Through the simulation, if we can demonstrate that the traffic can be balanced over links between servers and different levels of POXN/MPs in POCube and the network performance shows graceful degradation with the increase of the number of levels in Sandwich Tree, more servers or even more levels will not produce significant impact on the network performance.

5.2.2.1 Two-level POCube modeling

The POCube simulation topology is shown in Figure 5-10. Each switch is equipped with two pairs of unicast and multicast transceivers in order to transmit packets in a two-level POCube topology. $\text{POCube}_0 0$, $\text{POCube}_0 1$, $\text{POCube}_0 2$, and $\text{POCube}_0 3$ represent the first level, while $\text{POCube}_1 0$ represents the second level. The frame size has an Ethernet format that varies between 64 and 1518 bytes and follows a truncated geometric distribution. Each frame arriving at each server follows a Poisson distribution. The maximum cycle period is $144.96 \mu\text{s}$ for both the HEDAP and the MCDAP. Due to the overhead for the HEDAP and the MCDAP, the maximum efficiencies for the multicast and unicast planes are 0.917 and 0.229, respectively.

Figure 5-10. Simulation topology of a two-level POCube with 16 servers

The all-to-all traffic follows a uniform distribution. If a packet needs to be relayed, it randomly chooses one path from two parallel paths. For example, let us assume that server 1 wants to send a packet to server 6. The first option is to transfer the packet to server 5 (denoted by 10), first through the second-level $\text{POCube}_1 0$ and then through a transfer to server 6. The second option is to transfer the packet to server 2 (denoted by 01), first through the first-level $\text{POCube}_0 0$ and then through a transfer to server 6. Previous results [9] have shown that variations from 1 to 10% in the proportion of multicast traffic to total traffic (i.e., both multicast and unicast traffic) only affect the mean packet delay with 0.02 ms difference. Thus, in this simulation, we pick only one set of parameters as the simulation setup. The multicast and unicast portions are set to 1 and 99%, respectively.

We increase the load offered to each port and study when the system is stable and how the three kinds of delays are affected. More specifically, intra-POXN/MP traffic means that a packet does not need to be relayed by a server. For example, server 1 sends traffic to server 2, 3, and 4 within POCube_0 or sends traffic to server 5, 9, and 13 within POCube_1 , both of which belong to intra-POXN/MP traffic. Inter-POXN/MP traffic means that a packet must be relayed by a server. For example, if server 1 sends traffic to server 6, the packet will be relayed once by server 2 or server 5. An end-to-end delay means that all traffic is transmitted across the entire network.

Figure 5-11. Delay from source server to destination server going via one POXN/MP; delay from source server to destination server going via two POXN/MPs; and end-to-end delay for a POCube topology with different loads offered to the system

We evaluate all the uplink and downlink bandwidths in the POCube simulation. For the unicast plane, the average uplink bandwidth is 7.84 Gbps, with a standard deviation of 0.22 Gbps, and average downlink bandwidth is also 7.84 Gbps, with a standard deviation of 0.21 Gbps. Since the structure is symmetric and adopts a uniform all-to-all traffic pattern, no link will become the dominant bottleneck link.

The overhead for the unicast plane consists of inter-port guard intervals and twice worst-case propagation delays from the coupler fabric to a transmitter, which is calculated as $12 \mu\text{s}$, according to Table 5-1. Thus, the theoretical maximum achievable throughput of the MCDAP is $1 - 12/144.96 = 0.917$, with an approximate relative difference of 12.7%. The simulation results suggest that the bandwidth for uplinks and downlinks carrying multicast traffic is between 1.92 Gbps and 2.01 Gbps. The overhead for the HEDAP consists of an inter-port guard interval, the transmission delay of REQUEST message T^D , twice 1-km propagation delays from the coupler fabric to a transmitter, and a constant processing time T^C [9]. According to Table 5-1, for the traffic reaching the destination through one POXN/MP, the theoretical maximum achievable normalized throughput is $1 - 12.1114/144.96 = 0.916$. For the traffic going through two POXN/MPs, it is $1 - (12 + 12 + 4)/144.96 = 0.807$ (where $12 \mu\text{s}$ is the overhead for the POXN/MP-BP and $4 \mu\text{s}$ is the switching latency). In the simulation, the two kinds of traffic are equally divided, so the theoretical maximum achievable normalized throughput is $(0.916 + 0.807)/2 = 0.862$. Since ports can only send sequentially in the HEDAP, the maximum per-port efficiency is $0.916/4 = 0.229$. Thus, compared to the theoretical value, the simulation result has an approximate 12.2% relative difference. The normalized throughputs for the unicast and multicast planes can achieve 0.784 and 0.196, respectively. Compared with the theoretical value of system throughput, the simulation result has an approximate 10.2% relative difference.

