UNIVERSITY OF CALIFORNIA, SAN DIEGO

Optics in Data Center Network Architecture

A dissertation submitted in partial satisfaction of the requirements for the degree
Doctor of Philosophy

in

Computer Science (Computer Engineering)

by

Nathan Farrington

Committee in charge:

Professor Amin Vahdat, Chair
Professor Bill Lin
Professor George Papen
Professor Stefan Savage
Professor George Varghese

2012

DEDICATION

To my family.

EPIGRAPH

And the Lord said, "Behold, they are all one people with one language; and this is only the beginning of what they will do, and now nothing they have imagined will be impossible for them."

GENESIS 11:6

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor and SAIC Chair Amin Vahdat, for his mentorship and support over the past several years, from whom I have learned my valuable lessons, such as the importance of finishing projects, how to choose problems, and how to lead through code. I would like to thank the other steller members of my committee: Bill Lin, George Papen, Stefan Savage, and George Varghese. I could not ask for a better group of experts to donate their valuable time to helping me graduate. Although Amin is my sole advisor, I have worked closely with three others through the past few years that I consider to be honorary advisors: George Porter, Yeshaihu Fainman, and George Papen. I am very lucky to have had no less than four world-class researchers investing their time and effort in me.

Research in the field of computer science is seldom a solo activity. I would like to thank my numerous collaborators, who are listed next.

Chapter 2, in part, reprints material as it appears in the paper titled “Data Center Switch Architecture in the Age of Merchant Silicon”, published in the Proceedings of the 17th IEEE Symposium on High-Performance Interconnects, August 2009, by Nathan Farrington, Erik Rubow, and Amin Vahdat [1].

Chapter 2, in part, reprints material as it appears in the paper titled “Scale-Out Networking in the Data Center”, published in IEEE Micro, 30(4), pp 29–41, July/August 2010, by Amin Vahdat, Mohammad Al-Fares, Nathan Farrington, Radhika Niranjan Mysore, George Porter and Sivasankar Radhakrishnan [2].

Chapter 3, in part, reprints material as it appears in the paper titled “Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers”, published in the Proceedings of the 2010 ACM SIGCOMM Conference, by Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat [3].

Chapter 3, in part, reprints material as it appears in the paper titled “Hardware Requirements for Optical Circuit Switched Data Center Networks”, published in the Proceedings of the 2011 Optical Fiber Communication Conference, by Nathan Farrington, Yeshaiahu Fainman, Hong Liu, George Papen, and Amin

Vahdat [4].

Chapter 4, in part, has been submitted for publication in a paper titled "Mordia: A Data Center Network Architecture for Microsecond Circuit Switches", by Nathan Farrington, Richard Strong, Alex Forencich, Pang-chen Sun, Tajana Rosing, Yeshaiahu Fainman, Joseph Ford, George Papen, George Porter, and Amin Vahdat.

Chapter 4, in part, reprints material as it appears in the paper titled "Hunting Mice with Microsecond Circuit Switches", published in the Proceedings of the 11th ACM Workshop on Hot Topics in Networks, October 2012, by Nathan Farrington, George Porter, Yeshaiahu Fainman, George Papen, and Amin Vahdat [5].

VITA

2003	Bachelor of Science in Computer Engineering *cum laude*, University of Florida
2003–2008	Department of Defense, SPAWAR Systems Center Pacific
2009	Master of Science in Computer Science, University of California, San Diego
2010	Candidate of Philosophy in Computer Science (Computer Engineering), University of California, San Diego
2008–2012	Research Assistant, Department of Computer Science and Engineering, University of California, San Diego
2012	Teaching Assistant, Department of Computer Science and Engineering, University of California, San Diego
2012	Doctor of Philosophy in Computer Science (Computer Engineering), University of California, San Diego

PUBLICATIONS

Nathan Farrington, George Porter, Yeshaiahu Fainman, George Papen, and Amin Vahdat, "Hunting Mice with Microsecond Circuit Switches", *Proceedings of the 11th ACM Workshop on Hot Topics in Networks (HotNets 2012)*, October 2012.

Nathan Farrington, Yeshaiahu Fainman, Hong Liu, George Papen, and Amin Vahdat, "Hardware Requirements for Optical Circuit Switched Data Center Networks", *Proceedings of the Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC 2011)*, pp 1–3, March 2011.

Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. "Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers", *Proceedings of the ACM SIGCOMM 2010 Conference*, pp 339–350, August 2010.

Amin Vahdat, Mohammad Al-Fares, Nathan Farrington, Radhika Niranjan Mysore, George Porter, and Sivasankar Radhakrishnan, "Scale-out Networking in the Data Center", *IEEE Micro*, 30(4), pp 29–41, July/August 2010.

Nathan Farrington, Erik Rubow, and Amin Vahdat, "Data Center Switch Architecture in the Age of Merchant Silicon", *Proceedings of the 17th IEEE Symposium on High Performance Interconnects (HOTI 2009)*, pp 93–102, 2009.

ABSTRACT OF THE DISSERTATION

Optics in Data Center Network Architecture

by

Nathan Farrington

Doctor of Philosophy in Computer Science (Computer Engineering)

University of California, San Diego, 2012

Professor Amin Vahdat, Chair

Modern data center networks with their vast scale and bandwidth requirements are necessarily optical networks. Only optical communication technology has demonstrated capacities of 10 Gb/s, 40 Gb/s, and higher, over distances of more than 10 m. However, traditional data center network architectures limit the use of optics to the physical cables only, requiring wasteful electrical-optical-electrical conversions between every pair of switches. This dissertation presents three data center network architectures that use optics in novel ways for scaling the network, while simultaneously reducing CAPEX and OPEX, and managing cabling complexity.

The first architecture is a single large electronic packet switch with enough ports and enough capacity to interconnect every host in the data center. This architecture uses a FatTree topology of merchant silicon switch ASICs, and manages the resulting cabling complexity using a combination of horizontal-vertical arrangements of circuit boards, and aggregation to higher bitrates for multiplexing over thin optical fiber cables.

The second architecture is a hybrid electrical/optical network, with a smaller portion of the core switches being electrical packet switches and the remaining switches being optical circuit switches. Wavelength division multiplexing reduces the number of optical circuit switches and cables even further. This architecture has a lower CAPEX and OPEX than a traditional data center network, while also having a performance that approaches that of a traditional fully packet switched network. Measurements are presented from a fully working prototype network called Helios.

The third architecture is a microsecond-scale optical circuit switch that is fast enough to completely replace electronic packet switches for bulk traffic. A novel circuit scheduling algorithm is presented called traffic matrix scheduling that can support arbitrary communication patterns and executes in polynomial time. Measurements are presented from a fully working prototype switch called Mordia.

The combination of these architectures and their promise of less expensive bandwidth scaling will hopefully bring data center operators closer to the goal of treating the entire data center as a single computer.

Chapter 1

Introduction

Consider a data center with 1,024 racks, with each rack containing 20 servers, with each server containing 16 processor cores, 64 GB of memory, and 48 TB of storage. If these servers are interconnected with a high-performance data center network (DCN), the resulting cluster will be a supercomputer having 327,680 cores, 1.31 PB of memory, and 983 PB of storage. Such supercomputers are routinely used by companies such as Google, Microsoft, Facebook, Yahoo!, Amazon.com, eBay, and others, to provide modern web and mobile applications such as social networking, search, maps, video streaming, e-commerce, and multiplayer gaming.

This dissertation addresses the question of *how should this high-performance DCN be physically constructed*? Deeper integration of optical communications technologies, namely optical transmission and optical switching, into DCNs, can substantially improve performance, cost, and energy efficiency, thereby making possible to run applications that previously were too demanding for current network technologies. To briefly summarize the findings, DCNs should be constructed using select optical communications technologies to overcome the cost, power, performance, and scalability problems of traditional electronic packet switched (EPS) DCNs.

Modern data center networks with their vast scale and bandwidth requirements are necessarily optical networks. Only optical communication technology has demonstrated capacities of 10 Gb/s, 40 Gb/s, and higher, over distances of more than 10 m. However, traditional data center network architectures limit the use of optics to the physical cables only, requiring wasteful electrical-optical-electrical conversions between every pair of switches. This dissertation presents three data center network architectures that use optics in novel ways for scaling the network, while simultaneously reducing CAPEX and OPEX, and managing cabling complexity.

The first architecture (chapter 2) is a single large electronic packet switch with enough ports and enough capacity to interconnect every host in the data center. This architecture uses a FatTree topology of merchant silicon switch ASICs, and manages the resulting cabling complexity using a combination of horizontal-vertical arrangements of circuit boards, and aggregation to higher bitrates for multiplexing over thin optical fiber cables.

Although other FatTree switches comprised of merchant silicon ASICs have existed prior to this dissertation, these switches were all point solutions. A data center network engineer would still have to create a solution by networking these switches together. In other words, one would still need to resolve the problem of cabling complexity. In contrast, the ambitious design of the first switch architecture spans up to an entire data center while simultaneously eliminating cabling complexity. The number of long cables is dramatically reduced from 3,456 to 96, with the 96 cables further reduced to just 24 independent bundles. Although the author cannot take credit for this phenomena, industry has moved decidedly in the direction of constructing future data center networks as FatTrees of merchant silicon, using the same techniques presented in this first architecture.

The second architecture (chapter 3) is a hybrid electrical/optical network, with a smaller portion of the core switches being electrical packet switches and the remaining switches being optical circuit switches. Wavelength division multiplexing reduces the number of optical circuit switches and cables even further. This architecture has a lower CAPEX and OPEX than a traditional data center network, while also having a performance that approaches that of a traditional fully packet switched network. Measurements are presented from a fully working prototype network called Helios.

In an example data center network with 64 pods, each containing 1,024 host, allocating 10% of core switches as EPS switches and the other 90% of the core switches as OCS switches (Optical Circuit Switches), yields a cost reduction of \$62.2M down to \$22.1M, a power consumption reduction of 950.3 kW down to 157.2 kW, and a cabling complexity reduction of 65,536 long cables down to 14,016. This represents reductions of 2.8x, 6.0x, and 4.7x, respectively. Furthermore, this architecture is realizable today with commercially available components.

The second architecture shows clear wins in terms of CAPEX and OPEX reduction. But in order to realize these wins, the performance of the network must be on par with traditional FatTree EPS networks. Measurements are presented from the Helios prototype showing that a FatTree EPS network achieved an efficiency of 71.3% throughput compared to an ideal switch, while the Helios EPS/OCS hybrid network achieved an efficiency of 59.2%, over the same set of communication patterns. When these performance metrics are applied to the example data center network, the FatTree EPS would cost 1,331 \$/Gb/s and the hybrid EPS/OCS architecture would cost 569 \$/Gb/s. So while the FatTree EPS network might have more absolute performance, the performance achieved by the hybrid EPS/OCS network is more more cost effective. The same holds true for power consumption and cabling complexity.

The third architecture (chapter 4) is a microsecond-scale optical circuit switch that is fast enough to completely replace electronic packet switches for bulk traffic. A novel circuit scheduling algorithm is presented called traffic matrix scheduling that can support arbitrary communication patterns and executes in polynomial time. Measurements are presented from a fully working prototype switch called Mordia.

The primary difference between the second and third architectures is that the second architecture uses millisecond-scale OCS switches that assume a single input-output port assignment until the global traffic demand matrix changes sufficiently to warrant a reassignment, while the third architecture uses microsecond-scale OCS switches that can switch between a schedule of assignments to approximate any communication pattern. This makes microsecond-scale OCS switches much more useful than millisecond-scale switches and further reduces the need for EPS switches entirely.

The combination of these architectures and their promise of less expensive bandwidth scaling will hopefully bring data center operators closer to the goal of treating the entire data center as a single computer.

1.1 Scope

There are several related questions which are outside the scope of this dissertation. One similar and important question is *what is the best DCN topology*? When viewed from the lens of graph theory, a DCN can be understood as a graph, with vertices representing cores and switches, and weighted edges representing data communication links. The author's investigation into this problem suggestions that the question of the optimal topology is closely intertwined with both communication patterns as well as packaging constraints. There has been preliminary work in this field [6]. We differentiate between "topology" and "architecture" by noting that a topology is a parameterized mathematical construct, but by itself is not a network. In contrast, an architecture is a blueprint for constructing a network, and a network will have a particular topology. Architecture encompasses many practical details such as cost, power consumption, packaging, and cabling complexity, as well as traditional network metrics such as capacity and latency.

Another popular question is *what is the best communication protocol (e.g. link-layer, addressing, transport)*? There has been much work in this field as well. Currently, software-defined networking (SDN) and network virtualization are popular research directions, and many are seeking to understand how such technologies could fit into a DCN architecture. Eventually, it is hoped that all such questions will be answered, however this dissertation focuses only on the question of data center network architecture.

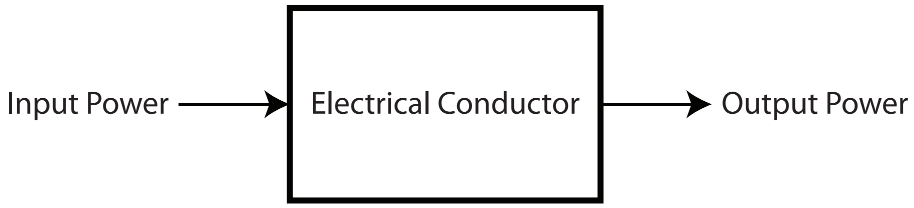


Figure 1.1: An electrical conductor attenuates the input power.

1.2 Copper is Dead

Although the title of this section might seem sensational, the fact that copper physically cannot support the high bitrates required by WANs, HPC interconnects, and now most recently DCNs, is actually conventionally accepted by the optical and electrical engineering communities. However, this fact is not yet well known in computer science.

Consider an electrical conductor used for data communication, such as a copper cable (Figure 1.1). If input power is applied to the electrical conductor then a smaller amount of output power will be measured on the other side. The input power is attenuated and typically is dissipated as heat.

What is the minimum output power? Measurements of a 10 Gb/s SFP+ optical transceiver showed a receiver sensitivity of -19 dBm (12.59 μW). In other words, the receiver required an input power level of at least 12.59 μW to differentiate between zeros and ones. Receiver sensitivity varies widely between different communication technologies (electrical, optical, wireless). The exact receiver sensitivity is actually immaterial for this section, but -19 dBm is assumed as the minimum output power for this example.

What is the maximum input power? Consider a copper wire with a diameter of 3.6 mm, corresponding to an American Wire Gauge (AWG) number of 7. This is a relatively thick wire for data communication. Such a wire would have a fusing current[1] of 561 A, meaning that after carrying 561 A for 10 seconds, this wire would melt[2]. For this example, 561 A is the maximum input current. Assuming a

[1]`https://en.wikipedia.org/w/index.php?title=American_wire_gauge&oldid=523999576`

[2]A safe amperage for this wire would be far less, around 30 A.

Table 1.1: Attenuation and maximum length of AWG 7 copper wire.

Bit Rate (Gb/s)	dB/m	Loss/m	Max length (m)
1	0.721	15.3%	106
10	2.28	40.8%	33.5
40	4.56	65.0%	16
100	7.21	81.0%	10.6

signaling level of 1 V, the maximum input power would be 57.49 dBm (561 W).

Subtracting these two numbers, $57.49 - (-19) = 76.49$, gives a power budget of 76.49 dB. In other words, if an input power of 57.49 dBm is attenuated by 76.49 dB, then the output power would be -19 dBm, which is the minimum output power in this example.

How much power is lost due to attenuation by the copper cable? Unfortunately, the attenuation of an electrical conductor increases with the square root of the signal frequency. Miller and Ozaktas [7] gives an attenuation of $7.2 \times 10^{-6}\sqrt{f}$ dB/m for AWG 7 copper wire. Table 1.1 shows the attenuation of AWG 7 copper wire in terms of loss per meter, for the common bit rates of 1 Gb/s, 10 Gb/s, 40 Gb/s, and 100 Gb/s. The results are sobering. At 10 Gb/s, 40.8% of the input power is lost per meter. At 100 Gb/s, 81.0% of the input power is lost per meter. This is compounded. For two meters, 96.39% of the input power is lost at 100 Gb/s. The last column shows the theoretical maximum cable length for AWG 7 copper wire.

One should note that the IEEE 1000BASE-T standard supports 1 Gb/s Ethernet (1.25 Gb/s at Layer 1) at distances of 100 m using 8 strands of the much thinner AWG 24 copper wire, with a power consumption far lower than 561 W. The reason is that 1000BASE-T uses sophisticated digital signal processing (DSP) to increase the signal-to-noise ratio. The Shannon-Hartley Theorem (1.1) gives the capacity, C, of a communication channel as

$$C = f \;\lg(1 + \mathrm{SNR}) \tag{1.1}$$

C is proportional to the signal frequency, f, but only logarithmically proportional to the signal-to-noise ratio. This means that DSP techniques that increase the SNR provide exponentially smaller benefit than increasing the signal frequency, and cannot be used to overcome the fact that the attenuation of copper cables is frequency-dependent. At sufficiently high bitrates, copper is dead.

In contrast, the attenuation of single-mode optical fiber is 2.0×10^{-4} dB/m, meaning that it has a loss rate of 0.0046% per meter, and this loss rate is independent of frequency. Optical fiber also has a theoretical maximum bandwidth (frequency) of 50 THz, corresponding to a maximum bitrate of 330 Tb/s [8]. In addition, optical fiber is extremely thin and light, so it can be bundled into densely into cables to further increase the capacity, and is resistant to electromagnetic interference.

But optical communications are not a panacea to the problems encountered in constructing high-performance interconnects. The required lasers add cost, heat, and reliability concerns. Optical fiber is also more fragile than copper wire. But despite these concerns, there is currently no known alternative communications technology that can compete with optical fiber for implementing high-performance networks. The future of data center networking is optical networking.

Chapter 2

Reducing Cabling Complexity with Optics

Traditional packet-switched data center networks (DCNs) that scale to tens of thousands of hosts currently use highly-specialized and custom ASICs, with correspondingly high development costs. Simultaneously, these networks also face significant performance and management limitations at scale. Just as commodity processors and disks now form the basis of computing and storage in the data center, there is an opportunity to leverage emerging high-performance commodity merchant switch silicon as the basis for a scalable, cost-effective, modular, and more manageable DCN fabric. Due to the ever increasing demand for bisection bandwidth, DCNs are necessarily constructed as a mesh of interconnected packet switches. However, these switches are currently packaged individually, which wastes resources, and connected manually, which is labor intensive and error prone. This chapter describes how to save cost and power by repackaging an entire DCN as a distributed multi-stage switch using a FatTree topology of merchant silicon ASICs instead of a crossbar topology of proprietary ASICs, with optical fiber used to replace multiple copper cables via bandwidth aggregation. Compared to a FatTree of discrete packet switches, a 3,456-port 10 Gigabit Ethernet realization of this design costs 52% less, consumes 31% less power, occupies 84% less space, and reduces the number of long, cumbersome cables from 6,912 down to 96, relative to existing approaches.

2.1 Introduction

With the help of parallel computing frameworks such as MapReduce [9], organizations routinely process petabytes of data on computational clusters containing thousands of nodes. For these massively parallel workloads, the principal bottleneck is often not the performance of individual nodes, but rather the rate at which nodes can exchange data over the network. Many data center applications demonstrate little communication locality, meaning that the communication substrate must support high aggregate bisection bandwidth for worst-case communication patterns. Unfortunately, modern DCN architectures typically do not scale beyond a certain amount of bisection bandwidth [10] and become prohibitively expensive well in advance of reaching their maximum capacity [11], in some cases oversubscribed by a factor of 240 [12].

There is interest in replacing these expensive packet switches with many smaller, commodity switches, organized into a FatTree topology [11]. But as the number of packet switches grows, so does the cabling complexity and the difficulty of actually constructing the network, especially on a tight schedule and with a minimum amount of human error. FatTrees have been used successfully in telecom networks [13], HPC networks [14], and on chips [15], but not yet in data center Ethernet networks. One reason is the fear of the resulting cabling complexity from trying to interconnect thousands of individual switches and the overhead of managing a large number of individual switch elements.

In addition to the cabling problem, networks of discrete packet switches are unnecessarily wasteful in the data center. The relative close proximity of the compute nodes and the single administrative domain provide opportunities for eliminating redundant components, such as packaging, power conditioning circuitry, and cooling. Multiple CPUs and memory banks could be consolidated to save cost and power. Consolidation could also reduce the cost of inter-switch links, which often use expensive and power hungry optical transceivers.

This chapter describes the design a multi-stage switch architecture leveraging merchant silicon to reduce the cost, power consumption, and cabling complexity of DCNs, while also increasing the bisection bandwidth available to parallel applications such as MapReduce. In essence, a FatTree of discrete packet switches is repackaged as a single distributed multi-stage switch, and redundant components are eliminated to save cost and power. One realization of this architecture is presented: a 3,456-port 10 Gigabit Ethernet (10 GbE) switch with 34.56 Tb/s of bisection bandwidth. Using commodity 24-port 10 GbE merchant silicon as the fundamental building block, cost, power, and cabling complexity is further reduced by the design and implementation of a new Layer 2 protocol, called Ethernet extension protocol (EEP), in hardware to aggregate four 10 GbE links into a single 40 GbE link, and vice versa. Aggregation to higher bitrates makes optical fiber cables more economical and helps justify their use in replacing multiple copper cables. The combination of custom packaging, the EEP protocol, and optical fiber, reduces the number of inter-switch cables from 6,912 to just 96. This architecture generalizes to 65,636 ports of 10 GbE with 64-port 10 GbE switch silicon.

2.1.1 Bisection Bandwidth, Cabling Complexity, and Fat-Trees

The primary goal when constructing a DCN for a computational cluster is to provide enough bisection bandwidth so that communication-intensive parallel computations can maintain high levels of CPU utilization. A bisection of a network is a partition into two equally-sized sets of nodes. The sum of the capacities of links between the two partitions is called the bandwidth of the bisection. The *bisection bandwidth* of a network is the minimum such bandwidth along all possible bisections. Therefore, bisection bandwidth can be thought of as a measure of worst-case network capacity.

Cabling complexity is the number of long inter-switch cables required to construct a particular DCN. Short intra-rack cables or cables that cross between adjacent racks are not difficult to install and maintain, but long cables require planning and overhead cable trays. They are also more difficult to test and to replace after a break. This in turn increases the cost of the network.

The FatTree topology is very promising because it provides an enormous amount of bisection bandwidth while using only small, uniform switching elements.