5.2.2.2 Two-tier Sandwich Tree modeling

We will illustrate how to integrate POXN/MPs and POXN/MP-BPs into Sandwich Trees in detail. In this subsection, we will simulate a POXN/MP-BP in a two-tier Sandwich Tree. The next subsection will show how to scale up continuously with one more level of POXN/MPs in a three-tier Sandwich Tree.

The simulation topology is shown in Figure 5-12. To compare performance, the same scale of POXN/MPs is simulated, as shown in Figure 5-13. Since it is too complicated and impractical to simulate a 48-port POXN/MP-BP on a packet-by-packet basis, we choose a simulation topology in which two POXN/MP-BPs form subnets 1 (including server 1 to server 4) and 2 (including server 5 to server 8), respectively. Each of the two POXN/MP-BPs connects a group of bundled ports on each switch, while connecting to four normal ports on four servers. Packets arriving at eight servers still follow a Poisson distribution. According to the configuration of the Cisco Nexus 3000 series switch [53], the switching time is proportional to the packet size and is less than 4×10^{-6} s.

We use an all-to-all traffic pattern, where a server distributes half of its traffic to servers connected to a different POXN/MP-BP. Further, this portion of traffic is uniformly distributed among these servers. In this setting, packets are evenly distributed over links based on a 4-tuple. This achieves the equivalent effect of employing Equal Cost Multi-Path (ECMP) at a flow or even finer packet granularity. Thus, we believe that the results here also hold for general ECMP-like routing, such as VLB [54], which is widely used in DCNs. Based on the description in section 4.2.3, the server follows the IP address format 10.pod.coupler.ID, and the switch follows the format 10.pod.switch.1. The last two quad-dotted pieces of information—‘coupler,’ ‘ID,’ ‘switch,’ and ‘1’—are added into the control messages for each server and switch, as relevant. Since, in this

scenario, no inter-pod traffic is introduced, the 'pod' information is not needed.

Figure 5-12. Simulation topology of POXN/MP-BP with eight servers

Figure 5-13. Simulation topology of a POXN/MP with eight servers

End-to-end (ETE) delay performance must be analyzed, since this is an important factor that can measure the quality of service (QoS) of a network. We measure only packet queueing delays, switching delays, and ETE delays.

Figure 5-14. Delay from source server to destination server going via one POXN/MP-BP; delay from source server to destination server going via two POXN/MP-BPs; and end-to-end delay for a two-tier Sandwich Tree topology (shown in Figure 5-12) with different loads offered to the system

Figure 5-14 shows three categories of delays: internal traffic delays, external traffic delays, and total traffic ETE delays. Internal traffic delays refer to traffic transmitted among servers 1, 2,

3, and 4 or among servers 5, 6, 7, and 8. External traffic refers to traffic relayed by the switch. For example, server 1 might send traffic to server 5, 6, 7, or 8, which would need to be relayed by switch 1 or 2 (shown in Figure 5-12). ETE traffic refers to all traffic (including internal and external traffic) transmitted across the entire network.

All three kinds of delays show an increasing trend with the increase in loads offered to each server. As can be seen in Figure 5-14, the closer they are to the load limit of 0.761, the faster their corresponding delays increase. As the load changes from 0.1 to 0.761, internal traffic always experiences at least average delays, while external traffic suffers the longest average delays. This result can be expected, since internal traffic does not need to undergo a switch before reaching its destination servers (e.g., server 1 sends traffic to server 2, 3, or 4 in Figure 5-13). By contrast, external traffic will experience more delays, including queueing delays, switching delays, and propagation delays due to switching to other POXNs (e.g., server 1 sends traffic to servers 5, 6, 7, or 8 in Figure 5-13). In addition, ETE delays illustrate the overall performance of all internal and external traffic in the network. $4.8 \times 10^{-4}s$

Figure 5-15. Delay from source server to destination server going via two POXN/MP-BPs include three main kinds of delays, which are server queueing delay, switch input port queueing delay, and switch output port queueing delay

To verify the simulation results, we measure the three kinds of delays that dominate ETE delays: server queueing delays, switch queueing delays, and switch delays (Figure 5-15). Propagation delays are not drawn in the figure, since these are represented by a constant value. Furthermore, twice the propagation delay plus the server queueing delay in Figure 5-15 is equal to the internal traffic delay in Figure 5-14. It is interesting to note that, near the load limit in Figure 5-15, both the server queueing delay and the ETE delay show drastic increases, while the switch queueing delay still retains its slow, increasing trend. This is because of our traffic pattern setup,

in which half of the traffic generated at each server will traverse the switch to reach its destination, resulting in the switch having a lower load than the source server.

To demonstrate our thinking, we also measure the average link utilization for all kinds of links, as shown in Figure 5-16, including two uplinks and two downlinks. The uplink from the source server to the POXN/MP-BP carries external traffic sent from the source server to the bundled port on the switch and carries internal traffic from source server to the other servers connected by the same POXN/MP-BP. These two kinds of traffic travel through the POXN/MP-BP, and then external traffic is carried by the uplink from the POXN/MP-BP to the bundled port. The link from the bundled port to the POXN/MP-BP carries the external traffic already relayed by the switch, which will reach its destination after traveling through another POXN/MP-BP. The link from the POXN/MP-BP to the normal port carries two kinds of traffic, including internal traffic sent from source servers connected by the same POXN/MP-BP and external traffic sent from other source servers connected by other POXN/MP-BPs. This is why the green and yellow lines have nearly double the link utilization of the blue and red lines.

Based on the measurements and analyses of the packet delay for these three different kinds of traffic, the main components of end-to-end delay, and the link utilization for the four different kinds of links, we can conclude that the introduction of one more tier in the new two-tier Sandwich Tree structure does not become the bottleneck, while the source server reaches its system limit with the increasing load.

Figure 5-16. All four kinds of link utilizations with different loads offered to each source server

Per our expectations, the switch side link utilization is nearly half of that of the source server utilization. This may explain why queueing delays on the switch maintain a slow increase while the ETE and queueing delays on the server show drastic increases in Figure 5-15. Compared with the queueing delay for internal traffic, external traffic experiences less packet delay when it traverses another POXN/MP-BP. This further explains why, in Figure 5-14, the external traffic delay is not double that of the internal traffic with the introduction of another tier of POXN/MPs.

Finally, we measure aggregated throughput as the sum of the traffic transfer rate through servers 1 through 8, which reaches about 60.88 Gbps for the unicast plane of the POXN/MP-BP. For the internal traffic, the per-port maximum efficiency for the unicast plane is still 0.917. By

contrast, for the external traffic, the per-port maximum efficiency for the unicast plane is 1-
 $(12+12+4)/144.96 = 0.807$ (12 μ s is the overhead for the POXN/MP-BP, and 4 μ s is the switching
latency). Since each server generates traffic that has an equal likelihood of being internal or
external traffic, the overall per-port maximum efficiency for the unicast plane is 0.862, according
to a weighted arithmetic mean.

Figure 5-17 depicts the difference between the Sandwich Tree built with POXN/MP-BPs and
that built with POXN/MPs in terms of aggregate throughput as a percentage of ideal aggregate
throughput (i.e., normalized aggregate throughput). If all terminals are capable of sending traffic
at a full rate, the value will be 1. From the simulation results, we see that the two-tier POXN/MP-
BP topology achieves a higher normalized throughput of 0.76.

Figure 5-17. Normalized aggregate throughput for a two-tier POXN/MP and a two-tier POXN/MP-

5.2.2.3 Three-tier Sandwich Tree modeling

Based on the last subsection, we add a core layer on the top of two-tier Sandwich Tree to create a three-tier Sandwich Tree architecture in order to demonstrate that POXN/MP-BP and POXN/MP can alternately substitute switches and operate efficiently and independently in a scaled-down network.

Accordingly, to build this two-pod, three-tier Sandwich Tree structure, we introduce another pod containing eight more servers. The simulation topology is depicted in Figure 5-18. In each pod, each switch uses its lower ports to connect to eight servers through the POXN/MP-BP interconnection, while connecting with the other three switches through the POXN/MP interconnection. We compare this model with the three-tier Sandwich Tree structure built using switches and POXN/MPs only (without POXN/MP-BPs), which is shown in Figure 5-19.