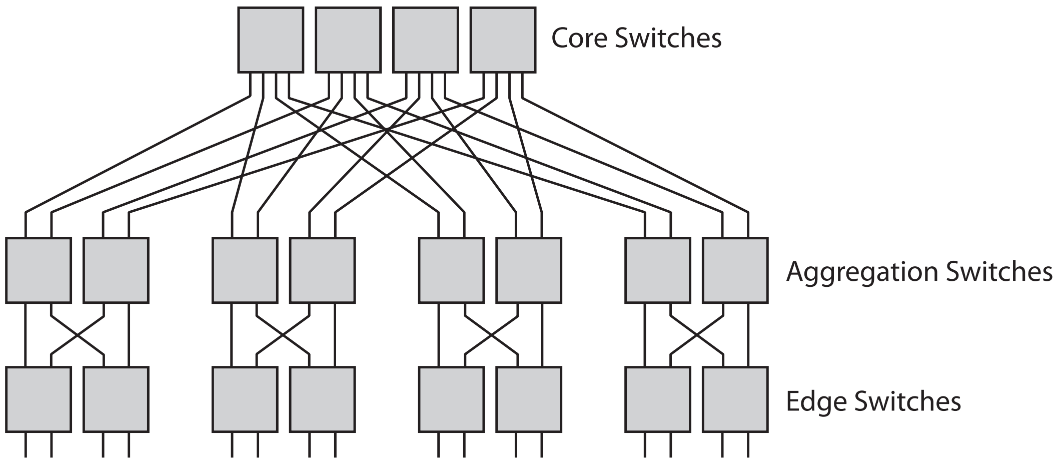


Figure 2.1: A 3-tier (5-stage) FatTree using 4-port switching elements.

Figure 2.1 shows a small example of the FatTree topology constructed from 4-port switching elements. This example is equivalent to a 16-port switch. Bandwidth scaling is achieved by having multiple, redundant paths through the network. However, if one is not careful, these links can translate into a large degree of cabling complexity, making the network impractical to construct and maintain.

2.2 Technologies

DCNs are composed of racks, switches, cables, and transceivers. This section briefly reviews these four technologies.

2.2.1 Racks

Almost all data processing and networking equipment is housed in large metal racks [16]. A typical rack measures 0.6 m wide by 1.0 m deep by 2.0 m high, has an unloaded weight of 170 kg, and can support a maximum load of 900 kg[17]. A 2.0 m high rack is partitioned into 42 vertical rack units (denoted as RU, or simply U). Each RU is 44.45 mm (1.75 in) high. Rack-mountable equipment occupies one or more rack units.

Racks are lined up side-by-side in rows. To assist with cooling, rows face front-to-front and back-to-back to form what are called *cold aisles* and *hot* *aisles* [18]. A cold aisle is at least 1.22 m (4 ft) wide and allows human access to the front panels of the racks. A hot aisle is at least 0.9 m wide and is heated by the air exhaust from the servers' cooling fans. Most cables are in the hot aisles.

Generally, there are three ways to route cables between racks. If the racks are in the same row, then the simplest way is to run the cables inside the racks. Specialized wire management solutions exist to facilitate this. If there are lots of cables, then it may be best to leave the top one or two RUs empty. If the racks are in different rows, then it is common to run the cables along the ceiling, suspended in overhead cable trays. This is standard practice since it reduces clutter and prevents safety hazards [18]. Some data centers have raised floors with removable tiles. The space underneath the floor can also be used to route cables, including power cables.

2.2.2 Switches

The most common Ethernet switch in the data center is the so-called *top-of-rack* (TOR) switch. This is a 1-RU switch that is placed in the top position in a rack and connects to all of the compute nodes in that rack. These switches are becoming commodities due to the recent emergence of merchant silicon. TOR_1 in Table 2.1 is a 48-port GbE / 4-port 10 GbE switch. Prices typically range between \$2,500 and \$10,000. TOR_{10} is a 24-port 10 GbE switch, with prices between \$5,000 and \$15,000. Both switches consume approximately 200 W of power.

Table 2.1: Ethernet switch models.

	TOR_1	TOR_{10}	EOR
GbE Ports	48	0	0
10 GbE Ports	4	24	128
Power (W)	200	200	11,500
Size (RU)	1	1	33

Also present in data centers are so-called *end-of-row* (EOR) switches. An EOR switch derives its name from the fact that it is placed in a rack, sometimes by itself due to its size, and all nearby switches in other racks connect directly to it. EOR switches are also called *modular* switches, because they accept a variety of modules, called *line cards*, with different Layer 1 interfaces and different combinations of switching fabrics and network processors. EOR in Table 2.1 is a 128-port 10 GbE switch. Prices range between \$500,000 and \$1,000,000.

2.2.3 Cables and Transceivers

Table 2.2 lists properties of Ethernet cables and transceivers. The Telecommunications Industry Association recommends Cat-6 UTP instead of Cat-5e for new data center installations [18]. MMF120 is 120 strands of MMF in a single cable. Note that the maximum range depends on the modal bandwidth of the cable. Ranges shown here reflect the highest-quality cable at the time of standardization.

Table 2.2: Data center communication cables and transceivers.

	Cat-6	Twinax	MMF120	Units
Cable cost	0.41	11.2	23.3	$/m
Cable weight	42	141	440	g/m
Cable diameter	5.5	7	25	mm
Gigabit Ethernet				
Standard	1000BASE-T	N/A	1000BASE-SX	
Range	100	N/A	550	m
Transceiver cost	10	N/A	36	$
Transceiver power	0.42	N/A	0.5	W
10 Gigabit Ethernet				
Standard	10GBASE-T	10GBASE-CX4	10GBASE-SR	
Range	30	15	300	m
Transceiver cost	100	100	250	$
Transceiver power	6	0.1	1	W
40 Gigabit Ethernet				
Standard	N/A	N/A	40GBASE-SR4	
Range	N/A	N/A	300	m
Transceiver cost	N/A	N/A	600	$
Transceiver power	N/A	N/A	2.4	W

The term “Ethernet cable” usually refers to unshielded twisted pair (UTP) copper cable. UTP cable is available in different grades, which do not always correspond directly with IEEE Layer 1 standards. For example, Cat-5e allows

GbE links of up to 100 m and Cat-6a allows 10 GbE links of up to 100 m. Twinax ("InfiniBand") shielded cable allows 10 GbE links of up to 15 m, but with different engineering trade offs. For example, since twinax is shielded, the manufacturing cost is higher than UTP, but transceivers are less expensive and consume less power.

Optical fiber cables are becoming more common in the data center as 10 GbE is introduced. Multimode fiber (MMF) is often preferred over single-mode fiber (SMF) because optical transceivers for multimode fiber are significantly less expensive. OM-3 "laser grade" MMF allows 10 GbE links longer than 300 m. Distribution fiber cable can pack multiple fiber strands within a single physical cable. Typical densities are multiples of 12. A bidirectional link actually requires two fibers, so a 120-strand fiber cable only has 60 bidirectional communication channels.

A transceiver converts signals between a circuit board and a communications cable. For UTP and twinax copper cable, transceivers are packaged into a chip and placed directly onto the circuit board. Ethernet is standardized for speeds of 1 Gb/s over UTP (1000BASE-T), and 10 Gb/s over twinax (10GBASE-CX4) and UTP (10GBASE-T). UTP uses the popular "RJ-45" connector, whereas twinax uses the Fujitsu "InfiniBand" connector.

There are a number of different Layer 1 protocols for optical fiber, depending on the carrier wavelength, the bit rate, the encoding, multimode vs. single mode, etc. The networking industry has taken the 7-Layer OSI model quite literally by separating the Ethernet MAC functions (Layer 2) from the Ethernet PHY transceiver functions (Layer 1) through the use of pluggable optical transceiver modules. These modules are standardized by various Multi Sourcing Agreements (MSAs) to define standard mechanical and electrical form factors. This allows Ethernet device manufacturers to specialize in Layer 2 and Layer 3 functionality, and allows the end user to choose their desired Layer 1 protocol just by plugging in a module.

The most common optical module standard for GbE is the SFP module [19]. Recently, SFP+ [20] has become the dominant standard for 10 GbE. These modules use the LC connector, which couples two fibers together into close proximity. The QSFP transceiver is being developed for 40 GbE using an emerging standard to be named 40GBASE-SR4. This module will be slightly larger than SFP and SFP+ modules and will use an MT connector with 8 optical fibers (4 transmit, 4 receive). MT connectors are available with up to 72 optical fibers.

2.3 Motivation

Al-Fares et al. [11] proposes the construction of a 3-tier FatTree of 2,880 commodity 48-port GbE TOR switches, providing 27.648 Tb/s of bisection bandwidth. However, this network is almost impossible to construct following their suggested packaging. It would require 1,128 separate cable bundles, each of which must be manually and independently routed and installed. They partitioned the network into 48 separate pods (switches and servers) that communicate through a core switch array of 576 48-port switches. Assuming that a pod has a 3 m diameter and the distance between two pods is 1.5 m, their proposal would require 226,972 m of cable, weighing 9,532 kg.

Two simple rules will at least allow the construction of small FatTrees, thereby gaining the benefit of the FatTree topology and commodity switches. First, minimize the number of unique cable bundles. The Al-Fares proposal required $k(k-1)/2$ bundles because the top tier of the network was collocated with the k pods. However, by placing the top tier in a central location, only k bundles are required (from the pods to the central location). Second, use optical fiber cable instead of copper cable. Compared to copper, fiber is much thinner and lighter. Cables containing more than 72 fibers are readily available.

Following these rules, an alternative FatTree network based on the recently commoditized 24-port 10 GbE TOR switch could be constructed. At the time of writing, these switches are available from several manufacturers for between \$5K and \$15K. Connecting 720 such switches into a 3-tier FatTree topology will yield a bisection bandwidth of 34.56 Tb/s. Each of the 24 pods will contain 24 switches, co-located in a single rack. The 144 core switches will be distributed over 4 racks. The 3,456 long fiber cables can be combined into just 24 bundles of cable. The estimated total cost is approximately \$7.2M, which is \$2,000 per port. This network is both less expensive and provides more bisection bandwidth than a traditional DCN constructed from modular packet switches [11].

2.3.1 Merchant Silicon and the Commodity Top-of-Rack Switch

The original IP routers used in the Internet were implemented entirely in software running on CPUs. As total Internet traffic grew, stand-alone IP routers emerged containing hardware acceleration for the most common network functionality (e.g. longest prefix matching, CRC calculation), with the CPU being relegated to less common or more complicated functionality (e.g. control messages, routing protocols). The hardware acceleration is now called the fast path, and the software running on the CPU is now called the slow path.

For small IP routers and Ethernet switches, it is possible to implement the entire fast path on a single chip. Network equipment manufacturers such as Cisco, HP, Juniper, Brocade, and Force10, have traditionally designed their own ASICs, for use solely in their own products. For larger products, multiple ASICs are used. Little is known publicly about these ASICs, except one can assume that their capabilities follow Moore's law.

The last decade has seen the introduction of merchant silicon: networking ASICs produced for the network equipment mass market. For example, there are currently at least three separate makers of 24-port 10 GbE switch ASICs: Broadcom, Fulcrum, and Fujitsu. All three product offerings provide similar degrees of functionality. Table 2.3 shows that the three chips also have similar properties. Due to NDAs, some numbers could not be published, but they are comparable.

Several companies have used these ASICs to create 24-port TOR switches. Since purchasing merchant silicon is much easier and less expensive than creating a custom switch ASIC, these companies can offer TOR switches at a lower cost to consumers, which was the original motivation for constructing an entire DCN out of TOR switches. Because these manufacturers choose from the same pool of

Table 2.3: Comparison of 10 GbE switch ASICs.

Maker Model	Broadcom BCM56820	Fulcrum FM4224	Fujitsu MB86C69RBC
Ports	24	24	26
Cost	NDA	NDA	\$410
Power	NDA	20 W	22 W
Latency	< 1 us	300 ns	300 ns
Area	NDA	40 x 40 mm	35 x 35 mm
SRAM	NDA	2 MB	2.9 MB
Process	65 nm	130 nm	90 nm

merchant silicon, they differentiate their products primarily through the features in their custom software. But the board and chassis designs are usually very similar.

Figure 2.2 shows the board layout of a 24-port 10 GbE TOR switch; the cost and power consumption of these parts are given in Table 2.6. The power supply unit (PSU) and fans are physically separated from the main circuit board for signal integrity. Each SFP+ cage allows for up to 8 SFP+ modules. There is one PHY chip per SFP+ module.

A switch ASIC by itself is not an Ethernet switch; it is only the fast path. To build a functional switch, several other chips are needed, as well as software for handling networking protocols such as Minimum Spanning Tree or OSPF. This software runs on a local CPU. The CPU does not need to be very powerful since the amount of required computation is typically small. Common operating systems include Linux and VxWorks. The CPU also requires supporting chips such as DRAM for system memory and flash for secondary storage.

A switch also contains several PHY chips, which are used to bridge the Layer 1 protocol supported by the ASIC with the Layer 1 protocol exposed by the switch. For example, an ASIC may only support XAUI natively, but by adding a PHY chip, the XAUI protocol can be converted into 10GBASE-T, which allows Cat-6a cables to be connected to the switch.

These three classes of chips: ASICs, CPUs, and PHYs, are the primary building blocks of switches and routers. Sometimes switches will contain other chips such as flash memory for nonvolatile storage or FPGAs for glue logic. Also present are power supply units (PSUs) and fans. Although optical transceiver

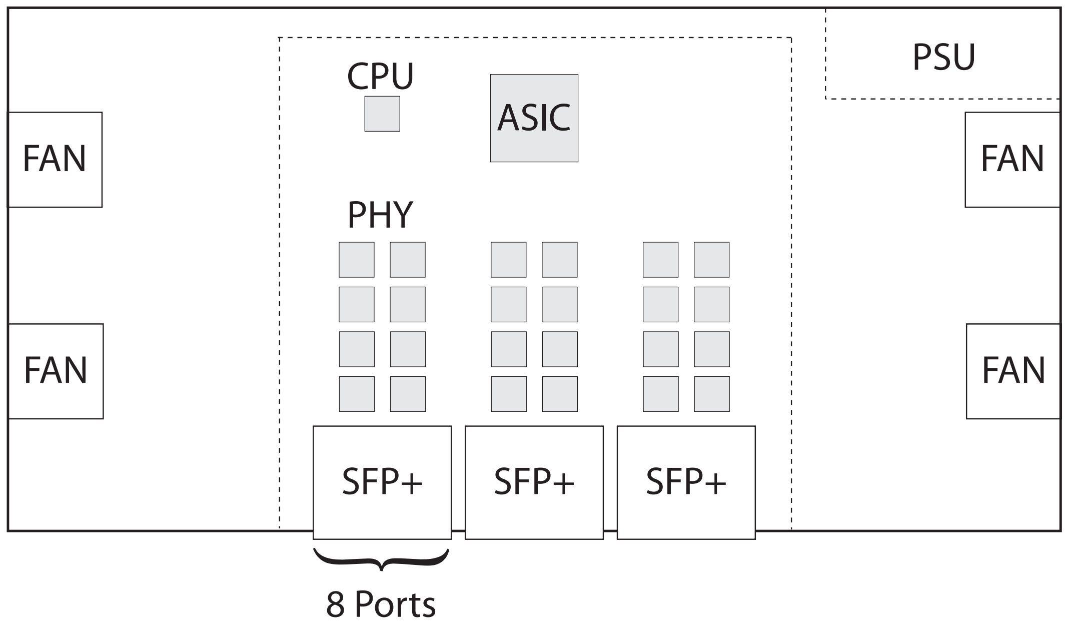


Figure 2.2: Layout of a 24-port 10 GbE TOR switch.

modules are not technically part of the switch, they are included in the analysis since they contribute greatly to the overall cost and power.

2.4 Design of a 3,456-port Switch

This section describes the design of a 3,456-port 10 GbE switch. What makes this switch novel is that it is comprised entirely of merchant silicon, connected in a FatTree topology. In contrast, large Ethernet switches from traditional network equipment manufacturers use multiple proprietary ASICs and a crossbar topology. One simple and completely optional ASIC, called EEP, is introduced in section 2.6 to further reduce the cost, power consumption, and cabling complexity of the overall design. In section 2.5, it is shown how this switch is strictly better than a FatTree of commodity TOR switches.

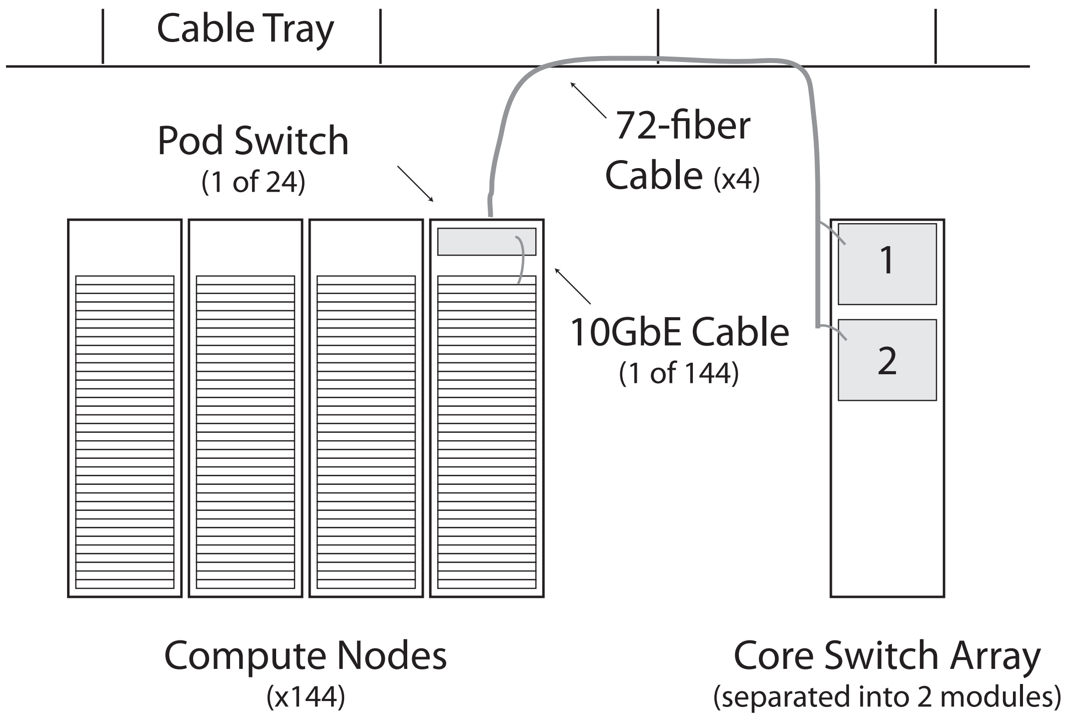


Figure 2.3: Example deployment of the 3,456-port switch.

2.4.1 Overview

Figure 2.3 shows an overview of the 3,456-port switch. Rather than build one monolithic switch, the design is separated into 24 pod switches and a core switch array. A pod switch occupies 4 rack units of space, whereas each of the two core switch array modules occupies 9 rack units of space. A pod switch is basically the bottom two tiers of the FatTree; the core switch array forms the top tier.

Each pod switch can function as a standalone 144-port 10 GbE switch. But when connected to the core switch array, the pod switches act as a single non-interfering [21, p112] switch. The pod switches can also be incrementally deployed as the network is built out. When fully deployed, the switch has 3,456 ports and 34.56 Tb/s of bisection bandwidth. Each pod switch connects to the core switch array with four parallel cables, each cable carrying 72 multimode fibers. These cables can be routed in an overhead cable tray.

The core switch array is not a monolithic switch; it is a collection of 144 individual 24-port switches. The array is separated into two modules to provide

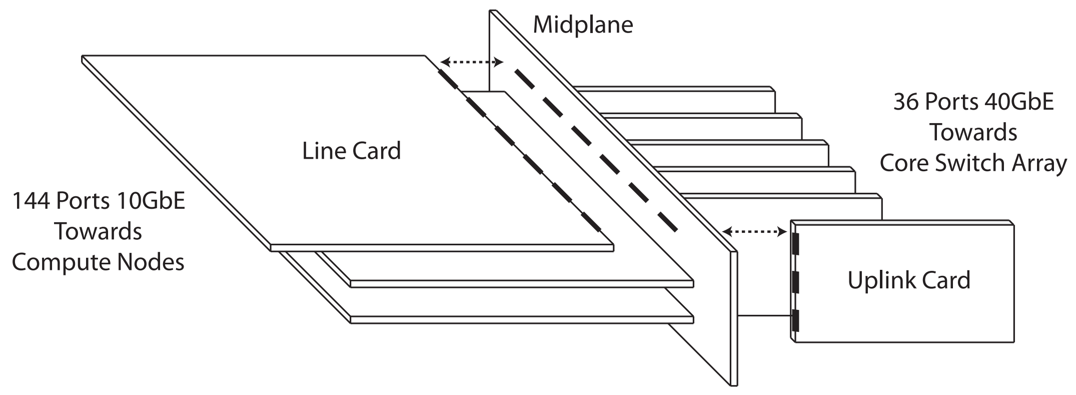

Midplane
36 Ports 40GbE Towards Core Switch Array
Line Card
144 Ports 10GbE Towards Compute Nodes
Uplink Card

Figure 2.4: Pod switch physical organization.

fault tolerance and to allow a limited form of incremental deployment. It is also possible to divide the core switch array into 4 modules, but our analysis showed that it would increase the overall cost of the system. The core switch array is shown installed into a single rack for illustration purposes, but in actual deployment, the modules would be physically separate to provide fault tolerance of critical network infrastructure, e.g. in the face of localized power failure.

2.4.2 Pod Switch: Organization

The pod switch is constructed from multiple circuit boards, assembled as shown in Figure 2.4. The midplane separates and connects the three line cards and the six uplink cards. The horizontal-vertical arrangement allows each line card to connect directly with each uplink card, and vice versa. At the center of the switch chassis is a single midplane circuit board. The midplane provides three important functions. First, it connects the line cards to the uplink cards through high-density high-speed electrical connectors. Second, it provides power to all of the line cards and daughter cards. Third, it contains a CPU that manages the state of the pod switch. The midplane design is quite simple since communication signals from each line card pass directly to each uplink card, and vice versa. Since the line cards are mounted horizontally and the uplink cards are mounted vertically, this can be called a *horizontal-vertical arrangement*.

The electrical connectors are inexpensive and available from several vendors [22]. The male connectors are attached to the midplane and the female connectors are attached to the cards. Each connector has at least 16 pins to support 8 differential channels of 10 GbE between one line card and one uplink card. There are 18 connectors per pod switch, plus additional connectors for power distribution.

In addition to these boards, the chassis also contains dual redundant power supply units (PSUs), which have been omitted from Figure 2.4 for clarity. They are located in the back of the chassis on the left and right sides of the uplink cards.

2.4.3 Pod Switch: Line Card

Figure 2.5 shows the layout of a line card. Each line card essentially replaces four discrete 24-port switches from the edge layer of the network and eliminates redundant components. The four switch ASICs separate the board into two halves. The bottom half of the board contains 48 SFP+ optical transceiver cages and 48 PHY chips, to convert between the SFP+ 10G electrical interface (SFI) and the IEEE XAUI standard for 10 GbE. Figure 2.5 shows 12 solid traces running from the SFP+ cages, to the PHYs, to one of the switch ASICs. The other traces are shown with dotted lines to indicate that they also continue, but have been removed from the diagram for clarity.