Figure 5-18. The three-tier simulation topology is a two-pod network composed of four POXN/MP-BPs, two POXN/MPs, and four switches

Figure 5-19. The three-tier simulation topology is a two-pod network composed of six POXN/MPs and eight switches

All the settings are the same as those in the two-tier simulation. We still measure average intra-pod traffic delay, inter-pod traffic delay, and total traffic ETE delay. In the first scenario, we evaluate unbiased traffic, which means that each server has a 1:1 proportion of inter-pod and intra-pod traffic to send.

Figure 5-20. Delay from source server to destination server going via one first-level POXN/MP and two first-level POXN/MP-BPs; delay from source server to destination server going via two first-level POXN/MP-BPs and one top-level POXN/MP; and end-to-end delay in a three-tier Sandwich Tree topology with different loads offered to the system

Figure 5-20 reveals that the three-tier network built through the combination of POXN/MP-BPs and POXN/MPs exhibits nearly the same increasing trend for all three kinds of delays in the network. It is good to know that the introduction of another layer of POXN/MP does not have a great impact on network latency. At the same load of 0.72, the average inter-pod traffic delay only increases by approximately 1.2×10^{-4} s compared to that of the two-tier Sandwich Tree. The path length difference shows that this increase is composed of three components: one queueing delay

in the switch, one switch latency, and one propagation delay. From the simulation results, we can determine that top-level POXN/MPs connected by upper switch ports yield an average throughput of 3.57 Gbps and a queueing delay of around 7.5×10^{-5} s. First, the average throughput of 3.57 Gbps means that half of the traffic is external. This agrees with our traffic pattern setup. Furthermore, the 7.5×10^{-5} s queueing delay agrees with the simulation results for the mean packet delay for the POXN/MP when the load is 0.36 at the device level [9]. Since upper switch ports run another, independent protocol for the POXN/MP, matching values are expected. Compared to the external traffic in the two-tier Sandwich Tree, in addition to the extra queueing delay at the top-level POXN/MPs, inter-pod traffic will experience two more propagation delays between the two pods through the POXN/MP and one more switching latency in the switch within the destination pod. These propagation and average switching delays are 10^{-5} s and 4.45×10^{-5} s, respectively. The sum of the three delays agrees with the increase in inter-pod traffic delays (1.2×10^{-4} s), further demonstrating the validity of our results.

In terms of normalized throughput, the addition of the top tier only decreases the performance by about only 3.2%. In comparison with the POXN/MP-only architecture, the Sandwich Tree containing both POXN/MP-BPs and POXN/MPs results in a higher normalized throughput, achieving a throughput of about 0.717, with a relative percentage difference of around 5.7.

For the internal traffic, the per-port maximum efficiency for the unicast plane is still 0.917. By contrast, for the inter-pod traffic, the per-port maximum efficiency for the unicast plane is $1 - (12 + 12 + 12 + 4 + 4) / 144.96 = 0.696$ ($12 \mu s$ is the overhead for the POXN/MP-BP, and $4 \mu s$ is the switching latency). Since each server generates traffic that has an equal probability of being inter-pod or intra-pod traffic, the overall per-port maximum efficiency for the unicast plane is 0.779, according to the weighted arithmetic mean. Compared with this value, the simulation results yield

a relative difference of about 7.9%.

In the second scenario, we evaluate another traffic pattern: that of biased traffic. This means that the servers have more rack-level shuffle traffic than internal traffic. This kind of situation is commonly seen in campus and enterprise datacenters. One representative application is that of VM migration, which happens when network operators are trying to balance the load among racks [41]. In this scenario, each server generates biased traffic following a series of proportions (2:1, 3:1, 5:1, and 10:1) for inter-pod traffic and intra-pod traffic.

As we can see from Figure 5-21, all of the traffic delays exhibit more rapid increases than the unbiased traffic in scenario 1. This is because the transmitters that finish sending their unicast traffic will suffer longer idle times due to the unbalanced traffic before the next cycle of data transfers, thus lowering the utilization of bandwidth and increasing the server queueing delay. In addition, ETE delays are very close to inter-pod traffic delays, since an overwhelming majority of traffic flows tend to be delivered from the source pod to other pods. Therefore, inter-pod traffic delays dominate the ETE delays for all kinds of traffic.

Figure 5-21. Delay from the source server to the destination server via one first-level POXN/MP-BP and two first-level POXN/MP-BPs; delay from the source server to the destination server via two first-level POXN/MP-BPs and one top-level POXN/MP; and ETE delay with different loads under a biased traffic scenario with a proportion of 10:1

In accordance to our expectations, the combination of POXN/MP-BPs and POXN/MPs outperforms the architecture consisting of POXN/MPs only, since the former has advantages in terms of its protocol's algorithm efficiency (details in 3.4.3), which significantly reduce the amount of idle time required in the unicast traffic transfer phase.