The CPU is shown off to the side. It connects directly to the four switch ASICs using the PCI Express bus. This allows the CPU software to configure the switch ASICs, and allows exceptional Ethernet frames to be sent from the switch ASICs over the PCI Express bus to the CPU for further processing.

The top half of the board contains an additional 48 PHYs and 6 electrical connectors. These PHYs convert between XAUI and 10GBASE-KR, which is the IEEE standard for 10 GbE over backplanes. 10GBASE-KR is more robust than XAUI when passing between different boards. Figure 2.5 shows an additional 12 traces from the switch ASIC to one row of PHYs on the top half of the board. Two traces from each switch ASIC are routed to each of the 6 uplink cards.

Circuit boards are divided into multiple layers and can be categorized as either signal layers or power/ground layers. Figure 2.6 shows that the line card requires four separate signal layers to route all copper traces. Congestion occurs in

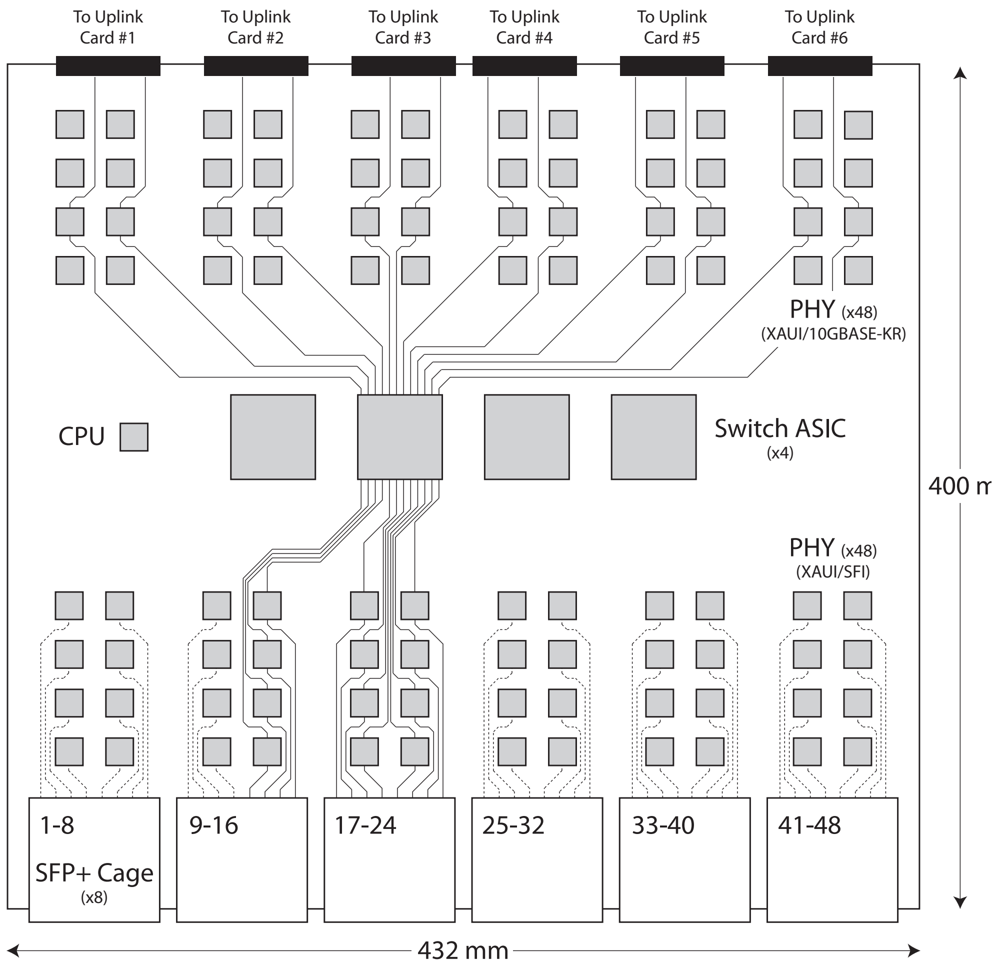


Figure 2.5: Pod switch line card.

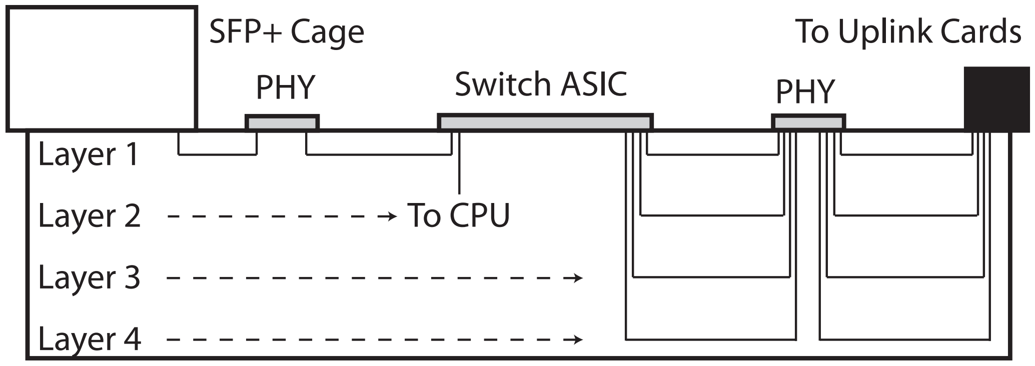


Figure 2.6: Layers of the pod switch line card circuit board.

the top half of the board where each switch ASIC must connect to each connector. This figure is not to scale.

2.4.4 Pod Switch: Uplink Card

The uplink card performs two functions. First, it acts as a switch fabric for the pod switch, allowing the 12 switch ASICs on the 3 line cards to connect to each other. Second, it forwards traffic to and from the core switch array.

Figure 2.7 shows the layout of an uplink card. Each uplink card essentially replaces two discrete 24-port switches from the aggregation layer of the network. Like the line card, the uplink card is also divided into two halves. The bottom half connects to the midplane with electrical connectors. The 8 traces from each line card are routed to 8 PHYs, and then split between the two switch ASICs. In Figure 2.7, traces connecting to the first switch ASIC are shown as solid lines, whereas traces connecting to the second switch ASIC are partially shown as dotted lines. This half of the board requires two separate signal layers. Other traces have been removed from the diagram for clarity.

The top half of the board contains six EEP ASICs and six QSFP cages. Each EEP ASIC connects to four ports of the switch ASIC using the XAUI protocol and connects to one of the QSFP modules using the QSFP electrical interface. The EEP ASIC is described in more detail in section 2.6. Essentially, the top half of the board is aggregating the 24 ports of 10 GbE into 6 ports running the custom

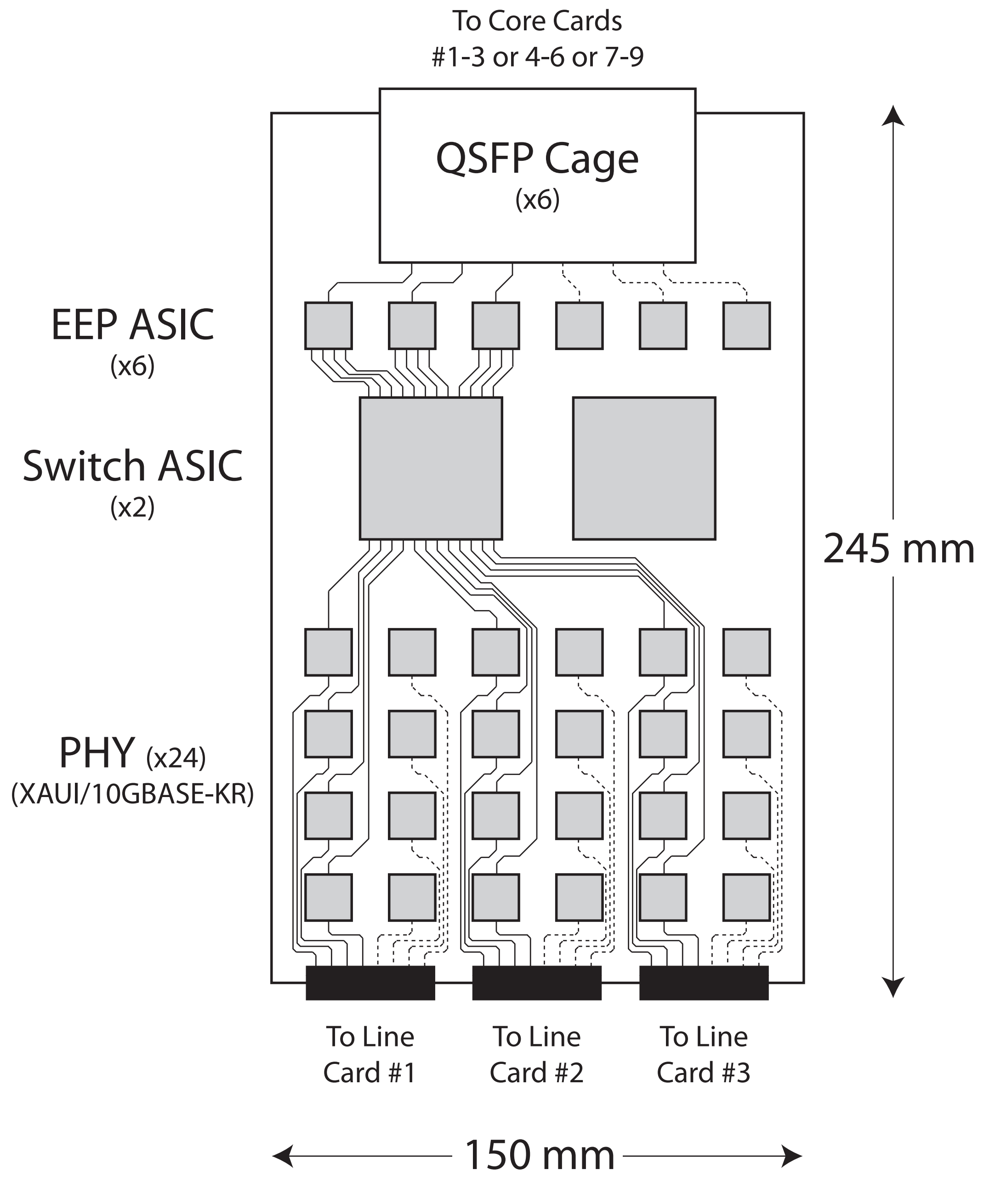


Figure 2.7: Pod switch uplink card.

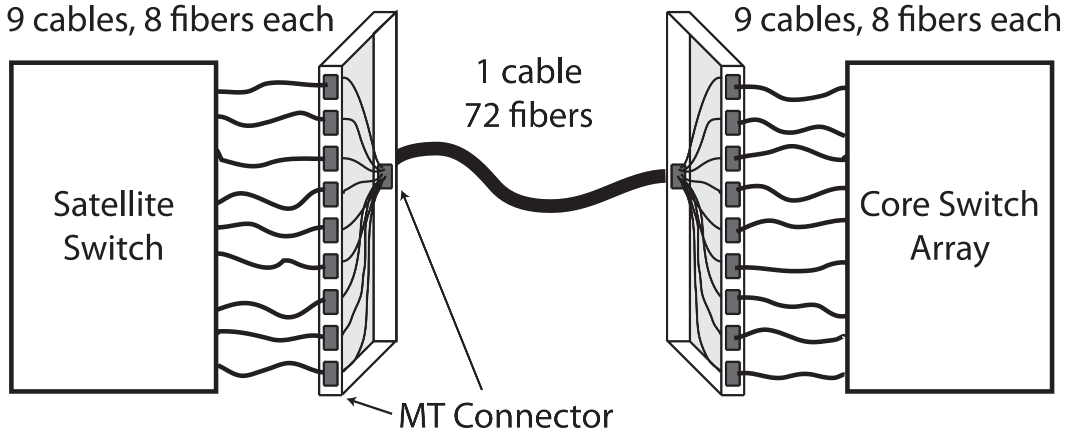


Figure 2.8: Fiber management units.

EEP protocol at 40 Gb/s.

Experience from a prototype system indicates that most exceptional packet processing happens either on the line card or in the core switch array, rather than on the uplink card. For this reason, a dedicated CPU is deliberately omitted from the uplink card. Instead, all 6 uplink cards are managed by system-level CPU located on the midplane. It is also possible to place a CPU on each uplink card with a modest increase in cost.

2.4.5 Fiber Management

QSFP modules use a high-density fiber connector called an MT-connector. QSFP specifically uses an MT-connector with 8 parallel fibers and collects them into one cable. Since each pod switch can support 36 such QSFP modules, this would mean 36 parallel 8-fiber cables that must be routed to the core switch array.

Cable management is simplified by aggregating nine 8-fiber cables into a single 72-fiber cable by using a mechanical device called a fiber management unit, shown in Figure 2.8. Such devices are commercially available. This reduces the 36 parallel 8-fiber cables into 4 parallel 72-fiber cables.

A 72-strand fiber cable from a pod switch is split into nine separate 8-fiber cables, corresponding to the pod switch's nine separate QSFP modules. Each 8-fiber cable terminates at a different core card. Just like the pod switch midplane, this is another horizontal-vertical arrangement, but performed with cables.

2.4.6 Core Switch Array

The core switch array contains 18 independent core switch array cards, partitioned into two modules with 9 cards each. The choice of core switch array packaging is largely arbitrary since the 144 separate 24-port switches do not communicate with each other directly. The cards do not connect to a backplane and their co-location is merely a matter of simplifying cable management and packaging. The absence of a backplane greatly reduces the cost of the core switch array and allows our design to scale to even larger FatTree switches in the future.

Figure 2.9 shows the layout of a core switch array card. Each card essentially replaces eight discrete 24-port switches from the core layer of the network. Each card contains a CPU, 8 switch ASICs, 48 EEP ASICs, and 48 QSFP modules. Figure 2.9 shows 8 traces from two EEP ASICs connecting to the 8 different switch ASICs. Although there are a total of 192 connections between the EEP ASICs and the switch ASICs, only 8 signal layers are required, one per switch ASIC. A 9th signal layer is required for the CPU to connect to the switch ASICs. Current circuit board manufacturing techniques make it difficult to design boards with more than 10 signal layers, so this layout is the most efficient in terms of board area utilization without becoming an engineering challenge. More exotic designs can double the number of signal layers to 20, but the engineering challenge increases correspondingly.

Figure 2.10 shows the internal topology of the 3,456-port switch. Only one of the 24 pod switches is shown. When fully deployed, each ASIC on each line card connects to each ASIC on each uplink card, and each pod switch connects to each of the 144 core switch ASICs. Each uplink card connects to exactly three core switch array cards, for a total of 18 core switch array cards in a fully deployed system. The fact that this is a small, finite number is useful for scaling to larger FatTree switches in the future. In order to achieve full bisection bandwidth, each pod switch must connect to each of the 144 core switch ASICs. If a mistake is made when constructing the network, a pod switch might end up with two or more

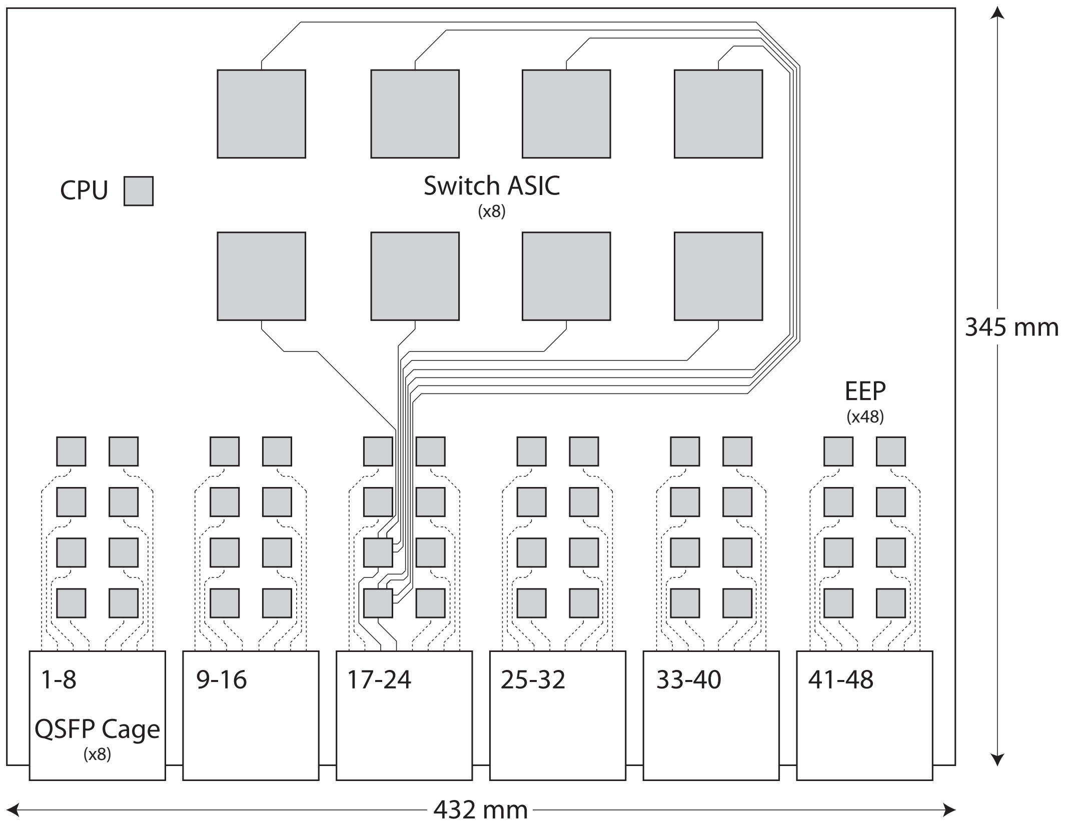


Figure 2.9: Core switch array card.

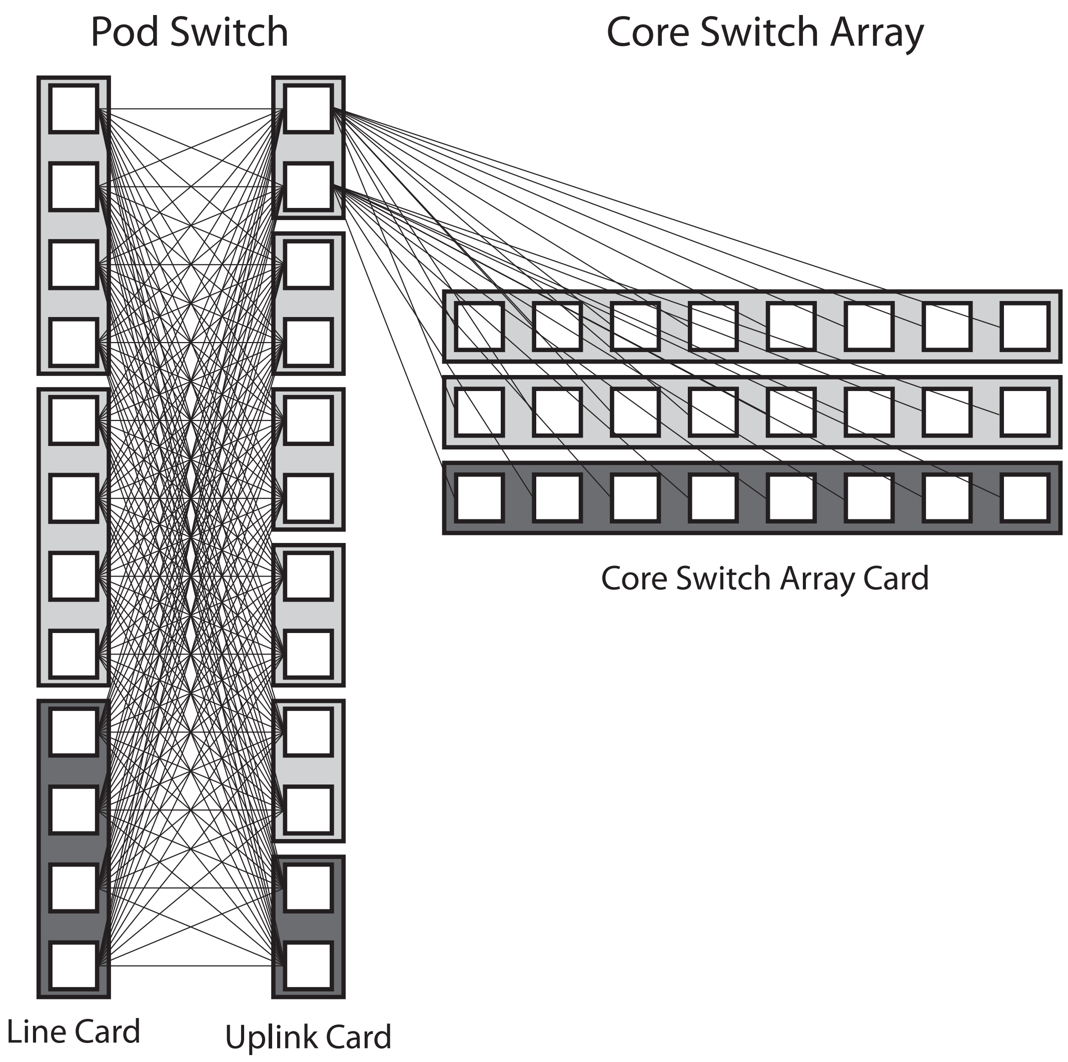


Figure 2.10: Topology of the 3,456-port switch.

parallel links to the same core switch ASIC, which would subtract from the overall bisection bandwidth.

By design, the EEP protocol helps prevent installation mistakes. The four links leaving an ASIC on a pod switch uplink card travel to a single core switch array card, where they connect to four *different* core switch ASICs. The core switch array card is designed such that the top row of 24 QSFP modules connect only to the top row of switch ASICs, and the bottom row of QSFP modules only to the bottom row of switch ASICs. Therefore, as long as each pod switch connects to *exactly* one upper and one lower port on each core switch array card, the network will achieve full bisection bandwidth.

2.4.7 Incremental Deployment

The 3,456-port switch was designed to support incremental deployment. A very small network can start with a single pod switch and no core switch array. Each pod switch line card can be installed as more ports are needed. However, all 6 uplink cards must be installed for non-interfering throughput. Uplink cards can also be installed incrementally to reduce cost or if some degree of oversubscription can be tolerated. Each uplink card provides 1/6 of the total aggregate pod switch bandwidth. For a single pod switch, the QSFP modules are not needed.

A single pod switch can provide up to 1.44 Tb/s of bisection bandwidth. Adding a second pod switch then requires the installation of the core switch array. However, if fault tolerance is not a concern, then it is possible to install just one of the two core switch array modules. Also, if some degree of oversubscription can be tolerated, then not all nine core switch array cards need be installed. Each core switch array card provides 1/18 of the total aggregate core switch array bandwidth.

One core switch array module can support up to 12 pod switches with no oversubscription. In this deployment, each pod switch will connect all of its uplink cables to the same core switch array module, rather than splitting them between the two modules. Once the 13th pod switch is installed, both core switch array modules will be required. The four 72-strand fiber cables from each of the pod switches must be distributed evenly among the core switch arrays. This could require disconnecting and reconnecting a small number of cables.