Figure 5-22. Comparison of the normalized aggregate throughputs for different proportions of inter-pod and intra-pod traffic for a combination of POXN/MPs and POXN/MP-BPs and POXN/MPs only in a three-tier Sandwich Tree structure

Figure 5-22 shows that a Sandwich Tree using POXN/MP-BPs and POXN/MPs achieves a higher normalized aggregate throughput than a Sandwich Tree using POXN/MPs only under different proportions of inter-pod and intra-pod traffic, especially for biased traffic. At a proportion of 10:1, the mixed topology still has a 0.64 normalized aggregate throughput, in contrast with a 0.56 throughput for the POXN/MP-only architecture. It is important to note that, in the first case, the 1:1 proportion is equal to the first simulation scenario (i.e., unbiased traffic). We can conclude that the introduction of POXN/MP-BPs is better able to support a scaled-down network with

multiple tiers than the POXN/MP-only solution. Furthermore, the higher traffic proportion is, the better the Sandwich Tree using POXN/MP-BPs and POXN/MPs will perform.

5.2.3 Performance comparison

In the last two sections, we have simulated POXN/MP-BPs at the device level, two-level POCube using POXN/MPs, two-tier Sandwich Trees using POXN/MP-BPs, two-tier Sandwich Trees using POXN/MPs, three-tier Sandwich Trees using POXN/MP-BPs and POXN/MPs, and three-tier Sandwich Tree using POXN/MPs only. Regarding packet delay and system throughput, all show decent performance.

In this section, we will summarize the performances of the various systems and compare them with those of existing DCNs under the conditions of a 16-server network in terms of cost, routing scheme, normalized system throughput, and broadcast traffic.

Table 5-2. Different structure with 16 servers

	Routing mechanism	Cost (USD)	Normalized system throughput	Broadcast
Sandwich Tree	IP routing	59.8K	0.72	Switches need three duplicate copies of an original packet
POCube	Source routing	44.8K	0.78	Servers need four duplicate copies of an original packet
BCube	Source routing	36K	0.75 (from BCube testbed [23])	Servers need 14 duplicate copies of an original packet
Fat tree	IP routing	63.2K	0.75 (from fat tree testbed [16])	Switches need 14 duplicate copies of an original packet
Based on burst-mode (C-through)	VLAN-based routing	>45.6K (Depends on which tree structure is used for the electrical network)	Depends on the load balance between the electrical and the optical network	Worst case for this structure, requiring 15 reconfiguration periods

Chapter 6 Conclusion and Future Work

The work described in this thesis has addressed the issue of building a cost- and power-efficient DCN capable of handling dynamic traffic patterns. Many designs are focused on building new DCNs using electronic switches and active optical switches. However, the former of these is inefficient in either cost or power consumption, while the latter suffers slow configuration times and high hardware costs. Furthermore, both electronic switches and optical switches are inefficient by nature when transmitting multicast and broadcast traffic, which prevents them from adapting to dynamic DCN traffic patterns.

In this thesis, we propose one approach to integrate low-cost and power-efficient POXN/MP elements into a recursively defined structure called the POCube. In this structure, two approaches to building the same large-scale datacenter are compared, and the results show that building the POCube with higher POXN/MP port counts and fewer levels is a decent choice with regard to the trade-off among cost, power consumption, average path length, and aggregated bottleneck throughput.

We also propose a second approach, called Sandwich Tree. The introduction of POXN/MP-BPs allows the structure to handle a certain number of switches with different types of ports that facilitate traffic aggregation. More importantly, the Sandwich Tree using POXN/MP-BPs topologies significantly reduce hardware costs and power consumption compared to the Sandwich Tree using POXN/MPs topologies and traditional fat tree topologies. In addition, a two-level routing table is leveraged for use with the new MCBDAF in order to deliver a packet from server to server in a three-tier Sandwich Tree. Furthermore, a new algorithm is proposed for the MCBDAF, which further improves the channel efficiency by reducing tuning and mismatch time. The new algorithm outperforms the previous algorithm in all scenarios, especially in cases of

biased traffic.

Our simulation results demonstrate that the Sandwich Tree has better network performance than the Sandwich Tree using POXN/MPs only in both two-tier and three-tier networks in terms of ETE delays and normalized throughput for unbiased and biased traffic.