This upgrade could be performed while the switch is operational, without dropping any packets. The idea is to use the management interface to take certain pod switch uplink cards offline, migrate the cables, and then reactivate the uplink cards. As long as only a few uplink cards are disabled at any one time, the overall network should maintain connectivity but with a slightly reduced amount of bisection bandwidth.

2.5 Comparison

In this section, the design from section 2.4 is compared with two other possible implementations of a 3,456-port 10 GbE FatTree switch. These are referred to as Network 1, 2, and 3, respectively.

Network 1. This network can be constructed immediately without any hardware engineering. It is comprised of 720 discrete 24-port 10G TOR switches (see TOR_{10} in Table 2.1) connected in the same FatTree topology described in section 2.4. All inter-switch links use SFP+ optical transceivers and multimode fiber cable.

Network 2. This network requires board and chassis design. Instead of discrete TOR switches, the boards and chassis from section 2.4 are used. However, Network 2 does not use EEP or 40 GbE QSFP modules. Instead, Network 2 simply replaces these components with additional 10 GbE PHYs and SFP+ modules.

Network 3. This network is precisely the design described in section 2.4. It has the highest non-recurring engineering (NRE) costs since it requires board, chassis, and ASIC design. However, the higher NRE costs translate into lower capital and operational costs.

In order to eliminate non-comparable costs such as software engineering and profit margins, all three designs are reduced to their major hardware components, namely chips and optical transceivers. Certain components, such as circuit boards, power supplies, fans, chassis, and cables, are excluded from this analysis as they do not contribute more than 10% of the total cost or power consumption.

The rationale for EEP comes from the properties of optical transceivers. From Table 2.4, it is actually less expensive in terms of cost, power consumption, and physical area to aggregate to higher-rate links, and then to disaggregate back to the original rate. Higher-rate transceivers are more efficient both in cost and in power consumption. The prices shown here are estimates provided to us by a large OEM manufacturer, and these prices will change over time.

Table 2.5 lists part counts for the three networks. Simply by repackaging the components, a significant number of SFP+ modules can be eliminated. Adding EEP and QSFP modules can further reduce part counts.

Table 2.4: Comparison of three optical transceiver standards.

	SFP	SFP+	QSFP
Rate	1 Gb/s	10 Gb/s	40 Gb/s
Cost/Gb/s	$35	$25	$15
Power/Gb/s	500mW	150mW	60mW

Table 2.5: Part counts.

	Network 1	Network 2	Network 3
ASIC	720	720	720
CPU	720	132	132
PHY	17,280	17,280	10,368
SFP+	17,280	10,368	3,456
EEP	0	0	1,728
QSFP	0	0	1,728

Table 2.6 lists cost and power consumption estimates for the different parts. Power consumption estimates are obtained from data sheets and product briefs. Cost estimates, which are obtained from distributors, necessarily represent a snapshot in time (circa 2009), so current pricing is recommended rather than relying on these numbers.

Table 2.6: Part costs and power estimates.

Part	Cost ($)	Power (W)
ASIC	410	22
CPU	130	8
PHY	10	0.8
SFP+	250	1
EEP	10	2
QSFP	600	2.5

Table 2.7 compares the three networks. Although all three networks have equivalent bisection bandwidth, Network 2 is strictly better than Network 1 and Network 3 is strictly better than Network 2. This does not account for one-time NRE costs, which will be larger for Network 3 than for Network 2 due to the EEP ASIC. Table 2.7 shows that the largest gains come from repackaging the FatTree. The EEP ASIC also provides a significant improvement.

Table 2.7 shows that optical transceiver modules account for 21% of the

Table 2.7: Comparison of three equivalent FatTree networks.

	Network 1	Network 2	Network 3
Bisection Bandwidth (Tb/s)	34.56	34.56	34.56
Cost ($M)	4.88	3.07	2.33
Power Consumption (kW)	52.7	41.0	36.4
Cabling Complexity	3,456	96	96
Space (Rack Units)	720	192	114

overall power consumption and 81% of the overall cost. The design would benefit greatly from less expensive optical transceiver technology. Two companies, Infinera [23] and Luxtera [24] are developing less expensive optical transceivers by utilizing recent advances in photonic integrated circuits that allow multiple optical transceivers to be fabricated on a single chip [25]. For example, Luxtera uses their chips to build a 40 Gb/s active cable, which is a fiber cable with transceivers permanently attached to both ends.

2.6 EEP: Ethernet Extension Protocol

This section describes the design of the EEP ASIC, an Ethernet traffic groomer. A traffic groomer aggregates frames from multiple low-rate links and tunnels them over a single higher-rate link. There are two existing standards for traffic grooming: SONET/SDH and IEEE 802.1ad. SONET/SDH was designed for transporting both voice and data traffic over long distances of optical fiber while also supporting QoS and providing very low jitter. IEEE 802.1ad[26] (VLAN Tunneling) bridges multiple remote Ethernet networks connected to a single carrier network. The additional functionality provided by SONET/SDH is not needed for this simple task.

At the same time, IEEE 802.1ad requires an entire Ethernet frame to be stored on chip before forwarding it through a higher-rate port. This is a limitation of the Ethernet protocol, which does not allow frame fragmentation. Store-and-forward can become a significant source of latency, especially when used in a multi-stage switch such as a FatTree.

A lightweight protocol, called EEP, was designed to eliminate the latency and buffering requirements of IEEE 802.1ad. Although EEP is new, the ideas are drawn from well-known techniques in other synchronous protocols such as ATM. EEP is very simple, requires no configuration, and is completely invisible to the rest of the network. This means that EEP is compatible with current data center bridging efforts such as IEEE 802.1Qau, 802.1Qaz, and 802.1Qbb.

Section 2.6.1 presents an IEEE 802.1ad compatible traffic groomer design. Section 2.6.2 presents a novel EEP traffic groomer design that offers lower latency and lower buffer requirements.

2.6.1 IEEE 802.1ad Traffic Groomer Design

This design is fully compatible with the IEEE 802.1ad specification for VLAN tagging. The design is divided into two independent datapaths. Figure 2.11 shows the grooming datapath. Ethernet frames arrive at the 10 GbE ports, where VLAN tags are added before entering a FIFO input queue. Concurrently, a round-robin arbiter monitors the input queues until it finds one that has a queued frame. The frame is removed from the queue and transmitted through the 40 GbE MAC. When the 40 GbE MAC finishes transmitting a frame, the arbiter finds the next frame to be transmitted.

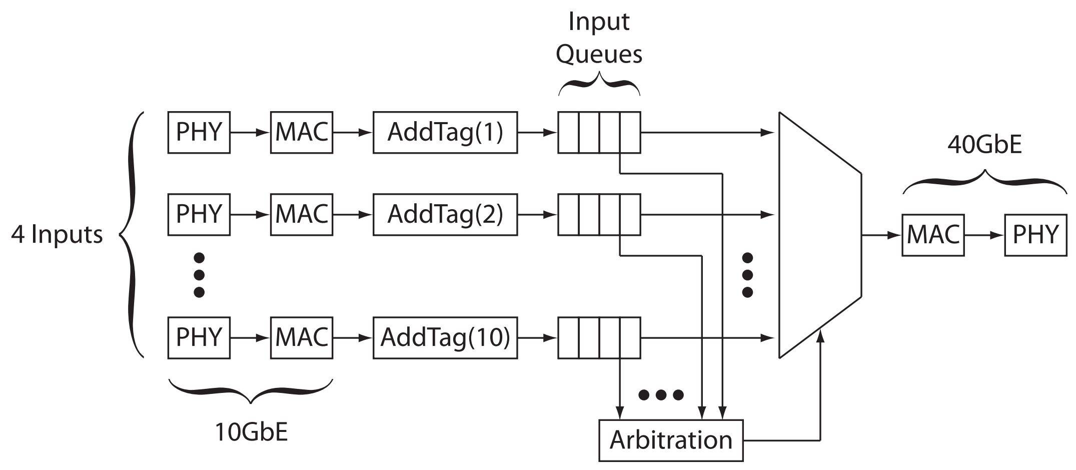


Figure 2.11: IEEE 802.1ad grooming datapath.

Figure 2.12 shows the degrooming datapath. Notice that the degrooming datapath does not use an arbiter. As a frame arrives at the 40 GbE port, the VLAN tag is removed and examined. The value of the VLAN ID controls a decoder which selects the particular FIFO output queue in which to place the frame. Concurrently, each 10 GbE MAC waits for data to be placed in its FIFO and then transmits the frame.

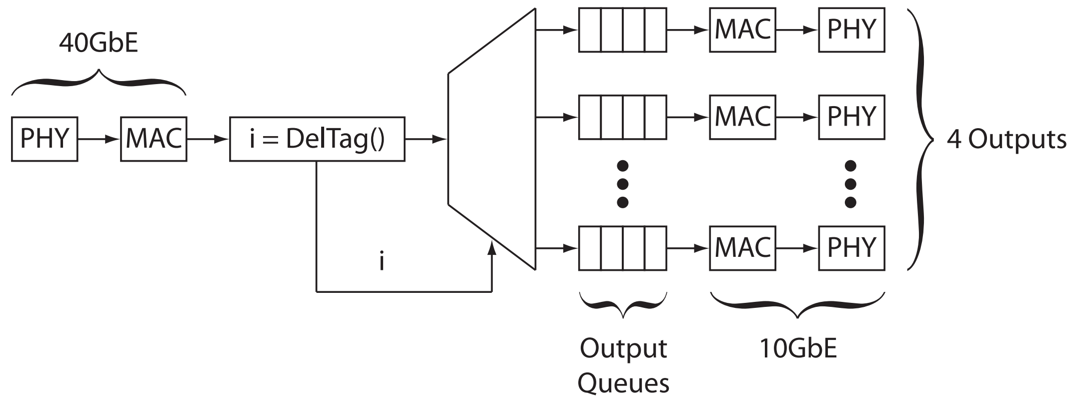


Figure 2.12: IEEE 802.1ad degrooming datapath.

This design is compatible with frames that already contain VLAN tags because existing tags are treated like payloads.

2.6.2 EEP Traffic Groomer Design

EEP provides the abstraction of a set of 16 virtual Ethernet links, multiplexed over a single physical Ethernet link. EEP breaks up an Ethernet frame into a sequence of 64B segments, and encapsulates each segment into an EEP frame. Each EEP frame originating from the same switch port is assigned the same Virtual Link ID, similar to a VLAN tag.

Figure 2.13 shows the EEP frame format. The very first EEP frame in the sequence (corresponding to the first 64B of an Ethernet frame) has the SOF (Start of Frame) bit set. All other EEP frames in the sequence have the SOF bit cleared. The last EEP frame has the EOF (End of Frame) bit set. The last EEP frame will likely not have a 64B payload. In this case, the LEN (Length) bit is set, which indicates that the second byte in the EEP frame is part of the header rather than the payload. This second header byte records the number of valid bytes in the payload. In the common case, each EEP frame will use one header byte. Only final EEP frames use two header bytes. Three unused bits are reserved for future use.

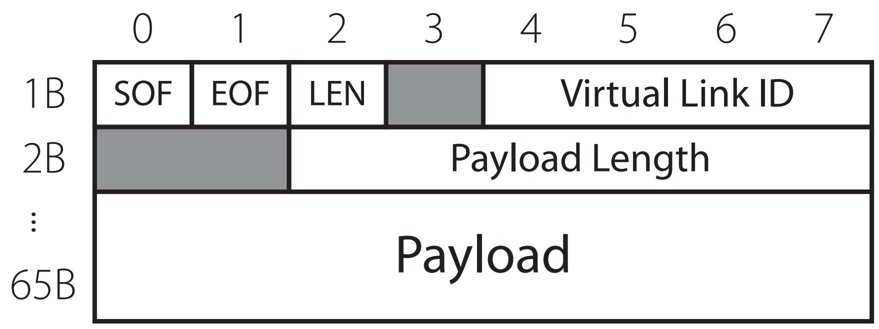


Figure 2.13: EEP frame format.

Although IEEE 802.1ad is limited to store-and-forward, EEP can use cut-through switching to begin transmitting an Ethernet frame on an output port after the first EEP frame has been received. However, one must be careful when using cut-through switching. The EEP receiver should never run out of bytes part way through the transmission of an Ethernet frame. This will corrupt the frame. This scenario can be avoided if the EEP transmitter services the input queues in a timely and round-robin fashion. If an EEP frame happens to get dropped and omitted from a reassembled packet, the frame check sequence (i.e., checksum) will invalidate the larger Ethernet frame, since it is passed on as part of the EEP payload.

One drawback to EEP is that it adds overhead to the link. A standard 1500B Ethernet payload actually requires 1538B for transmission, once the preamble, header, frame check sequence, and inter-frame gap are accounted for. EEP will add an additional 25B of headers to create 24 EEP frames from the original Ethernet frame. Normally, this would reduce the capacity of the link by 1.6%, a small but annoying amount. However, the PHYs can be overclocked relative to the switching fabric and the optical transceivers by at least 1.6% to recover the lost capacity. This technique of overclocking the PHYs to provide additional capacity for a control-plane is frequently used in both proprietary and merchant silicon.

2.6.3 Implementation

Both the IEEE 802.1ad and EEP traffic groomers are implemented in hardware using a Xilinx Virtex-5 LX110T FPGA. Due to limited resources, the implementations are simplified to use four GbE ports and one 10 GbE port rather than four 10 GbE ports and one 40 GbE port.

The custom logic totaled 3,480 lines of Verilog-2001. Xilinx IP was used for the PHY, MAC, and FIFO cores. The designs are verified with a custom test bench totaling 1,200 lines of SystemVerilog-2005 and simulated with ModelSim 6.4. Xilinx XST 10.1 was used for logic synthesis.

Table 2.8 shows how much FPGA resources are used for both designs. The EEP implementation is slightly smaller because cut-through switching was used to reduce the buffer size. The implementations are small enough to fit into Xilinx's second smallest LXT device, indicating that an ASIC conversion would produce an extremely inexpensive and low-power chip.

Table 2.8: Xilinx Virtex-5 device utilization.

	IEEE 802.1ad	EEP
Flip Flops	5,427 (7%)	4,246 (6%)
LUTs	6,243 (9%)	5,004 (7%)
BlockRAM/FIFOs	11 (7%)	0 (0%)

Some merchant silicon vendors offer custom ASIC design services, where they will integrate customer logic and IP into one of their existing devices. For example, Broadcom offers this service. One could imagine integrating EEP and other PHYs directly onto an existing 24-port 10 GbE switch ASIC, to further reduce cost and power. However, such an approach is outside the scope of this paper.

2.6.4 Evaluation

Figure 2.14 shows the round-trip latency of both implementations. Because of store-and-forward, IEEE 802.1ad adds latency proportional to the size of the Ethernet frame. EEP adds a constant amount of latency regardless of frame size.

These measurements are obtained using a NetFPGA [27] configured for packet generation and capture. Both implementations are verified to work correctly under heavy load without dropping or corrupting frames. The IEEE 802.1ad implementation was also verified to interoperate with other Ethernet devices.

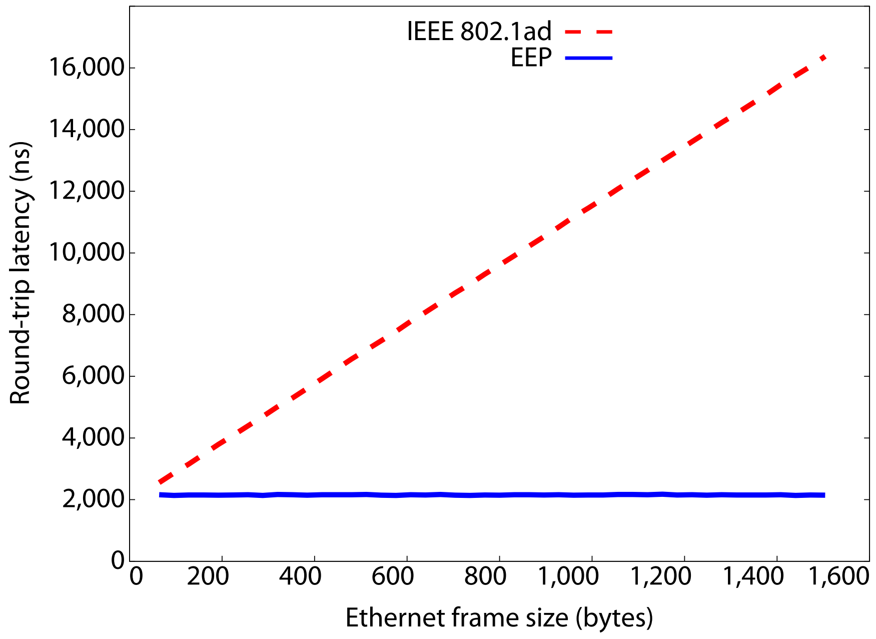


Figure 2.14: Traffic groomer latency measurements.

Figure 2.15 shows the oversubscription caused the IEEE 802.1ad and EEP protocols, as a function of Ethernet payload size. IEEE 802.1ad VLAN tunneling has a fixed 4B overhead, regardless of frame size. This can cause some congestion on the 10 GbE link if all the GbE links are fully utilized. The effect is more significant for small frames. In the case of EEP, overhead is added by the 1-2 byte EEP header. However, we can reduce overhead by eliminating the unnecessary 7-byte preamble and by grouping EEP frames together to minimize the overhead of Start of Frame bytes and the inter-frame gap (IFG). Because of this, we can avoid oversubscription for payloads below 943 bytes.

Figure 2.16 shows the minimum amount of buffer memory required by both the IEEE 802.1ad and EEP traffic groomers, as a function of the number of ports N. In the worst case, all N ports could begin receiving maximum-sized frames simultaneously. Then after the 12B IFG, it could happen again. In both designs,

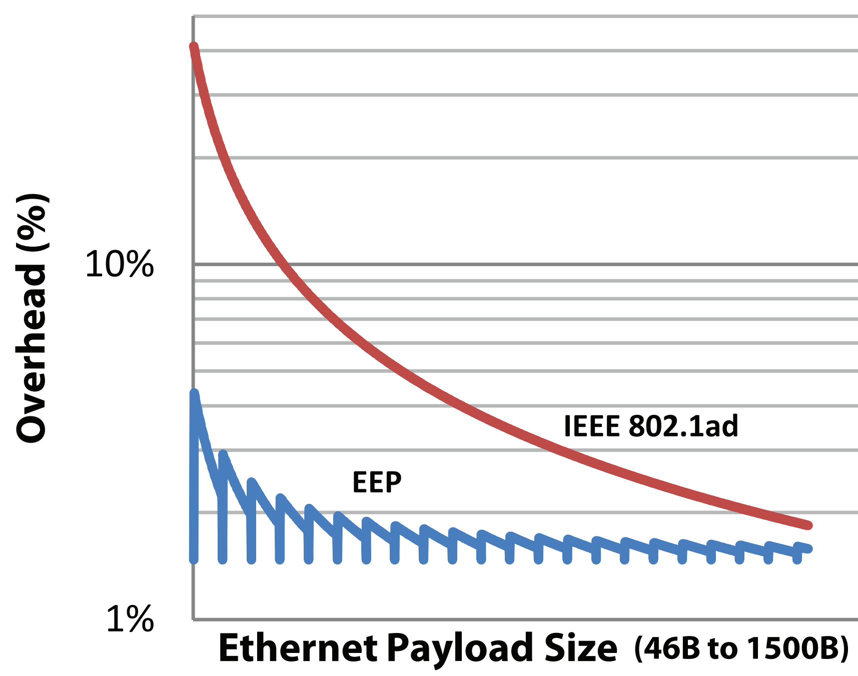


Figure 2.15: Traffic groomer oversubscription.

by the time the second round of frames has finished being received, the first round should have been transmitted. Therefore, any traffic groomer needs no more than $2 \cdot N \cdot$ MTU bytes of buffer memory, where MTU is the maximum transmission unit.

When $N = 10$, the IEEE 802.1ad traffic groomer would require 29.7 KB of buffer memory for regular 1500B payloads and 176 KB of buffer memory for 9 KB jumbo frames. EEP would only require 1.3 KB of buffer memory, due to its 68B MTU. For this reason, a hardware implementation of EEP would be less expensive than for IEEE 802.1ad, especially when supporting jumbo frames. This will be even more true as bit rates increase to 40 Gb/s and 100 Gb/s.

2.7 64-port Switch ASICs

With the rising popularity of containerized data centers (pods) and the emergence of 64-port 10 GbE switch ASICs, it is possible to build a 1,024-port 10 GbE "pod switch" with 10.24 Tb/s of uplink capacity to the core switch array

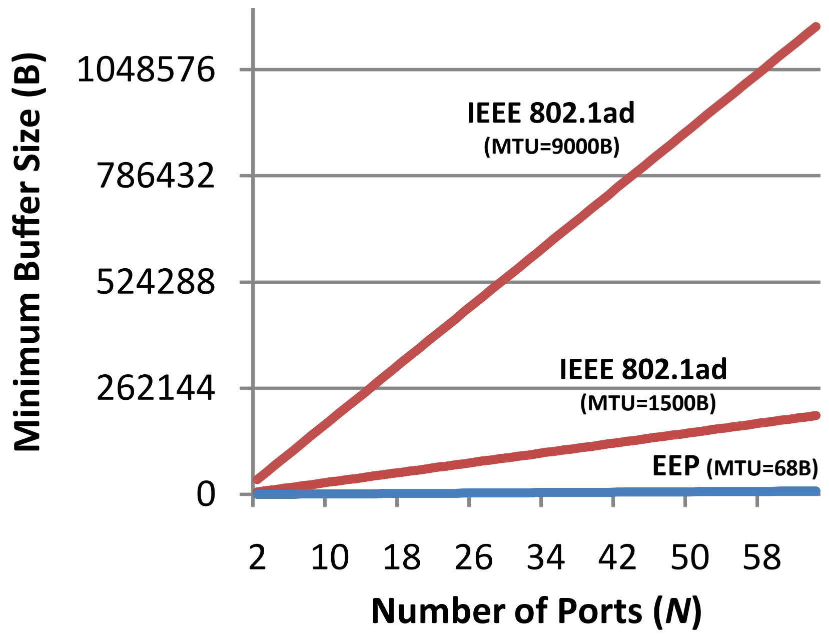


Figure 2.16: Traffic groomer minimum buffer sizes.

(Figure 2.17). This modular switch can support up to 16 line cards, each providing 64 SFP+ cages, and 8 uplink cards, each with 1.28 Tb/s of uplink capacity. The horizontal-vertical arrangement is used on the midplane to connect each switch ASIC on each linecard to each switch ASIC on each uplink card. In this example, EEP performs traffic grooming from eight 10 GbE channels to a single 80 Gb/s EEP channel. A fully-constructed data center network would then have 64 pods, each containing 1,024 hosts, and the data center network would have 655.36 Tb/s of capacity.