Regarding to the future work, a testbed can be built to implement our proposed DCNs using hardware when the budget is allowed. We have already evaluated the performance of two proposed DCNs in terms of link utilization, end-to-end delay and normalized system throughput in the simulation environment. To further demonstrate, it is necessary to measure the network performance again in the testbed, because it will directly reflect the results in a real network environment.

Appendix

In this thesis, we adopt the 95% confidence interval indicating if interval estimates are made on each occasion from the same population, the intervals will contain the true population parameter in 95 %.

We keep each sample independent. More specifically, in the algorithm simulation, we generate different series of data in a row by mixing a certain number of garbage values between any two series. In the device-level and network-level simulation, we use different simulation seeds provided by the OPNET simulation modeler.

Our observation count is 16 for all kinds of simulations and 95% confidence interval for the mean of a normal population is $\bar{x} \pm 1.96 \times \frac{\sigma}{\sqrt{16}}$, where 1.96 is the critical value, $\bar{x}$ is the sample mean, and σ is the standard deviation.

For the device-level and network-level, we can see the margin of error is increasing when the load offered to the system is increasing because the variation will increase when the amount of packets are accumulated increasingly at each port. In contrast, the link utilization does not vary dramatically with the increasing load. The link utilization is calculated by the ratio of average of link throughput to the link capacity over a time period. Since link throughput is already measured over the elapsed time interval in the simulation, the link utilization is averaged again based on the values of link throughput. This leads to the confidence interval shows a slow increase without much variance. To verify the simulation results, we calculate theoretical values of 95% confidence interval for Poisson distribution ($\lambda \pm 1.96 \times \sqrt{\frac{\lambda}{16}}$, where λ is the mean and 16 is the observation count) and it is in accordance with the simulation results.

References

- [1] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Holzle, S. Stuart, and A. Vahdat, "Jupiter Rising: A Decade of Clos topologies and centralized control in google's datacenter network," in *Proc. ACM Conf. Special Interest Group Data Communications*, Aug. 2015, pp. 183–197.
- [2] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Toward Dark Silicon in Servers," in *IEEE Micro*, vol. 31, no. 4, 2011, pp. 6–15.
- [3] Mark Horowitz, E. Alon, D. Patil, S. Naffziger, and K. Bernstein, "Scaling, power, and the future of CMOS," *IEEE International Electron Devices Meeting, 2005. IEDM Technical Digest.*, Washington, DC, 2005, pp. 7-15.
- [4] A. Davis, n.p. Jouppi, M. McLaren, N. Muralimanohar, R.S. Schreiber, N. Binkert, and J-H. Ahn, "The Role of Photonics in Future Datacenter Networks," in *Optical Interconnects for Future Data Center Networks* 1st ed, New York, Springer Science and Business Media, 2013, pp. 67-93.
- [5] C. J. Sher Decusatis, A. Carranza, and C. M. Decusatis, "Communication within clouds: open standards and proprietary protocols for datacenter networking," *Communications Magazine, IEEE*, vol. 50, no. 9, 2012, pp. 26-33.
- [6] T. Benson, A. Akella, and D. A. Maltz., "Network traffic characteristics of datacenters in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 267-280.
- [7] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving Datacenter Performance and Robustness with Multipath TCP", *Proceedings of the ACM SIGCOMM 2011 conference*, vol.41, no .4, 2011, pp. 266-277.
- [8] W. Ni, C. Huang, Y. Liu, K-W. Leong, and J. Wu, "POXN: A new passive optical cross-connection network for low-cost power-efficient datacenters," *Journal of Lightwave Technology*, vol. 32, no. 8, 2014, pp. 1482-1500.
- [9] Y. An and C. Huang, "A high-throughput energy-efficient passive optical network with multiple planes," MA thesis, Dept. Sysc. Eng., Carleton Univ., Ottawa, ON, 2016.
- [10] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol.19, no. 2, 2014, pp. 171-209.
- [11] R. Singh, P. Shenoy, M. Natu, V. Sadaphal, and H. Vin, "Predico: A System for What-if Analysis in Complex Datacenter Applications," in *Proc. of 12th International Middleware Conference*, Dec. 2011, pp.120-139.