2.8 Related Work

FatTree Networks: FatTrees have been used for decades to construct HPC interconnects. Perhaps the most famous example is the Thinking Machines CM-5 [14]. The CM-5 used proprietary 8-port switch ASICs with a custom frame format. The bottom two tiers of the FatTree are connected on circuit boards and the upper tiers used copper cables. A fully deployed CM-5 covered an area of

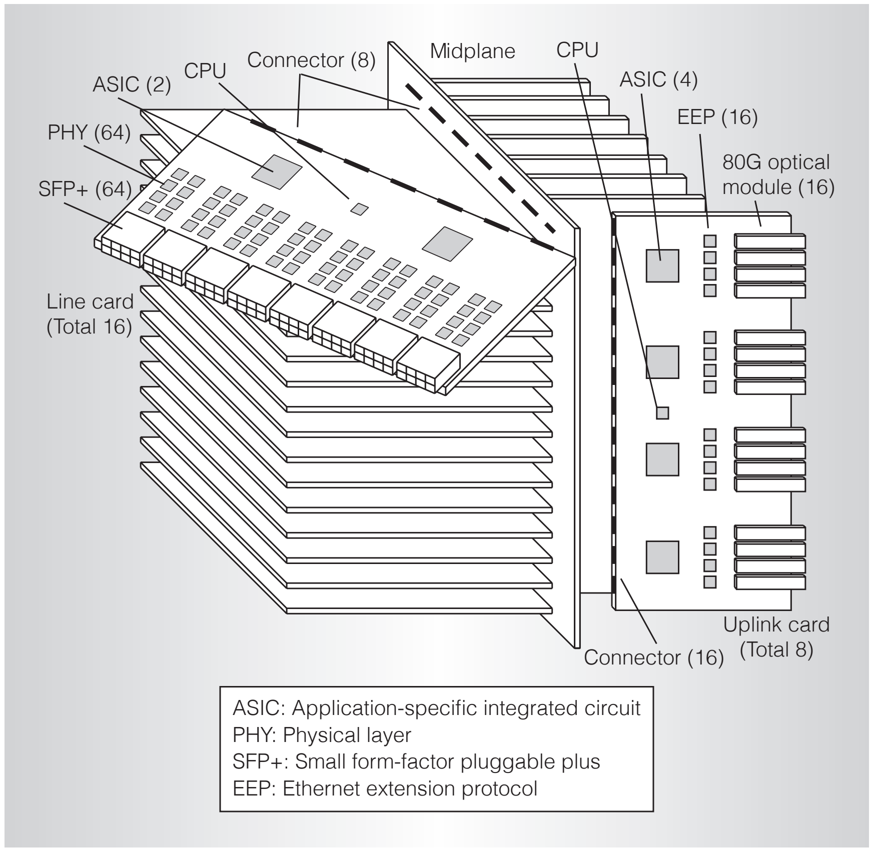


Figure 2.17: 1,024-port pod switch with 64-port switch ASICs.

900 m^2.

The Folded Clos topology [21] is often referred to as a FatTree, and this chapter continues to use the label of "FatTree" to refer to the Folded Clos.

Al-Fares et al. [11] was the first to suggest constructing a large Layer 3 DCN from a 3-tier FatTree using commodity TOR switches. It overcomes the limitations of ECMP multipath routing with both static and dynamic techniques. Its static multipath algorithm relies on a novel use of TCAMs; although it is possible to implement the same search using hashing and commodity SRAM. The two different dynamic routing techniques are both simple and inexpensive to implement, and perform better than static routing.

PortLand [28] and VL2 [12] describe large Layer 2 Ethernet data center networks. The two principal challenges are to limit broadcast traffic and to allow for multipath routing. PortLand is a native Layer 2 network and translates Ethernet MAC addresses inside the network. This feature is supported by all high-performance merchant silicon evaluated in this chapter. VL2 uses IP-in-IP encapsulation, which adds at least 20B of overhead to each packet and requires end-host operating system modifications.

FatTree Switches: Network equipment vendors have only recently begun building switches from merchant silicon using a multi-stage FatTree topology internally. The Arista 7148SX [29] is a 48-port 10 GbE TOR switch with six Fulcrum switch ASICs connected into a 2-tier FatTree on a single board. The larger Woven EFX-1000 [30] is a 144-port 10 GbE modular switch with 18 Fulcrum switch ASICs connected in a 2-tier FatTree. Twelve line cards each contain twelve 10 GbE ports and one ASIC. Six additional fabric cards each contain one ASIC. A midplane connects the line cards to the fabric cards. Both of these designs seek to replace and individual TOR or EOR switch with a FatTree of smaller switch ASICs. The design seeks to replace an entire data center network.

The Sun Microsystems 3,456-port InfiniBand switch [31] is constructed as a 3-tier FatTree of 720 Mellanox InfiniScale III [32] 24-port switch ASICs. Unlike our design, everything is packaged into a single chassis. Each of the 24 line cards contains 24 switch ASICs and each of the 18 fabric cards contains 8 switch ASICs.

A midplane connects the line cards to the fabric cards using 432 connectors with 64 pin pairs each. One advantage of this monolithic design is that it eliminates expensive optical transceivers since all chips are close enough to communicate over copper circuit board traces. However, the high density leads to a cable management problem; proprietary splitter cables are used to aggregate three 10 Gb/s InfiniBand channels over one cable. The monolithic design also makes it difficult to support next-generation higher-density switch ASICs. While omitted for brevity, our pod-based design should generalize to much larger networks, for example 65,536 ports using 64-port 10 GbE merchant silicon. In this case, each of the 64 pod switches will support 1,024 ports.

2.9 Discussion

This chapter showed that the construction of FatTree networks from discrete packet switches is not a scalable solution. The design of a 3,456-port 10 GbE switch using a FatTree topology of merchant silicon switch ASICs, was presented, as part of a broader architecture, to overcome the packaging problems of a network of discrete switches. The design provides 34.56 Tb/s of bisection bandwidth when fully deployed.

This chapter showed how an Ethernet Extension Protocol (EEP) could further reduce the cost, power consumption, and cabling complexity of such a switch by aggregating multiple lower speed links onto a single higher-speed link. EEP was compared to the traditional IEEE 802.1ad standard for Ethernet traffic grooming, showing that EEP is superior in terms of latency and on-chip buffering.

Packaging and cabling is only one aspect of constructing a large FatTree-based Ethernet switch. There are also other issues which must be addressed such as managing the forwarding tables of individual switching elements, handling ARP and other broadcast traffic in a large Layer 2 domain, efficient multicast, fault tolerance, and routing TCP flows through the switch fabric. Researchers are currently working on these related problems.

Acknowledgements

Chapter 2, in part, reprints material as it appears in the paper titled "Data Center Switch Architecture in the Age of Merchant Silicon", published in the Proceedings of the 17th IEEE Symposium on High-Performance Interconnects, August 2009, by Nathan Farrington, Erik Rubow, and Amin Vahdat [1].

Chapter 2, in part, reprints material as it appears in the paper titled "Scale-Out Networking in the Data Center", published in IEEE Micro, 30(4), pp 29–41, July/August 2010, by Amin Vahdat, Mohammad Al-Fares, Nathan Farrington, Radhika Niranjan Mysore, George Porter and Sivasankar Radhakrishnan [2].

Chapter 3

Millisecond-scale Optical Circuit Switching

The basic building block of ever larger data centers has shifted from a rack to a modular container with hundreds or even thousands of servers. Delivering scalable bandwidth among such containers is a challenge. A number of recent efforts promise full bisection bandwidth between all servers, though with significant cost, complexity, and power consumption. This chapter presents Helios, a hybrid electrical/optical switch architecture that can deliver significant reductions in the number of switching elements, cabling, cost, and power consumption relative to recently proposed data center network architectures. This chapter explores architectural trade offs and challenges associated with realizing these benefits through the evaluation of a fully functional Helios prototype.

3.1 Introduction

The past few years have seen the emergence of the modular data center [33], a self-contained shipping container complete with servers, network, and cooling, which many refer to as a *pod*. Organizations like Google and Microsoft have begun constructing large data centers out of pods, and many traditional server vendors, such as HP and IBM, now offer products in this space. Pods are now the focus of systems [34] and networking research [35]. Each pod typically holds between

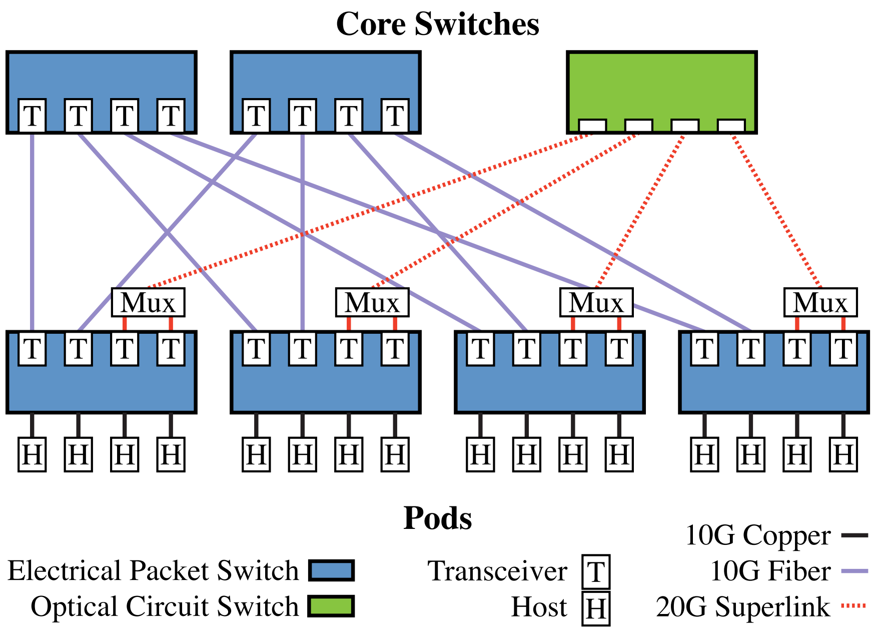


Figure 3.1: Helios is a 2-level FatTree of pod switches and core switches.

250 and 1,000 servers. At these scales, it is possible to construct a non-blocking switch fabric to interconnect all of the servers within an individual pod. However, interconnecting hundreds to thousands of such pods to form a larger data center remains a significant challenge.

Supporting efficient inter-pod communication is important because a key requirement in data centers is flexibility in placement of computation and services. For example, a cloud computing service such as EC2 may wish to place the multiple virtual machines comprising a customer's service on the physical machines with the most capacity irrespective of their location in the data center. Unfortunately, if these physical machines happen to be spread across multiple pods, network bottlenecks may result in unacceptable performance. Similarly, a large-scale Internet search engine may run on thousands of servers spread across multiple pods with significant inter-pod bandwidth requirements. Finally, there may be periodic bandwidth requirements for virtual machine backup or hotspots between partitioned computation and storage (e.g., between EC2 and S3).

In general, as services and access patterns evolve, different subsets of nodes within the data center may become tightly coupled, and require significant bandwidth to prevent bottlenecks. One way to achieve good performance is to statically provision bandwidth between sets of nodes with known communication locality. Unfortunately, given current data center network architectures, the only way to provision required bandwidth between dynamically changing sets of nodes is to build a non-blocking switch fabric at the scale of an entire data center, with potentially hundreds of thousands of ports.

While recent proposals [11, 12, 36, 35] are capable of delivering this bandwidth, they introduce significant cost and complexity. Worse, full bisection bandwidth at the scale of an entire data center is rarely required, even though it is assumed to be the only way to deliver on-demand bandwidth between arbitrary hosts in the face of any localized congestion. One common approach to bring down networking cost and complexity is to reduce the capacity of the inter-pod network by an oversubscription ratio and to then distribute this capacity evenly among all pods [10]. For a network oversubscribed by a factor of ten, there will be bandwidth bottlenecks if more than 10% of hosts in one pod wish to communicate with hosts in a remote pod.

Optical circuit switching and wavelength division multiplexing (WDM) are promising technologies for more flexibly allocating bandwidth across the data center. A single optical port can carry many multiples of 10 Gb/s assuming that all traffic is traveling to the same destination. The key limitation is switching time, which can take as long as tens of milliseconds. Thus, it makes little sense to apply optical switching at the granularity of hosts that transmit no faster than 10 Gb/s and that carry out bursty communication to a range of hosts. However, the pod-based design of the data center presents the opportunity to effectively leverage optical switching due to higher stability in the pod-level aggregated traffic demands. When there is a lot of burstiness of aggregated traffic demand between different pairs of pods, the number of circuits required to support this bursty communication in the data center would be prohibitive. Such bursty communication is better suited to electrical packet switching.

This chapter thus proposes Helios, a hybrid electrical/optical data center switch architecture capable of effectively combining the benefits of both technologies, and delivering the same performance as a fully-provisioned packet-switched network for many workloads but at significantly less cost, less complexity (number of cables, footprint, labor), and less power consumption. Helios identifies the subset of traffic best suited to circuit switching and dynamically reconfigures the network topology at runtime based on shifting communication patterns. Helios requires no modifications to hosts and only straightforward software modifications to switches. Further, the electrical and optical interconnects in the core array can be assembled from existing commercial products.

A fully functional Helios prototype has been constructed using commercial 10 GigE packet switches and MEMS-based optical circuit switches. Section 3.2 describes how copper links are no longer viable for 10 Gb/s links beyond interconnect distances of 10 m. This gap must be filled by optical interconnects, with new trade offs and opportunities. Section 3.3 extrapolates Helios to a large-scale deployment and find an opportunity for significant benefits, e.g., up to a factor of 3 reduction in cost, a factor of 6 reduction in complexity, and a factor of 9 reduction in power consumption. Section 3.4 describes a control algorithm for measuring and estimating the actual pod-level traffic demands and computing the optimal topology for these demands, irrespective of the current network topology and the challenges it imposes on demand estimation. Section 3.5 finds that the appropriate mix of optical and electrical switches depends not only on the volume of communication but on the specific communication patterns. And Section 3.6 describes the insights gained from building Helios. Section 3.7 identifies hardware requirements for improving the performance of Helios and other hybrid electrical/optical data center networks.

3.2 Background and Motivation

This section discusses the problem of oversubscription, and how existing data center network architectures can only solve that problem through over-provisioning, i.e., providing enough bisection bandwidth for the worst case, not the common case. Next, the technologies such as optical circuit switches and wavelength division multiplexing that make Helios possible are described.

3.2.1 The Oversubscription Problem

To make the discussion concrete, first consider the architecture of a large scale data center built from pods. Each pod has 1,024 servers and each server has a 10 GigE NIC. It is assumed that the task of interconnecting individual servers within a pod is largely solved with existing [37] and soon-to-be-released dense 10 GigE switches. A number of vendors have released commodity 64-port 10 GigE switches on a chip at price points in the hundreds of dollars per chip. Laying these chips out in a fully-connected mesh [1, 38] makes it possible to build a non-oversubscribed modular pod switch with 1,024 10 GigE server-facing ports and 1,024 10 GigE uplinks for communication to other pods, by way of a *core* switching layer. Bandwidth through the core may be limited by some *oversubscription ratio*.

A number of recent architectures [12, 11, 36, 35] can be employed to provide full bisection bandwidth among all hosts, with an oversubscription ratio of 1. For instance, a core switching layer consisting of 1,024 64-port 10 GigE switches could provide non-blocking bandwidth (655 Tb/sec) among 64 pods (each with 1,024 10 GigE uplinks) using 65,536 physical cables between the 64 pods and the core switching layer. The cost and complexity of deploying such a core switching layer and the relatively rare need for aggregate bandwidth at this scale often leads to oversubscribed networks. In the running example, a pod switch with 100 10 GigE uplinks would mean that server communication would be oversubscribed by a factor of 10, leading to 1 Gb/sec for worst-case communication patterns. One recent study of data center network deployments claimed an oversubscription ratio of 240 for GigE-connected hosts [12].

There is an interesting subtlety regarding fully provisioned inter-pod data center network topologies. It can be claimed that provisioning full bandwidth for arbitrary all-to-all communication patterns at the scale of tens of thousands of servers is typically overkill since not many applications run at such scale or have such continuous communication requirements. However, it is worth noting that a fully provisioned large-scale topology is required to support localized bursts of communication even between two pods as long as the set of inter-communicating pods is not fixed. In the example data center with an oversubscription ratio of 10, the topology as a whole may support 65 Tb/s of global bisection bandwidth. However, if at a particular point in time, all hosts within a pod wish to communicate with hosts in remote pods, they would be limited to 1 Tb/s of aggregate bandwidth even when there is no other communication anywhere else in the data center.

The key observation is that the available bisection bandwidth is inflexible. It cannot be allocated to the points in the data center that would most benefit from it, since it is fixed to a pre-defined topology. While supporting arbitrary all-to-all communication may not be required, supporting bursty inter-pod communication requires a non-blocking topology using traditional techniques. This results in a dilemma: unfortunately, network designers must pay for the expense and complexity of a non-blocking topology despite the fact that vast, though dynamically changing, portions of the topology will sit idle if the network designer wishes to prevent localized bottlenecks.

The goal is to address this dilemma. Rather than provision for the worst case communication requirements, consider enabling a pool of available bandwidth to be allocated when and where it is required based on dynamically changing communication patterns. As an interesting side effect, the technologies leveraged can deliver full bisection bandwidth at significantly lower cost, complexity, and energy than existing data center interconnect technologies as long as one is willing to make certain assumptions about stability in communication patterns.

3.2.2 Enabling Technologies

The Trend of Optics in the Data Center

The electronics industry has standardized on silicon as their common substrate, meaning that many VLSI fabs are optimized for manufacturing Si-based semiconductors [39]. Unfortunately for optics, it is not currently possible to fabricate many important optical devices, such as lasers and photodetectors, using only Si. Optics have traditionally used more exotic group III-V compounds like

GaAs and group III-V quaternary semiconductor alloys like InGaAsP [40]. These materials are more expensive to process, primarily because they cannot piggyback on the economies of scale present in the electronics market.

Transceivers for copper cables are Si-based semiconductors, and are ideal for short-reach interconnects. However, the definition of "short" has been trending downward over time. There is a fundamental trade off between the length of a copper cable and available bandwidth for a given power budget [41]. For 10 GigE, this "power wall" limits links to approximately 10 m. Longer cables are possible, though power can exceed 6 W/port, unsustainable in large-scale data centers. Fortunately, 10 m corresponds to the distance between a host and its pod switch. So for the short-term, we assume all host links inside of a pod will be copper.

Large data centers require a significant number of long links to interconnect pods, making it essential to use optical interconnects. Unfortunately, optical transceivers are much more expensive than copper transceivers. For example, an SFP+ 10 GigE optical transceiver can cost up to \$200, compared to about \$10 for a comparable copper transceiver. This high cost is one factor limiting the scalability and performance of modern data centers. For the fully provisioned topology, the 65,536 optical fibers between pods and core switches would incur a cost of \$26M just for the required 131,072 transceivers (not accounting for the switches or the complexity of managing such an interconnect). It is possible that volume manufacturing will drive down this price in the future, and emerging technologies such as silicon nanophotonics [42] may further reduce cost by integrating optical components on standard Si substrates. Despite these industry trends, data center networking has reached a point where the use of optics is required, and not optional.

MEMS-based Optical Circuit Switching

A MEMS-based optical circuit switch (OCS) [43] is fundamentally different from an electrical packet switch. The OCS is a Layer 0 switch — it operates directly on light beams without decoding any packets. An OCS uses an $N \times N$ crossbar of mirrors to direct a beam of light from any input port to any output port. The mirrors themselves are attached to tiny motors, each of which is approximately 1 mm^2 [44]. An embedded control processor positions the mirrors to implement a particular connection matrix and accepts remote commands to reconfigure the mirrors into a new connection matrix. Mechanically repositioning the mirrors imposes a switching time, typically on the order of milliseconds [44].

Despite the switching-time disadvantage, an OCS possesses other attributes that prove advantageous in the data center. First, an OCS does not require transceivers since it does not convert between light and electricity. This provides significant cost savings compared to electrical packet switches. Second, an OCS uses significantly less power than an electrical packet switch. The Glimmerglass OCS consumes 240 mW/port, whereas a 10 GigE switch such as the 48-port Arista 7148SW [38] consumes 12.5 W per port, in addition to the 1 W of power consumed by an SFP+ transceiver. Third, since an OCS does not process packets, it is data rate agnostic; as the data center is upgraded to 40 GigE and 100 GigE, the OCS need not be upgraded. Fourth, WDM (§3.2.2) can be used to switch an aggregate of channels simultaneously through a single port, whereas a packet switch would first have to demultiplex all of the channels and then switch each channel on an individual port.

Optical circuit switches have been commercially available for over a decade. Switches with as many as 320 ports [45] are commercially available, with as many as 1,000 ports being feasible. In this chapter, it is assumed that an OCS port costs $500 and that per-port power consumption is 240 mW. As a point of comparison, the 48-port 10 GigE Arista 7148SX switch has a per-port cost of $500. These values are summarized in Table 3.1.

Wavelength Division Multiplexing

WDM is a technique to encode multiple non-interfering channels of information onto a single optical fiber simultaneously. WDM is used extensively in WAN networks to leverage available fiber given the cost of trenching new cables over long distances. WDM has traditionally not been used in data center networks where fiber links are shorter and the cost of deploying additional fiber is low.

Table 3.1: Component cost and power consumption.

Component	Cost	Power
Packet Switch Port	\$500	12.5 W
Circuit Switch Port	\$500	0.24 W
Transceiver ($w \leq 8$)	\$200	1 W
Transceiver ($w = 16$)	\$400	1 W
Transceiver ($w = 32$)	\$800	3.5 W
Fiber	\$50	0

Coarse WDM (CWDM) technology is less expensive than Dense WDM (DWDM) because it uses a wider channel spacing (20 nm channels across the 1270 nm - 1630 nm spectrum). CWDM lasers do not require expensive temperature stabilization. However, there is little demand for CWDM transceivers because they are not compatible with erbium-doped fiber amplifiers used for long-haul communications [46]. Such amplification is not required for short data center distances. CWDM SFP+ transceivers are practically the same as "standard" 10GBASE-LR 1310 nm SFP+ transceivers, except for a different color of laser. With sufficient volume, CWDM transceivers should also drop to about \$200.

DWDM transceivers have a much narrower channel spacing (0.4 nm) which allows for more than 40 independent channels (in the C-band). They are more expensive because narrow channels require temperature stabilization. Finisar and others are developing DWDM SFP+ modules. An estimate of the cost in volume is approximately \$800.

There are two major technology trends happening in optics. The CWDM and DWDM transceivers mentioned earlier use edge-emitting lasers. There also exists a competing technology called Vertical-Cavity Surface-Emitting Lasers (VCSELs) that are currently less expensive to manufacture and test. Prior research [47] has even demonstrated 4 CWDM channels using VCSELs, and 8 channels may be possible. If multiple channels could be integrated into a single chip, then the cost of deploying Helios might be lower than using edge-emitting laser technology. However, Helios is independent of technology choice and exploration of dense transceiver integration is beyond the scope of this dissertation.