- [12] G. Wang, D.G. Anderson, M. Kaminsky, K. Papagiannaki, T.S. Eugene Ng, M. Kozuch, and M. Ryan, "c-Through: Part-time optics in datacenters," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, 2011, pp. 327-338.
- [13] V. Shrivastava, P. Zerfos, K-W. Lee, H. Jamjoom, Y-H. Liu, and S. Banerjee, "Application-aware virtual machine migration in datacenters," *In Proc. of IEEE INFOCOM Mini-conference*, Apr. 2011, pp. 66-70.
- [14] G. DeCandia, G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Voshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, vol. 41, no. 6, 2007, pp. 205-220.
- [15] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, Jan. 2008, pp. 107-113.
- [16] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, Aug. 2008, pp. 63-74.
- [17] H. Liu, C. F. Lam, and C. Johnson, "Scaling optical interconnects in datacenter networks opportunities and challenges for WDM," *in Proc. IEEE Symp. High Perform. Interconnects*, Aug. 2010, pp. 113-116.
- [18] 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.17, no. 3, July. 2010, pp. 32-39.
- [19] J. Rath (2010), *Google eyes 'optical express' for its network* [Online]. Available: <http://www.datacenterknowledge.com/archives/2010/05/24/google-eyes-optical-express-for-its-network/>
- [20] M. Alizadeh, P. Costa, A. Rowstron, G. O'Shea, and A. Donnelly, "Symbiotic Routing in Future Datacenters," *in Proceedings of SIGCOMM*, vol. 40, no. 4, Aug. 2010, pp. 51-62.
- [21] W. Ni, C. Huang, and J. Wu, "Provisioning high-availability datacenter networks for full bandwidth communication," *Elsevier Computer Networks, Special Issue on Communications and Networking in the Cloud*, vol. 68, Aug. 2014, pp. 71-94.
- [22] F. Yao, J. Wu, G. Venkataramani, and S. Subramaniam, "A comparative analysis of datacenter network architectures," *2014 IEEE International Conference on Communications (ICC)*, Jun. 2014, pp. 3106- 3111.
- [23] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: a high performance, server-centric network architecture for modular datacenters," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, Oct. 2009, pp. 63-74.

- [24] A. Greenberg, J.R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible datacenter network," *ACM SIGCOMM computer communication review*, vol. 39, no. 4, Oct. 2009, pp. 51-62.
- [25] N. G. Dueld, P. Goyal, A. Greenberg, P. Mishra, K.K. Ramakrishnan, and J.E. Merwe, "A flexible model for resource management in virtual private network," *ACM SIGCOMM computer communication review*, vol. 29, no. 4, Oct. 1999, pp. 95-108.
- [26] P. Geoffray and T. Hoefer, "Adaptive Routing Strategies for Modern High Performance Networks," *High Performance Interconnects*, Aug. 2008, pp. 165-172.
- [27] A. Komornicki, G. Mullen-Schulz, and D. Landon (2009, Jan. 14), Roadrunner: Hardware and Software Overview [Online]. Available: <http://www.redbooks.ibm.com/redpapers/pdfs/redp4477.pdf>.
- [28] X.-Y. Lin, Y.-C. Chung, and T.-Y. Huang, "A Multiple LID Routing Scheme for Fat tree Based InfiniBand Networks," *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, Apr. 2004, pp. 11-21.
- [29] J. Ahn, N. Binkert, A. Davis, M. McLaren, and R.S. Schreiber, "HyperX: Topology, Routing, and Packaging of Efficient Large-Scale Networks," *Proceedings of the Conference on High Performance Computing Networking*, No. 41, Nov. 2009, pp. 1-11.
- [30] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: A Scalable and Fault-Tolerant Network Structure for Datacenters," *SIGCOMM '08 Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, Aug. 2008, pp. 75-86.
- [31] A. Singla, C.-Y. Hong, L. Popa, and P.B. Godfrey, "Jellyfish: Networking Datacenters Randomly," *NSDI'12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, Apr. 2012, pp. 225-238.
- [32] A. R. Curtis, S. Keshav, and A. Lopez-Ortiz, "LEGUP: using heterogeneity to reduce the cost of datacenter network upgrades," *Proceedings of the 6th International conference*, Nov. 2010.
- [33] B. D. McKay and N. C. Wormald, "Uniform generation of random regular graphs of moderate degree," *J. Algorithms*, Mar. 1990, pp. 52-67.
- [34] G. Wang, D.G. Anderson, M. Kaminsky, M. Kozuch, T.S. Eugene Ng, K. Papagiannaki, M. Glick, and L. Mummert, "Your Datacenter Is a Router: The Case for Reconfigurable Optical Circuit Switched Paths," *In Proceedings of the 2009 ACM conference on SIGCOMM*, Oct. 2009.
- [35] S. Parveen, R. Radziwilowicz, S.A. Paredes, and T.J. Hall, "Evaluation of burst aggregation methods in an optical burst switched agile all-photonics network," *SPIE Photonics North 2005*, Oct. 2005.