3.3 Architecture

This section presents a model of the Helios architecture and also analyzes cost, power, complexity, and ideal bisection bandwidth trade offs. Some relevant measures of complexity include the human management overhead, physical footprint, and the number of long interconnection cables for the switching infrastructure. Due to the distances involved, it is assumed that all inter-pod 10 GigE links are optical (§3.2.2).

3.3.1 Overview

Figure 3.1 shows a small example of the Helios architecture. Helios is a 2-level FatTree of pod switches and core switches. Core switches can be either electrical packet switches or optical circuit switches; the strengths of one type of switch compensate for the weaknesses of the other type. The circuit-switched portion handles baseline, slowly changing inter-pod communication. The packet-switched portion delivers all-to-all bandwidth for the bursty portion of inter-pod communication. The optimal mix is a trade off of cost, power consumption, complexity, and performance for a given set of workloads.

In Figure 3.1, each pod has a number of hosts (labeled 'H") connected to the pod switch by short copper links. The pod switch contains a number of optical transceivers (labeled "T") to connect to the core switching array. In this example, half of the uplinks from each pod are connected to packet switches, each of which also requires an optical transceiver. The other half of uplinks from each pod switch pass through a passive optical multiplexer (labeled "M") before connecting to a single optical circuit switch. These are called superlinks, and in this example they carry 20G of capacity ($w = 2$ wavelengths). The size of a superlink (w) is bounded by the number of WDM wavelengths supported by the underlying technology. In this chapter, it is assumed that $w \in 1, 2, 4, 8, 16, 32$.

This example delivers full bisection bandwidth. However, we differentiate the bandwidth and say that 50% of the bisection bandwidth is *shared* between pods at packet timescales, and the remaining 50% is *allocated* to particular source-

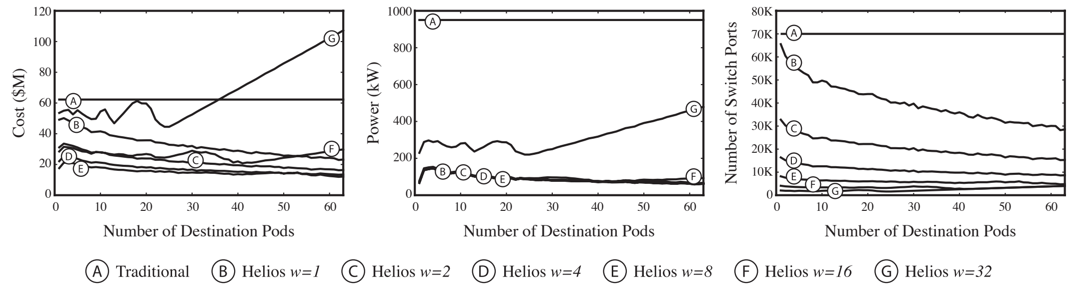


Figure 3.2: Simulation comparing traditional and Helios-like network.

destination pod pairs, and can be reallocated on millisecond timescales. As long as a given workload has at least 50% of its inter-pod traffic changing over multi second timescales, it should work well with this particular mix of packet and circuit switches.

3.3.2 Simulation Study at Scale

The goal of the simulation study is to understand the behavior of a large Helios deployment, more specifically, to understand the best allocation of packet switches and circuit switches and the best choice of w for superlinks. In order to answer these questions, a simplified model of Helios is created and a simulator and a traffic generator are implemented to analyze that model.

The simulator performs a static analysis, meaning that the circuit switch is set to a particular, fixed configuration for the duration of the simulation. In this way, the simulation results will predict a lower bound to the cost, power, and complexity savings of an actual Helios deployment that is able to change the circuit switch configuration at runtime to match the dynamically changing workload.

Simplified Model of Helios

The simplified topology model consists of $N = 64$ pods, each with $H = 1,024$ hosts. All core packet switches are combined into a single packet switch with a variable number of ports, P. The same is true for the circuit switch with C ports. Instead of dynamically shifting traffic between these two switches over time, traffic is statically assigned to one switch or the other. P and C expand as needed to completely encompass the communication demands of the hosts. There will be communication patterns where $C = 0$ is optimal (e.g., highly bursty, rapidly shifting traffic) and others where $P = 0$ is optimal (e.g., all hosts in one pod communicate only with hosts in exactly one other pod). The simplified model explores the space in between.

Table 3.1 shows the component costs. Three different types of transceivers are listed, showing the increase in cost for transceivers that support a greater number of wavelengths. The transceiver for $w = 32$ is an XFP DWDM transceiver, which costs more and consumes more power than an SFP+, since at the time this research was conducted (circa 2009), there were no SFP+ DWDM transceivers.

A bijective traffic model is used and defined as follows: the number of flows a host transmits always equals the number of flows the host receives. Analysis would be complicated by non-bijective patterns since certain bottlenecks would be at hosts rather than in the network. The traffic model consists of a variable called `num_dests`, the maximum number of destination pods a given host can communicate with throughout the course of the simulation. For simplicity and to maximize burstiness, it is assumed that when hosts change the destinations that they send traffic to, they do so simultaneously. `num_dests` is varied between 1 and $N - 1$ and the effect on system cost, power consumption, and number of required switch ports, is observed.

Simulation Methodology

For all 63 values of `num_dests`, multiple communication patterns are generated and simulated across the 6 topologies corresponding to different values of w. A traditional entirely-electrical switch is also considered, consisting of a sufficient number of 10 GigE ports to deliver non-blocking bandwidth among all pods.

For each communication pattern, a matrix of functions, $[f_{i,j}(t)]_{N \times N}$, is generated, where the value of the function is the instantaneous number of flows that pod i is sending to pod j. For each $f_{i,j}(t)$ over the simulation time interval $t \in [0, T]$, the minimum number of flows $m_{i,j} = min_t(f_{i,j}(t))$ is recorded and this number of ports is allocated to the core circuit switch. The rationale is as follows.

Consider a single large electrical packet switch. Each pod switch connects to the packet switch using H links. However, since it is known that pod i will always send at least $m_{i,j}$ flows to pod j, the total bisection bandwidth will not be reduced by disconnecting $m_{i,j}$ of pod i's physical links from the core packet switch and connecting them directly to pod j's transceivers. Since the circuit switch consumes less power and requires fewer transceivers and fibers, it is always advantageous to send this minimum traffic over the circuit switch (assuming equal cost per port). This process is repeated for every pair of pods for a total of $M = \sum_i^N \sum_j^N m_{i,j}$ ports, transceivers, and fibers. The remaining $(NH - M)$ links are allocated to the core packet switch.

Simulation Results

Figure 3.2 shows the simulation results for varying the number of destination pods for the different values of w to see the impact on cost, power consumption, and number of switch ports. Cost necessarily reflects a snapshot in time and sample values are included as a reasonable basis for comparison. The cost values are conservative.

These results are compared to a traditional architecture consisting entirely of electrical packet switches. Even without the ability to dynamically change the topology during run-time, the simulation results indicate a factor of three reduction in cost ($40M savings) and a factor of 9 reduction in power consumption (800 kW savings). A factor of 6.5 reduction in port count, or 55,000 fewer ports, is also achieved. Port count is a proxy for system complexity as fewer ports means less cables to interconnect, a smaller physical footprint for the switching infrastructure and less human management overhead.

Smaller values of `num_dests` result in a more expensive implementation because of the wider variation between the maximum number of flows and the minimum number of flows in $[f_{i,j}(t)]_{N \times N}$. When `num_dests` is large, these variations are minimized, reducing the number of links assigned to the more expensive core packet switch.

The parameter w influences cost both positively and negatively. Larger values of w reduce the number of fibers and core circuit switch ports, reducing cost. But larger values of w also lead to more *internal fragmentation*, which is unused capacity on a superlink resulting from insufficient demand. The effect of w on system cost is related to the number of flows between pods. In these experiments, it is found that a setting between $w = 4$ and $w = 8$ is optimal.

Surprisingly, if communication is randomly distributed over a varying number of destination pods, then even statically configuring a certain number of optical circuits can deliver bandwidth equal to a fully provisioned electrical switch, but with significantly less cost, power, and complexity. Much of this benefit stems from the ability to aggregate traffic at the granularity of an entire pod. Certainly this simple analysis does not consider adversarial communication patterns, such as when all hosts in a pod synchronously shift traffic from one pod to another at the granularity of individual packets.

3.4 Design and Implementation

This section describes the design and implementation of Helios, its major components, algorithms, and protocols.

3.4.1 Hardware

The prototype (Figures 3.3 and 3.4) consists of 24 servers, one Glimmerglass 64-port optical circuit switch, three Fulcrum Monaco 24-port 10 GigE packet switches, and one Dell 48-port GigE packet switch for control traffic and out-of-band provisioning (not shown). In the WDM configuration, there are 2 superlinks (dotted lines are $w = 2$) from each pod and the first and second transceivers of each superlink use the 1270 nm and 1290 nm CWDM wavelengths, respectively. The 24 hosts are HP ProLiant DL380 rackmount servers, each with two quad-core Intel Xeon E5520 Nehalem 2.26 GHz processors, 24 GB of DDR3-1066 RAM, and a Myricom dual-port 10 GigE NIC. They run Debian Linux with kernel version 2.6.32.8.

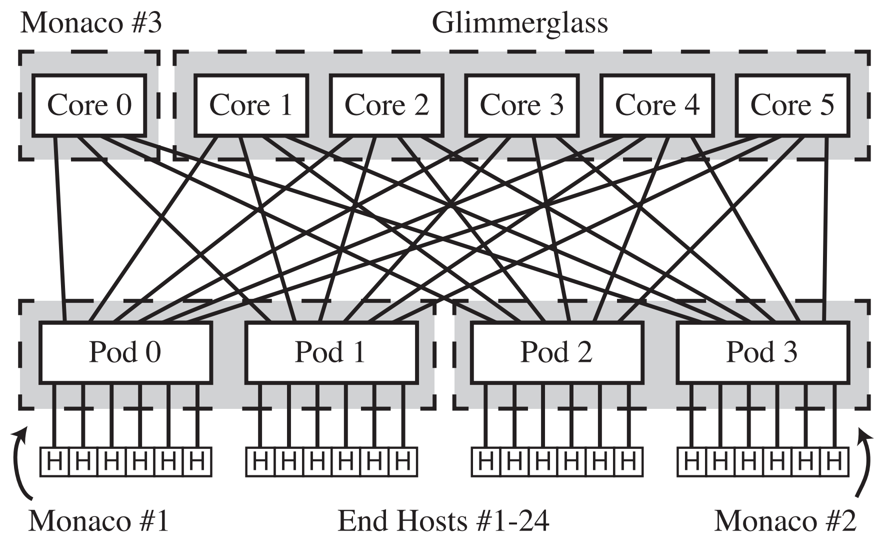


Figure 3.3: Helios prototype with five circuit switches.

The 64-port Glimmerglass crossbar optical circuit switch is partitioned into multiple (3 or 5) virtual 4-port circuit switches, which are reconfigured at runtime. The Monaco packet switch is a programmable Linux PowerPC-based platform that uses an FM4224 ASIC for the fast path. Two Monaco switches are partitioned into two 12-port virtual pod switches each, with 6 downlinks to hosts, 1 uplink to a core packet switch, and 5 uplinks to core circuit switches. Four ports of a third Monaco switch are used as the core packet switch. Virtualization allowed the construction of a more balanced prototype with less hardware.

3.4.2 Software

The Helios software consists of three cooperating components (Figure 3.5): a Topology Manager (TM), a Circuit Switch Manager (CSM), and a Pod Switch Manager (PSM). The TM is the heart of Helios, monitoring dynamically shifting communication patterns, estimating the inter-pod traffic demands, and computing new topologies (circuit configurations). The TM is a user-level process totalling 5,049 lines of Python with another 1,400 lines of C to implement the TCP fix-point algorithm (§3.4.3) and the Edmonds maximum weighted matching algorithm (§3.4.3). In the prototype, it runs on a single server, but in practice it can be par-

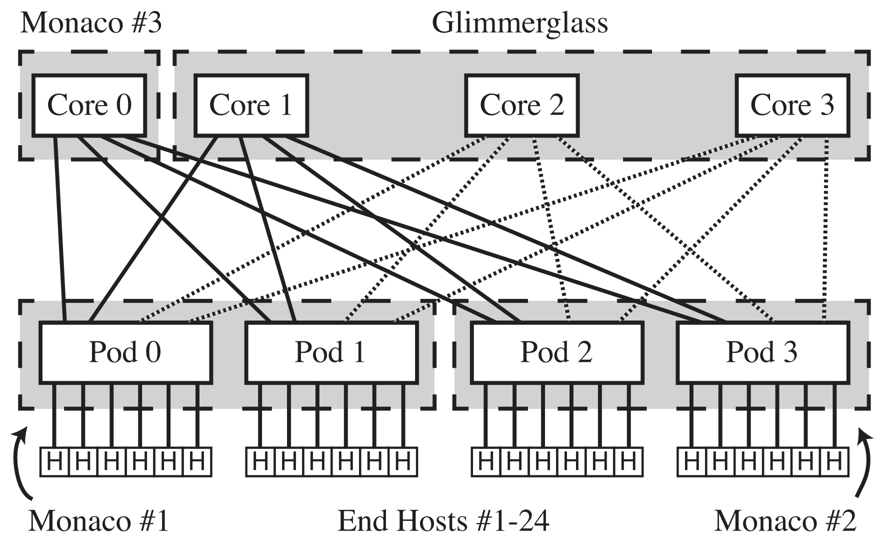


Figure 3.4: Helios prototype with three circuit switches and WDM.

titioned and replicated across multiple servers for scalability and fault tolerance. Section §3.4.3 discusses the details of the TM's control loop.

The CSM is the unmodified control software provided by Glimmerglass, which exports a TL1 command-line interface. In synchronous mode, the RPC does not return until after the operation has completed. In asynchronous mode, the RPC may return before the command has completed, and a second message may be generated after the operation finally completes. Synchronous mode is used for the evaluation. Neither mode has the ability to notify when the circuit configuration has begun to change or has finished changing.

The PSM runs on each pod switch, initializing the pod switch hardware, managing the flow table, and interfacing with the TM. Layer 2 forwarding is chosen rather than Layer 3 forwarding for the prototype, but the design is general to both approaches. The PSM supports multipath forwarding using Link Aggregation Groups (LAGs). Each destination pod is assigned its own LAG, which can contain zero or more physical circuit switch uplink ports. If a LAG contains zero ports, then the PSM assigns the forwarding table entries for that destination pod to the packet switch uplink ports. The number of ports in a LAG grows or shrinks with topology reconfigurations. The Monaco switch's default hashing over Layer 2, 3

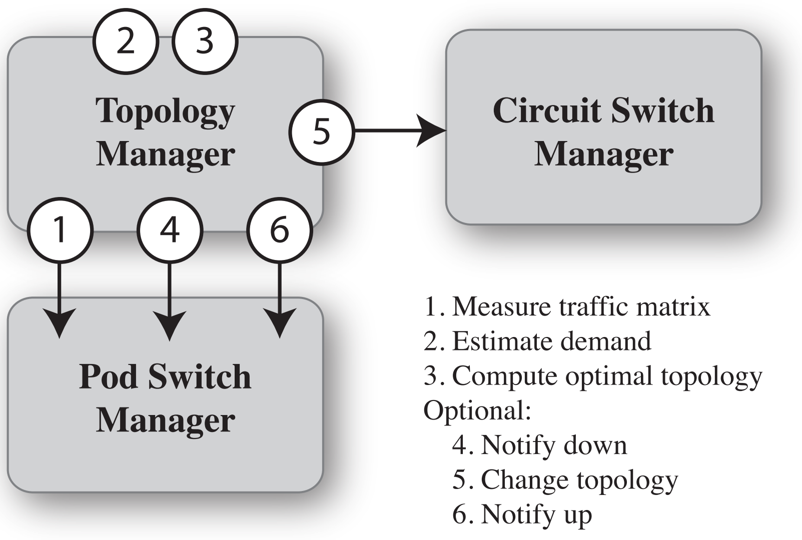


Figure 3.5: Helios control loop.

and 4 headers is used to distribute flows among ports in a LAG.

A limitation of the prototype is that a pod switch cannot split traffic over both the packet switch port as well as the circuit switch ports for traffic traveling to the same destination pod. This is due to a software restriction in the Monaco where a switch port cannot be a member of multiple LAGs.

Each PSM maintains flow counters for traffic originating from its pod and destined to a different pod. Most packet switches include TCAMs for matching on the appropriate packet headers, and SRAM to maintain the actual counters. The Monaco switch supports up to 16,384 flow counters.

The PSM is a user-level process, totalling 1,184 lines of C++. It uses the Apache Thrift RPC library to communicate with the TM. It uses Fulcrum's software library to communicate with the Monaco's FM4224 switch ASIC.

3.4.3 Control Loop

The Helios control loop is illustrated in Figure 3.5.

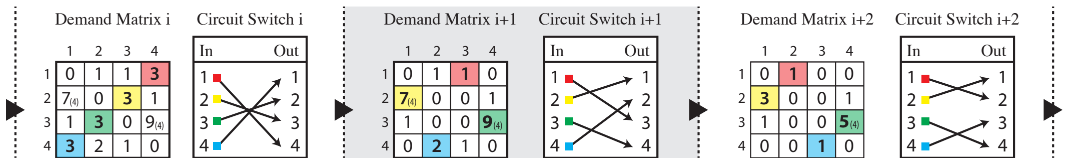


Figure 3.6: Example of Compute New Topology with 4 pods and $\boldsymbol{w} = \mathbf{4}$.

Measure Traffic Matrix

The TM issues an RPC to each PSM and retrieves a list of flow counters. It combines them into an octet-counter matrix and uses the previous octet-counter matrix to compute the flow-rate matrix. Each flow is then classified as either a mouse flow (<15 Mb/s) or an elephant flow and the mice are removed from the matrix. This static cutoff is chosen empirically for the prototype, and more sophisticated mice-elephant classification methods [48] can also be used. Intuitively, elephant flows would grow larger if they could, whereas mice flows probably would not; and most of the bandwidth in the network is consumed by elephants, so the mice flows can be ignored while allocating circuits.

Estimate Demand

The flow-rate matrix is a poor estimator for the true inter-pod traffic demand since it is heavily biased by bottlenecks in the current network topology. When used directly, the topology changed too infrequently, missing opportunities to achieve higher throughput. A better demand estimator is the max-min fair bandwidth allocation for TCP flows on an ideal non-oversubscribed packet switch. These computed fixpoint values are the same values that the TCP flows would oscillate around on such an ideal switch. An earlier algorithm [49] is used to compute these fixpoint values. The algorithm takes the flow-rate matrix as input and produces a modified flow-rate matrix as output, which is then reduced down to a pod-rate (demand) matrix by summing flow rates between pairs of pods.

Compute New Topology

The goal for computing a new topology is to maximize throughput. This problem is formulated as a series of max-weighted matching problems on bipartite graphs, illustrated with an example in Figure 3.6. Each individual matching problem corresponds to particular circuit switch, and the bipartite graph used as input to one problem is modified before being passed as input to the next problem. The first set of graph vertices (rows) represents source pods, the second set (columns) represents destination pods, and the directed edge weights represent the source-destination demand or the superlink capacity of the current circuit switch (in parentheses), whichever is smaller. The sum of the edge weights of a particular maximum-weighted matching represents the expected throughput of that circuit switch. Note that circuits are configured unidirectionally, meaning that having a circuit from pod i to pod j does not imply the presence of a circuit from pod j to pod i, as shown in Figure 3.6, stage $i + 1$.

Edmonds algorithm [50] is chosen to compute the max-weighted matching because it is optimal, runs in polynomial time ($O(n^3)$ for the bipartite graph), and because there are libraries readily available. Faster algorithms exist; however as shown in §3.5.6, the Edmonds algorithm is fast enough for the prototype.

Notify Down, Change Topology, Notify Up

If the new topology is different from the current topology, then the TM starts the reconfiguration process. First, the TM notifies the PSMs that certain circuits will soon be disconnected. The PSMs act on this message by removing the affected uplinks from their current LAGs, and migrating forwarding entries to the core packet switch uplinks if required. This step is necessary to prevent pod switches from forwarding packets into a black hole while the topology is being reconfigured. Second, the TM instructs the CSMs to actually change the topology. Third, the TM notifies the PSMs that the circuits have been re-established (but to different destination pods). The PSMs then add the affected uplinks to the correct LAGs and migrate forwarding entries to the LAGs if possible.

During both Notify Down and Notify Up, it is possible for some packets to

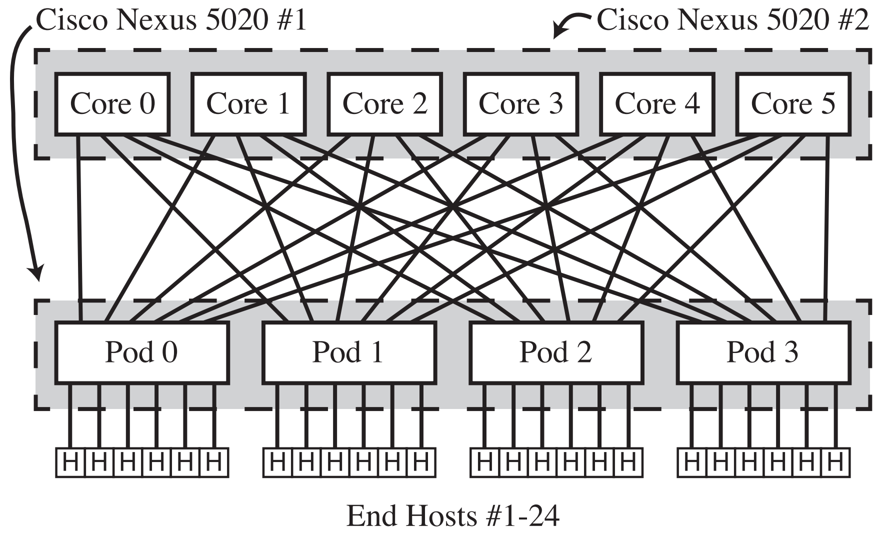


Figure 3.7: Traditional FatTree network of packet switches.

be delivered out of order at the receiver. However, the experiments do not show any reduction in throughput due to spurious TCP retransmissions.

3.5 Evaluation

The performance of Helios is evaluated against a fully-provisioned electrical packet-switched network arranged in a traditional FatTree topology (Figure 3.7). This traditional FatTree network serves as an upper bound for the prototype's performance, since it does not have to pay the overhead of circuit reconfiguration. One Cisco Nexus 5020 52-port 10 GigE switch is partitioned into four virtual 12-port pod switches, with one virtual switch per untagged VLAN. All six uplinks from each pod switch are combined into a single Link Aggregation Group (Ether-Channel), hashing over the Layer 2 and 3 headers by default. A second such switch is partitioned into six virtual 4-port core switches.