- [36] A. Rahbar and O. Yang, "Contention avoidance and resolution schemes in bufferless all-optical packet-switched networks: a survey", *IEEE Commun. Surveys Tutorials*, vol. 10, no.4, 2008, pp. 94–107.
- [37] N. Farrington, G. Porter, S. Radhakrishnan, H.H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular datacenters," *In Proceedings of the ACM SIGCOMM 2010 conference*, vol. 40, no. 4, Oct. 2010, pp. 339-350.
- [38] X. Ye, Y. Yin, and S. J. B. Yoo, "DOS: A scalable optical switch for datacenters," *In Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, 2010, pp. 1-12.
- [39] R. Proietti *et al.*, "Scalable and distributed contention resolution in AWGR-based data center switches using RSOA-based optical mutual exclusion," *IEEE J. Sel. Topics Quantum Electron.*, vol. 19, no. 2, Jul. 2013.
- [40] R. Proietti, C.J. Nitta, Y. Yin, R. Yu, S.J.B. Yoo, and V. Akella, "Scalable optical interconnect architecture using AWGR-based TONAK LION switch with limited number of wavelengths," *J. Lightw. Technol.*, vol. 31, no. 24, Oct. 2013, pp. 4087–4097.
- [41] Y. Liu, P. Gao, B. Wong, and S. Keshav, "Quartz: a new design element for low-latency DCNs," *In Proceedings of the 2014 ACM conference on SIGCOMM*, vol. 44, no.4, Oct. 2014, pp. 283-294.
- [42] Unit price of non-tunable LR [Online]. Available: www.fiberstore.com/c/10g-sfp-plus_63.
- [43] Cisco 10GBASE SFP+ Modules Data Sheet. Cisco Corp., [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/interfaces-modules/transceiver-modules/data_sheet_c78-455693.html.
- [44] Cisco 10GBASE Dense Wavelength-Division Multiplexing SFP+ Modules Data Sheet. Cisco Corp., [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/interfaces-modules/transceiver-modules/data_sheet_c78-455693.pdf.
- [45] Cisco Nexus 3548 and 3524 Switches Data Sheet. Cisco Corp., [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-3548-switch/data_sheet_c78-707001.html.
- [46] L. Chisvin and R. J. Duckworth, "Content-Addressable and Associative Memory: Alternatives to the Ubiquitous RAM," *Computer*, vol. 22, no. 7, Jul. 1989, pp. 51–64.
- [47] R. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: a scalable fault-tolerant layer 2 data center network fabric," *In Proceedings of the ACM SIGCOMM 2009 conference*, vol. 39, no. 4, Oct. 2009, pp.

39-50.

- [48] Electricity price [Online]. Available: https://www.eia.gov/electricity/monthly/epm_table_grapher.cfm?t=epmt_5_6_a.
- [49] N. Rasmussen, “*Calculating total cooling requirements for datacenters*,” American Power Conversion [Online]. Available: http://www.apcmedia.com/salestools/NRAN-5TE6HE/NRAN-5TE6HE_R3_EN.pdf?sdirect=true.
- [50] T. Benson, A. Anand, A. Akellam, and M. Zhang, “Understanding datacenter traffic characteristics,” *ACM SIGCOMM Computer Communication Review*, vol. 40, no 1, Jan. 2010, pp. 92-99.
- [51] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges”, *Journal of Internet Services and Applications*, vol. 1, no. 1, May. 2010, pp. 7-18.
- [52] A. Greenberg, J. Hamilton, D.A. Maltz, and P. Patel, “The Cost of a Cloud: Research Problems in Datacenter Networks,” *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, Jan. 2009, pp. 68-73.
- [53] Understanding Switch Latency Cisco Nexus 3000 series switches Data Sheet. Cisco Corp., [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-3000-series-switches/white_paper_c11-661939.html
- [54] W. Ni, C. Huang, J. Wu, and M. Savoie, “Availability of survivable Valiant load balancing (VLB) networks over optical networks”, *Optical Switching and Networking*, vol.10, no. 3, July 2013, pp. 274-289.