3.5.1 Communication Patterns

This section describes the communication patterns employed to evaluate Helios. Since no data center network benchmarks have yet been established, the first attempt at generating traffic is to run communication-intensive applications such as Hadoop's Terasort. Hadoop only achieved a peak aggregate throughput of 50 Gb/s for both configurations of Helios and the traditional FatTree network. Synthetic communication patterns that are not CPU bound or disk I/O bound are used to stress Helios beyond what is possible using Hadoop. Each pattern is parameterized by its *stability*, the lifetime in seconds of a TCP flow. After the stability period, a new TCP flow is created to a different destination. In all communication patterns, the same number of simultaneous flows from each host are used to minimize differences in hashing effects.

Pod-Level Stride (PStride). Each host in a source pod i sends 1 TCP flow to each host in a destination pod $j = (i + k) \bmod 4$ with k rotating from 1 to 3 after each stability period. All 6 hosts in a pod i communicate with all 6 hosts in pod j; each host sources and sinks 6 flows simultaneously. The goal of PStride is to stress the responsiveness of the TM's control loop, since after each stability period, the new flows can no longer utilize the previously established circuits.

Host-Level Stride (HStride). Each host i (numbered from 0 to 23) sends 6 TCP flows simultaneously to host $j = (i + 6 + k) \bmod 24$ with k rotating from 0 to 12 after each stability period. The goal of HStride is to gradually shift the communication demands from one pod to the next. HStride's pod-to-pod demand does not change as abruptly as PStride's and should not require as many circuits to be reconfigured at once. Intuitively, the aggregated pod-to-pod demand is more stable than individual flows.

Random. Each host sends 6 TCP flows simultaneously to one random destination host in a remote pod. After the stability period, the process repeats and different destination hosts are chosen. Multiple source hosts can choose the same destination host, causing a hotspot.

3.5.2 Debouncing and EDC

The initial prototype measurements show very poor performance, with throughput barely above that of the core packet switch used in isolation, when stability is less than 4 seconds. Figure 3.8 shows a PStride pattern with a stability

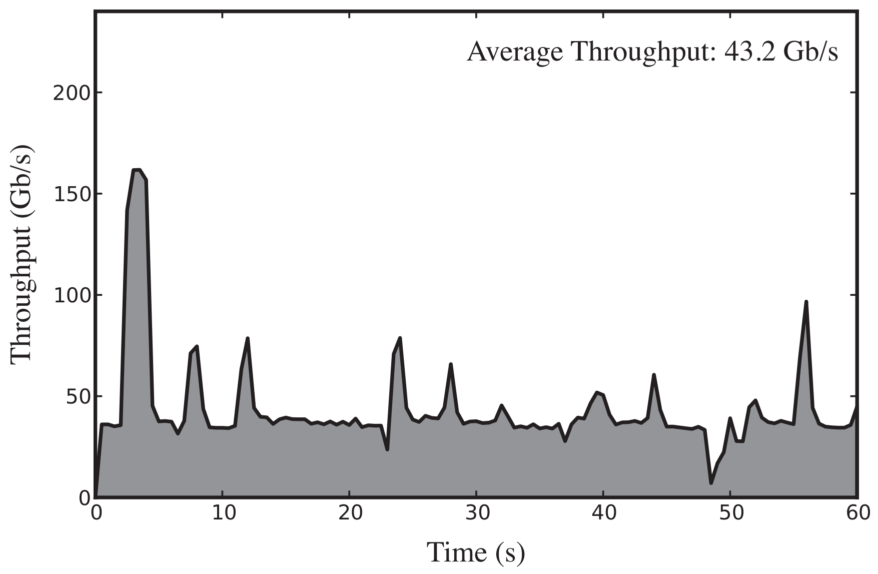


Figure 3.8: Throughput with baseline Fulcrum software.

of 4 seconds. Upon closer inspection, it is observed that there are durations of time where individual TCP flows are idle for more than 2 seconds.

This poor performance is caused by *Debouncing*. Debouncing is the technique of cleaning up a signal generated from a mechanical connection. When plugging a cable into a switch, the link goes up and down rapidly over a short time frame. Without debouncing, there is a potential for these rapid insertion and removal events to invoke multiple expensive operations, such as causing a routing protocol to issue multiple broadcasts. Debouncing works by detecting a signal and then waiting a fixed period of time before acting on the signal. The Monaco switch waits for two seconds before actually enabling a switch port after detecting the link. Since switch designers typically assume link insertion and removal are rare, they do not optimize for debouncing time. However, Helios rapidly disconnects and reconnects ports as the circuits are dynamically reconfigured to match shifting communication patterns.

After modifying Fulcrum's switch software to disable this functionality, the result is a dramatic performance improvement, as shown in Figure 3.9. The periods of circuit switch reconfiguration are now visible, but performance is still below expectations.

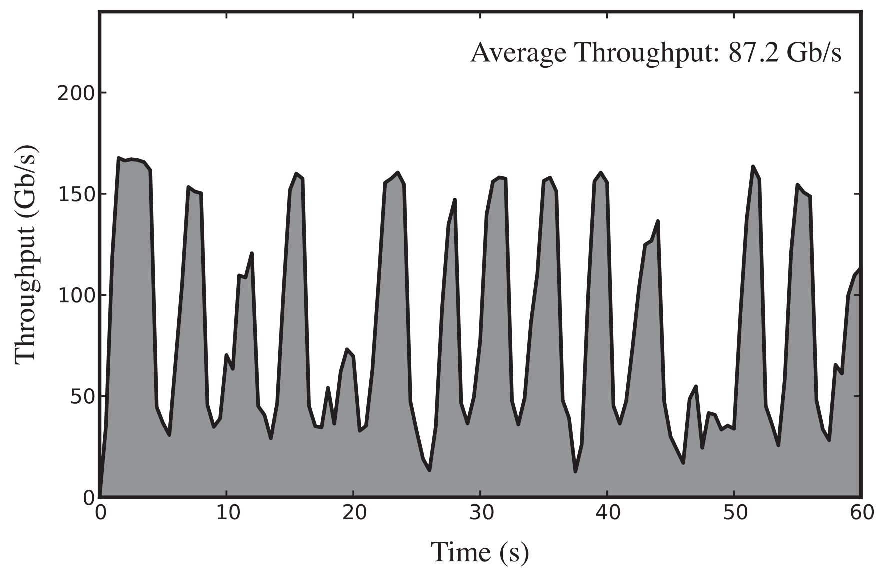


Figure 3.9: Throughput without "debouncing".

This remaining poor performance is caused by EDC. The Layer 1 PHY chip on the Monaco, a NetLogic AEL2005, takes 600 ms after a circuit switch reconfiguration before it becomes usable. Most of this time is spent by an *Electronic Dispersion Compensation (EDC)* algorithm, which removes the noise introduced by light travelling over a long strand of fiber. After disabling EDC, the system still worked correctly because the prototype uses short fiber cables and the transceivers have a relatively high power output. After disabling EDC, performance again improved (Figure 3.10). Now each change in flow destinations and corresponding circuit switch reconfiguration is clearly visible, but the temporary periods of lost capacity during circuit reconfigurations are much shorter. Even with EDC disabled, the PHY still takes 15 ms after first light before it becomes usable. In general, this work suggests opportunities for optimizing PHY algorithms such as EDC for the case where physical topologies may change rapidly.

One problem with FatTrees, including Helios, is that throughput is lost due to hotspots: poor hashing decisions over the multiple paths through the network. This can be seen in Figure 3.10 where the heights of the pillars change after every stability period. These types of hashing effects should be less pronounced with more flows in the network. Additionally, complementary systems such as

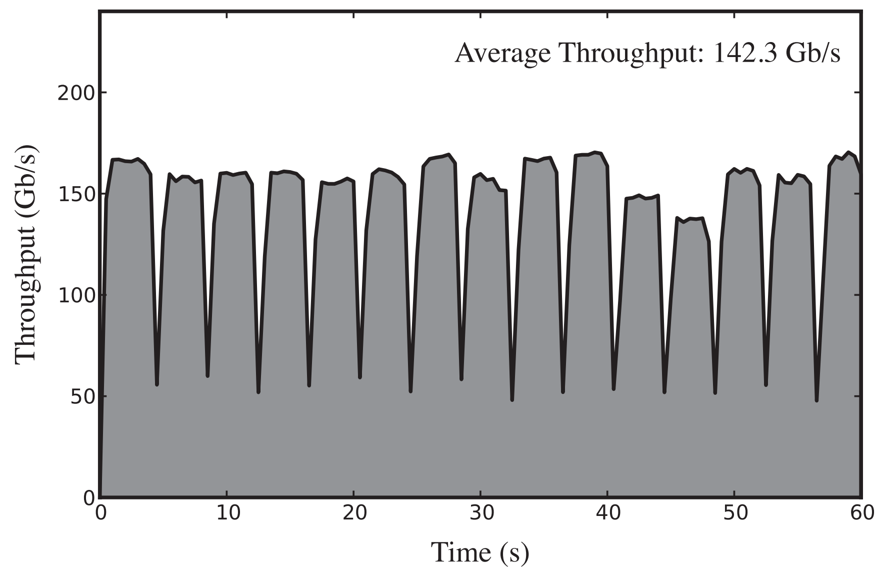


Figure 3.10: Throughput without "debouncing" and EDC.

Hedera [49] can be used to remap elephant flows to less congested paths.

3.5.3 Throughput versus Stability

Having considered some of the baseline issues that must be addressed in constructing a hybrid optical/electrical switching infrastructure, attention is now turned to Helios performance as a function of communication characteristics. The most important factor affecting Helios performance is inter-pod traffic matrix stability, which is how long a pair of pods sustains a given rate of communication. Communication patterns that shift too rapidly would require either too many under-utilized circuits or circuit-switching times not available from current technology.

Figure 3.11 shows the throughput delivered by Helios for the communication patterns in §3.5.1. Each bar represents the mean across 5 trials of the average throughput over 60 second experiments. The stability parameter is varied from 0.5 seconds up to 16 seconds. First, it is observed that higher stability leads to higher average throughput because more stable communication patterns require fewer circuit switch reconfigurations. Second, the throughput achieved by Helios

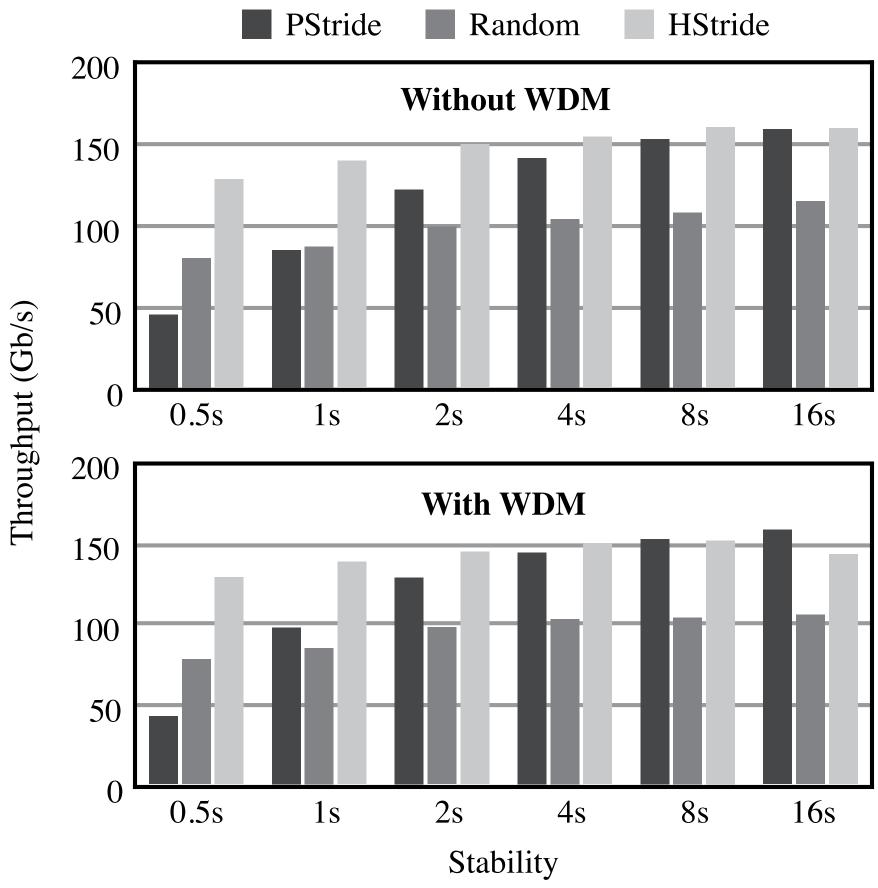


Figure 3.11: Throughput as a function of stability.

with WDM is comparable to the throughput achieved without WDM. This indicates that WDM may be a good way to reduce the cost, power consumption, and cabling complexity of a data center network without hurting performance for certain communication patterns. Third, for the same value of stability, throughput is generally better for HStride compared to PStride since the inter-pod traffic matrix is more stable. This suggests that even for low stability of individual flows, Helios can still perform well.

3.5.4 Unidirectional Circuits

Helios is designed to use either unidirectional circuits or bidirectional circuits. If port A in pod 1 connects to port B in pod 2, then bidirectional circuits would have port B in pod 2 connected back to port A in pod 1 through another circuit. Traditional approaches to establishing circuits employ bidirectional circuits because of the assumption of full duplex communication. However, unidirectional circuits have no such constraint and can better adapt to asymmetric traffic demands. For example, when transferring a large quantity of data from one pod to another, most of the traffic will flow in one direction.

This is illustrated in Figure 3.12, which shows throughput for a PStride communication pattern with 4 seconds of stability. Throughput with unidirectional circuits closely matches that of the traditional network during the stable periods, whereas bidirectional circuit scheduling underperforms when the pod-level traffic demands are not symmetric. During intervals with asymmetric communication patterns, bidirectional circuit scheduling is only half as efficient as unidirectional scheduling in this experiment. The higher throughput of the traditional network comes from two factors: first, the prototype does not split traffic between circuit switches and packet switches, and second, the traditional network does not suffer from temporary capacity reductions due to circuit switch reconfigurations. All other Helios results in this chapter use unidirectional circuits.

The 10G Ethernet standard includes a feature that complicates unidirectional circuits. If a Layer 1 receiver stops receiving a valid signal, it shuts off its paired Layer 1 transmitter. The assumption is that all links are bidirectional, so the loss of signal in one direction should disable both sides of the link in order to simplify fault management in software. In practice, no performance degradation from this feature is seen since circuits are reconfigured in parallel, and since all links are kept active. However, this feature increases the size of the failure domain: if one transceiver fails, all other transceivers in the same daisy-chained cycle are disabled. This problem can be solved by having the TM connect a failed transceiver in loopback, thus preventing it from joining a cycle with other transceivers.

3.5.5 How Responsive is Helios?

The length of the control loop is the most important factor in determining how quickly Helios can react to changing communication patterns. As discussed in §3.4.3, the control loop is composed of three mandatory stages (1-3) executed every

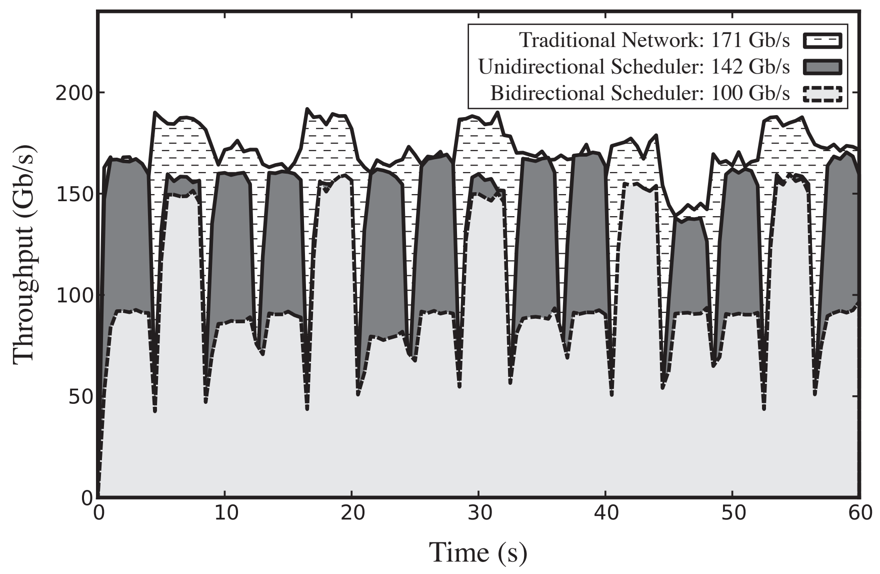


Figure 3.12: Unidirectional vs. bidirectional circuit scheduling.

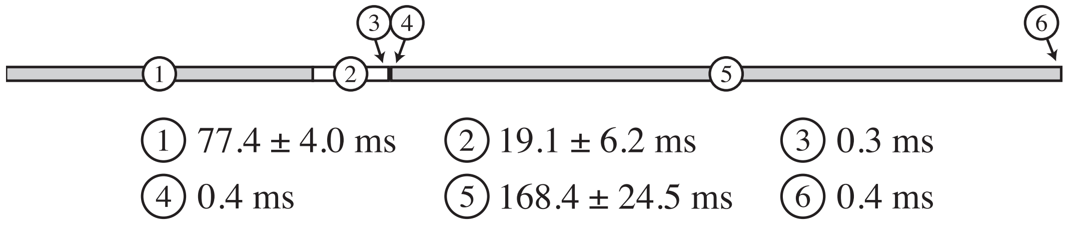


Figure 3.13: Profile of the Helios control loop.

cycle, and three optional stages (4-6) executed only when changing the topology (see Figure 3.5). Measurements of the prototype, summarized in Figure 3.13, show that the mandatory portion of a cycle takes approximately 96.5 ms, whereas the optional portion takes an additional 168.4 ms.The two primary bottlenecks are stage 1 (Measure Traffic Matrix) and stage 5 (Change Topology).

Analysis of Measure Traffic Matrix The Monaco switch uses a embedded PowerPC processor with a single core. Since each Monaco implements two pod switches, the 77.4 ms is actually two sequential RPCs of 38.7 ms each. The measurements showed that it took 5 μs to read an individual flow counter. Since each pod switch maintains 108 flow counters, this accounts for 0.54 ms of the 38.7 ms.

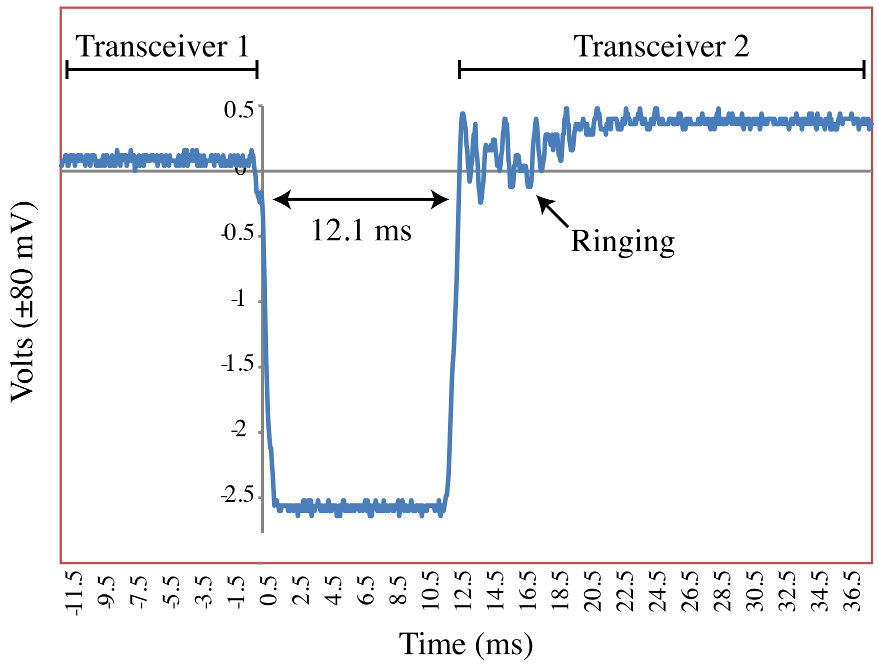


Figure 3.14: OCS characterization during reconfiguration.

Most of the 38.7 ms is actually being spent serializing the flow counter measurements into an RPC response message.

Analysis of Change Topology The Glimmerglass switch is advertised as having a 25 ms switching time, but host-level measurements showed a switching time of 168.4 ms. Figure 3.14 shows a time series measurement of the output power level from the Glimmerglass switch during a reconfiguration event. Lasers were connected to two different input ports. Initially, the OCS was configured to map input port 1 to the output port. At time $t = 0$, the OCS undergoes reconfiguration and maps input port 2 to the output port. At time $t = 12.1$, the OCS finished undergoing reconfiguration, and input port 2 is now mapped to the output port. The output port is connected to an oscilloscope via a photodetector. The 12 ms switching time is actually twice as fast as advertised. There is an additional period of mechanical ringing that occurs while the MEMS motor settles into place, but this can be taken advantage of in data center networks since the minimum signal level during this time is still high enough to send data.

Table 3.2: Circuit scheduler computation time.

	Scheduler Runtime (in ms)	
Number of pods	Bidirectional	Unidirectional
4	0.05	0.02
8	0.19	0.07
16	1.09	0.32
32	3.94	1.10
64	15.01	3.86
128	118.11	29.73

When the PHY overhead of 15 ms (with EDC disabled) is added to this measurement of 12 ms, the result should be a switching time of 27 ms. According to Glimmerglass, the synchronous RPC mechanism takes approximately 170 ms for internal processing before returning. The asynchronous RPC mechanism is measured to take only 30 ms, so a higher-performance prototype could use asynchronous RPCs.

3.5.6 How Scalable is Helios?

The algorithm for demand estimation is similar to earlier work [49]; its runtime has been measured to be less than 100 ms for large data centers, with additional speedups available through parallelization across multiple cores or servers. The overhead of circuit scheduling, another important component of the control loop, is also measured. Table 3.2 shows that the runtime of the circuit scheduling phase is moderate, less than 4 ms for unidirectional circuits and 64 pods.

By addressing overheads in the software as well as existing packet switch and optical circuit switch software, it should be possible to reduce the length of the control loop to well below 100 ms. However, some interesting optimizations would be required. For instance, reading a single hardware flow counter in the Fulcrum switch takes 5 us using its embedded processor. This means that reading the entire set of 16,384 counters would take more than 80 ms given the existing software/hardware structure. Developments such as OpenFlow [51] will provide incentives to integrate higher-speed commodity processors into switch hardware and provide efficient APIs for collecting statistics from switches [52].

3.6 Lessons Learned

An important lesson learned is that much of the networking equipment used to construct the Helios prototype is not engineered for the particular use cases. In case of the Fulcrum Monaco switch, the under-powered embedded processor took an unreasonably long time to transfer the flow counters to the Topology Manager (38.7 ms) compared to the time to simply read the flow counters locally (0.5 ms). If the Monaco's switch ASIC could perform flow classification locally [48] then perhaps this could cut down on the communication overhead. Additionally, if the debouncing feature is optional then software modifications would not be necessary.

In case of the NetLogic PHY, the overhead of EDC is likely acceptable if reconfigurations are only required once every 10 seconds or so. However, more bursty communication patterns, likely to be common in data center settings, would require a faster EDC algorithm.

In case of the Glimmerglass switch, the RPC overheads seem to be unnecessarily long and perhaps easy to fix. The 12.1 ms switching time looks to be fast enough for the periods of stability evaluated (500 ms and longer). But even switching times as low as 1 ms have been demonstrated [44], so additional optimizations might be possible.

Another lesson learned is that there is a subtle trade off between performance and fairness. For example, certain communication patterns can lead to poor but fair performance, or excellent but unfair performance when considering individual pod-level traffic demands. Striking the appropriate balance is a question of policy. The trade off is fundamentally caused by the fact that circuit switches have large switching times, and for certain communication patterns, fairness may have to be sacrificed for good performance or throughput by avoiding circuit reconfigurations.

Finally, Ethernet's fault management features assume bidirectional links, yet Helios achieves better performance using unidirectional circuits. It seems promising to extend Ethernet to the case where an out-of-band network could be used for endpoints to communicate their link status, thus removing the bidirectional assumption. This would make fault detection and recovery techniques in the Topology Manager less important.

3.7 Hardware Requirements

This section identifies hardware requirements for improving the performance of Helios, and by extension, other optically circuit-switched data center networks.

3.7.1 Performance Measurements

The Helios prototype uses a 64 $\times$ 64 Glimmerglass 3D MEMS OCS. OCS reconfiguration is divided into two consecutive time periods (Figure 3.15): command processing ($T_1 = 5$ ms) and mirror reconfiguration ($T_2 = 12$ ms). In addition, receiver electronics initialization (T_3) takes 15 ms and begins after OCS mirror reconfiguration. The dips in Figure 3.10 are caused by the $T_1+T_2+T_3 = 32$ ms period when network traffic stops flowing over the existing circuits and is forced over the single core EPS; throughput recovers when the new circuits are established. If T_1, T_2, and T_3 can be reduced, then the performance of a hybrid EPS/OCS network can approach that of a pure EPS network, even for bursty traffic.

During T_1, the OCS receives a reconfiguration command message over a 1G Ethernet TCP/IP port and processes this message in software on an embedded CPU running Linux. Although the existing circuits remain established during this time, T_1 reduces network throughput by delaying the time between when a change in the traffic pattern is detected and when the mirrors are actually reconfigured. T_1 itself can be reduced by using a faster CPU and by streamlining the software. By simplifying the protocol to be UDP/IP-based, it should be possible to reduce T_1 to approximately 10 μs with an FPGA implementation.

During T_2, an embedded microcontroller moves a subset of the input and output mirrors to establish new circuits. Minimizing T_2 is critical because no

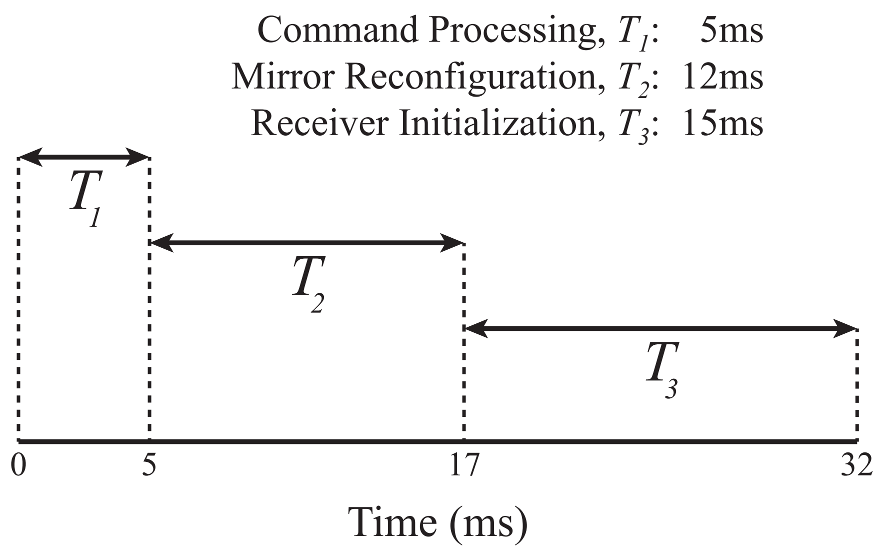


Figure 3.15: OCS measurements.

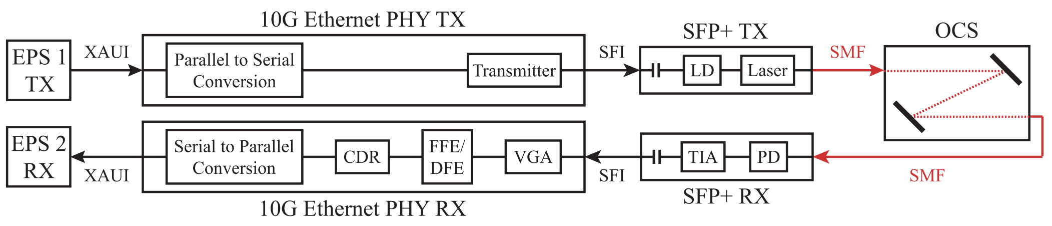


Figure 3.16: End-to-end data path.

network traffic can flow over the affected circuits during this time. T_2 can be reduced further by using smaller mirrors with less mass, a smaller turning range, and fewer turning steps, but this will increase optical loss and reduce maximum port count [53]. For example, in a 100-port 3D MEMS OCS with a 600 μm diameter mirror size, Yamamoto, et al. [54] achieved a reconfiguration time of 1.5 ms, or 3 ms when the switching is fully stable. Texas Instruments DLP technology can reconfigure in 15 μs, but uses mirrors that are approximately 5 μm in diameter and can only move between two fixed positions, which could lead to unacceptable optical loss and insufficient port counts if used as a 3D MEMS OCS [55].

Figure 3.16 shows the data path from the transmit port of EPS pod switch 1, through an OCS, to the receive port of EPS pod switch 2. The electronics on the receive path experience loss of signal during an OCS reconfiguration.

During T_3, the circuits are established, but the receiver electronics are still being initialized after a loss of signal during T_2. Minimizing T_3 is also critical because no network traffic can flow over the affected circuits until initialization is complete. T_3 is actually the maximum initialization time among all circuits in the receive direction of the data path (Figure 3.16), specifically the transimpedance amplifier (TIA), the variable gain amplifier (VGA), the feed-forward equalizer and decision-feedback equalizer (FFE/DFE), and the clock/data recovery unit (CDR).

The NetLogic AEL2005 PHY used in the Helios prototype has FFE/DFE and VGA initialization times of 600 ms and 15 ms, respectively. These two components are responsible for electronic dispersion compensation (EDC), which is not needed for Helios because the optical transceivers use limiting amplifiers and only short runs of single-mode fiber (SMF). FFE/DFE was disabled for the measurements in Figure 3.15 but VGA was left enabled. CDR units initialize quickly, with a typical locking time of 50 ns. After disabling the VGA and using smaller DC blocking caps, the primary bottleneck would be the continuous TIA, with typical initialization times of 2 μs to 80 μs [56]. This could be reduced by using a burst-mode TIA such as designed for 10G EPON. A burst-mode TIA has been demonstrated with an initialization time of less than 200 ns [57].

3.7.2 Analysis

For an adversarial traffic pattern, throughput drops to zero during the entire $T_1 + T_2 + T_3 = 32$ ms period. Equation 3.1 gives the throughput ratio as a function of T_1, T_2, T_3, and S (period of traffic stability). Throughput is zero when $S \leq T_1 + T_2 + T_3$. The additional loss in throughput caused by the performance of the real-time circuit scheduler is outside the scope of this paper. Figure 3.17 shows plots of (3.1).

$$\frac{S - T_1 - T_2 - T_3}{S} \tag{3.1}$$

From Figure 3.17, we can see that the current Helios prototype performs nearly as well as a pure EPS network when S is at least 500 ms, as the plot with the yellow triangle shows. The plot with the green square shows the performance for

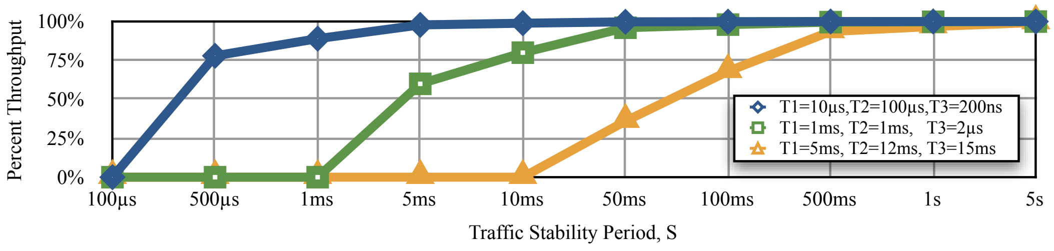


Figure 3.17: Throughput compared to a pure EPS.

the case when the command processing time and mirror reconfiguration time are both reduced to 1 ms each, and when EDC is fully disabled in the PHY. The plot with the blue diamond might be close to the maximum achievable performance, with a command processing period and mirror reconfiguration period of 10 μs and 100 μs, respectively, and when the receive path is optimized for burst-mode operation. In this example, a Helios network would achieve 75% of the throughput of an EPS network if the pod-to-pod traffic demand is stable for 500 μs, i.e. if a pod transmits exactly 5 megabytes at a time to another pod. Close to 100% throughput can be achieved when transferring 50 megabytes of data at a time.

3.8 Related work

Combining circuit switching with packet switching to take advantage of the relative merits of each is not new. For example, the pioneering work on ATM Under IP [58] showed how to amortize the cost of virtual circuit establishment over the long flows that would most benefit from it. Relative to this work, Helios considers the special characteristics of data center deployments, optical interconnects, and flow aggregation through WDM.

A hybrid optical/electrical interconnect was previously proposed for high performance computing [59]. The underlying motivation is similar in that certain HPC applications establish regular communication with remote partners and would benefit from circuit switching. Relative to Helios, this effort considers circuit establishment on a host to host basis rather than taking advantage of the available aggregation available in larger deployments at the granularity of pods. Further, the decision to switch between packet and circuit is made in the hosts, requiring either operating system modification or compiler support. Helios transparently switches communication at the core layer in the network based on observations of dynamically changing communication patterns.

Schares et al. [60] propose a hybrid electrical/optical network for stream processing. A working prototype using a 3D MEMS OCS was demonstrated. The scheduler was implemented as an extension of SODA (Scheduling Optimizer for Distributed Applications). VLANs are used for routing, similar to SPAIN [61].

Wang et al. [62, 63] also propose a hybrid electrical/optical switch for data center deployments. The hosts are considered to be part of the routing/switching infrastructure. Hosts perform output queuing of individual flows, waiting to gather enough data to leverage an available circuit to a remote host. This approach requires modifications to the hosts and can significantly increase latency, an important concern for many data center applications. Traffic demands are estimated by observing queue lengths across all hosts, whereas Helios leverages existing flow counters in commodity switches.

Halperin et al. [64, 65] propose using high-frequency, short-distance wireless connectivity between data center racks to deliver the necessary bandwidth among the nodes that dynamically require it. Optical and wireless interconnects provide different trade offs. For example, wired optical interconnects can deliver fundamentally more bandwidth at lower power consumption, especially when leveraging WDM. Wireless may be easier to deploy since it does not require any wiring, though management and predictability remain open questions.

MDCube [66] also considers interconnects for modular data centers. Each container uses BCube [35] internally and MDCube interconnects the containers using electrical packet switches. Relative to this effort, Helios explores the cost, complexity, and energy benefits of a hybrid electrical optical interconnect and hence the work can be considered as potentially delivering additional benefits to MDCube.

Terabit Burst Switching (TBS) [67] also employs WDM to increase link capacity. The primary difference is that TBS is based on the concept of a burst, i.e., a relatively large packet. TBS couples the control plane with the data plane by having each Burst Switching Element decode a special packet that includes burst length. This packet may involve aggregation over multiple flows all headed to the same destination by performing appropriate buffering at the ingress points in the network. In contrast, Helios targets data center deployments with small RTTs and latency-sensitive applications. A centralized Topology Manager implements the control plane and unilaterally reconfigures the underlying topology based on estimated traffic demands. Decoupling the control plane from the data plane allows the network to run with unmodified hardware and software.

Keslassy et al. [68] presents the design a multi-terabit, multi-chassis Internet router using optical switching and WDM to reduce cabling complexity and required port count. Relative to this effort, Helios dynamically reconfigures the optical circuit elements to match dynamically changing communication patterns whereas Keslassy et al. use multiple fixed configurations to perform a variant of Valiant Load Balancing to distribute traffic among multiple parallel optical crossbars.

PIM[69] and iSLIP[70] schedule packets or cells across a crossbar switch fabric over very short timescales. The crossbar topology is fixed and the goal is to maximize throughput by finding a series of maximal matchings from input ports to output ports. Likewise, Helios shares the same goal of maximizing throughput, except the technique is to modify the underlying network topology to provide additional physical links between pod pairs with higher traffic demands. Due to the longer circuit switching times, the matching must necessarily occur over longer timescales and with coarser-grained information about traffic demands.

Acknowledgements

Chapter 3, in part, reprints material as it appears in the paper titled "Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers", published in the Proceedings of the 2010 ACM SIGCOMM Conference, by Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat [3].

Chapter 3, in part, reprints material as it appears in the paper titled “Hardware Requirements for Optical Circuit Switched Data Center Networks”, published in the Proceedings of the 2011 Optical Fiber Communication Conference, by Nathan Farrington, Yeshaiahu Fainman, Hong Liu, George Papen, and Amin Vahdat [4].

Chapter 4

Microsecond-scale Optical Circuit Switching

This chapter explores two research questions. First, can faster OCS switches improve the performance of hybrid EPS/OCS data center networks? Second, what is the best way to integrate these faster OCS switches into the data center network? In an effort to answer these two questions, this chapter presents the design, implementation, and evaluation of Mordia, a 24×24-port OCS prototype. Mordia is three orders of magnitude faster than current commercial OCS switches, with a programming time of 68.5 μs, a switching time of 2.8 μs, and a receiver electronics initialization time of 8.7 μs. The Mordia architecture can scale to 704 ports with current commodity optical components. Host-level TCP and UDP measurements are presented using a novel implementation of TDMA in software.

Previous work in hybrid EPS/OCS data center networks, including Helios, has relied on a technique called hotspot scheduling, in which the traffic demand matrix is estimated, hotspots identified, and circuits established to automatically offload traffic from the packet-switched network. While this hybrid approach does reduce CAPEX and OPEX, it still relies on having a well-provisioned packet-switched network to carry the remaining traffic. This chapter describes a generalization of hotspot scheduling, called traffic matrix scheduling, where most or even all bulk traffic is routed over circuits. Traffic matrix scheduling rapidly time-shares circuits across many destinations at microsecond time scales. The traffic matrix scheduling algorithm can route arbitrary traffic patterns and runs in polynomial time.

4.1 Introduction

Data center networks are crucial to the scalability and performance of data center applications, yet are often underprovisioned due to their high CAPEX and OPEX [11]. Recent work has suggested the construction of hybrid networks consisting of both traditional electronic packet switches (EPS) as well as optical circuit switches (OCS) [3, 62, 71, 72]. The idea is that stable traffic is routed through the OCS switches and the remaining bursty traffic is routed through the EPS switches. Since a single OCS can replace multiple EPS, the overall CAPEX and OPEX can be reduced, while still providing the performance of a fully-provisioned network.

Unfortunately, data center operators have not yet chosen to deploy OCS switches inside the data center, with the primary reason being the relatively slow OCS switching times, typically 12 ms or longer (Section 3.7.1). Experience with Helios (chapter 3) suggests that current commercial OCS are best suited for workloads such as rack-to-rack backup or virtual machine migration, which both exhibit long periods of stability. However, there is a significant amount of all-to-all traffic in the data center, seen in applications such as MapReduce and on-line webpage rendering using distributed memory caches.

This chapter describes the design and implementation of an OCS architecture and prototype, called Mordia, for use in data center networks, with the specific goal of supporting a wider range of communication patterns, including all-to-all. Section 4.2 explains the technology behind fast OCS switches. Section 4.3 describes the Mordia OCS prototype and presents the data plane (section 4.3.1), control plane (section 4.3.2), implementation (section 4.3.3), scalability (section 4.3.4), and analysis (section 4.3.5). Section 4.4 presents an evaluation of both the prototype itself, as well as host-to-host traffic transiting the prototype switch. Section 4.5 describes the Traffic Matrix Scheduling (TMS) algorithm.

Table 4.1: Comparison of 2D and 3D MEMS-based OCS.

Metric	**2D MEMS**	**3D MEMS**
Maximum port count	8	320
Typical insertion loss	12 dB	3 dB
Switching time	2.8 μs	12 ms

4.2 Technology

A survey was conducted to study the different technologies known for switching light. Each technology has its own advantages and disadvantages in terms of cost, power consumption, insertion loss, and switching speed. A technology such as the semiconductor optical amplifier (SOA) has a switching speed of a few nanoseconds, but high cost, high power consumption, high insertion loss, and high noise [46], meaning that the resulting OCS would be unattractive in a data center environment. The survey concluded that MEMS technology offers the best overall trade offs for data center environments. There are two classes of MEMS-based OCS: 2D and 3D [46]. For reference, the OCS used in Helios (chapter 3) is 3D MEMS technology. Table 4.1 summarizes the differences between these two classes. For both classes, the energy usage is comparable. Cost is not compared since it is largely a function of commercial volume.

3D MEMS technology is superior to 2D MEMS for the metrics of port count and insertion loss. But the lower reconfiguration time of 2.8 μs makes 2D MEMS more attractive. The challenge then becomes how to construct a large port count OCS out of multiple 2D MEMS chips. A naive approach would cascade multiple 2D MEMS OCS into a multistage interconnection network [21]. Unfortunately, each stage adds insertion loss, so this technique can only be repeated a small number of times.

To overcome the problem of insertion loss, the broadcast-and-select [46] architecture is chosen for the Mordia OCS prototype. Each OCS input port uses a different wavelength channel. First, all of the channels are combined together using wavelength division multiplexing (WDM). Next, this aggregate of channels is broadcast to all OCS output ports. Finally, each output port selects a particular wavelength channel from the aggregate. This selection is accomplished

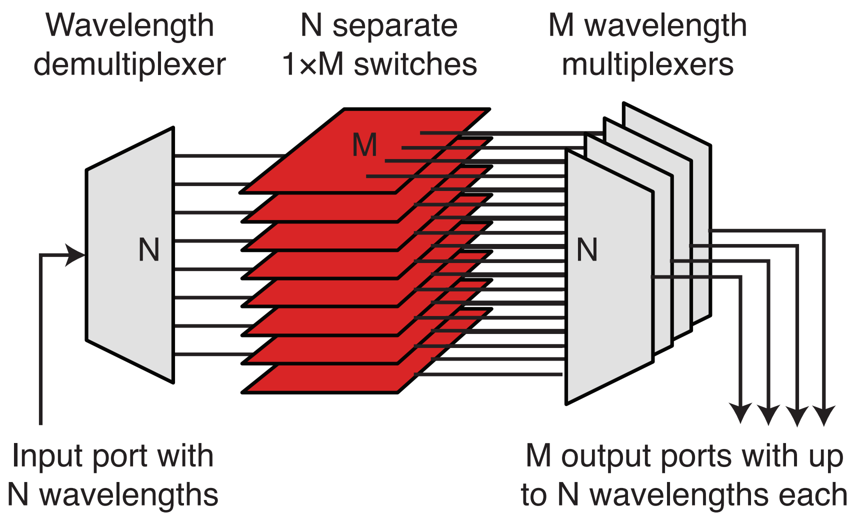

Wavelength demultiplexer
N separate 1×M switches
M wavelength multiplexers
M
N
N
Input port with N wavelengths
M output ports with up to N wavelengths each

Figure 4.1: A 1×M wavelength-selective switch (adapted from [73]).

using a wavelength-selective switch (WSS), which is shown in Figure 4.1. N input wavelength channels are demultiplexed across N separate $1 \times M$ space switches to perform per-wavelength switching. In Mordia, only one wavelength channel is sent to a particular output port at a time. For Mordia, the WSS uses a 2D MEMS-based space switch.[1]

4.3 Mordia OCS Prototype

4.3.1 Data Plane

The data plane is responsible for conveying packets from input ports to output ports. It is physically constructed as a unidirectional ring of N individual wavelength channels, carried in a single optical fiber. Each input port is assigned its own specific wavelength that is not used by any other input port. An output port can tune to receive any of the wavelengths in the ring, and thus deliver packets from any of the input ports. Consequently, this architecture supports circuit unicast, circuit multicast, circuit broadcast, and also circuit loopback[2], in

[1]The technique described uses fixed transmitters and tunable receivers. An alternate implementation could use tunable transmitters and fixed receivers.

[2]Loopback is useful for testing.

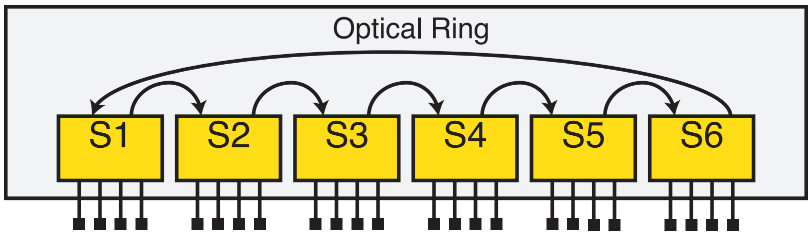


Figure 4.2: Unidirectional optical ring of stations.

which traffic from each port transits the entire ring before returning back to the source.

Wavelengths are added or dropped from the ring at *stations*. Figure 4.2 shows that the Mordia OCS prototype is constructed as an optical ring of six stations with four ports per station for a total of 24 ports. A single ring scales to a total of 11 stations and 44 ports when using the ITU-T G.694.1 DWDM 100 GHz frequency grid, or 22 stations and 88 ports using the 50 GHz grid. Section 4.3.4 describes how to stack multiple rings within the same OCS to increase the total port count, and how to determine the maximum number of stations that can be attached to a single ring. A station contains four input/output port pairs and is shown in Figure 4.3. A station contains all of the optical components necessary to combine, split, amplify, attenuate, and switch wavelength channels.

At each station, four wavelengths are added to the ring from the input ports. These four wavelengths are then attenuated with variable optical attenuators (VOA) so that their power level matches the power levels of the other wavelengths currently in the ring. Next these four wavelengths are multiplexed together with the other wavelengths in the ring. Next the aggregate wavelengths go to a 10/90 splitter. 10% of the power goes to a debug port that can be monitored using a optical spectrum analyzer (OSA). The remaining 90% of the power goes to an EDFA-based fiber amplifier, which outputs a constant 23 dBm of power. It is important to match the power levels of all wavelength channels prior to amplification.

The output of the amplifier from one station is connected to the input of the next station in the ring. The aggregate wavelengths first pass through another

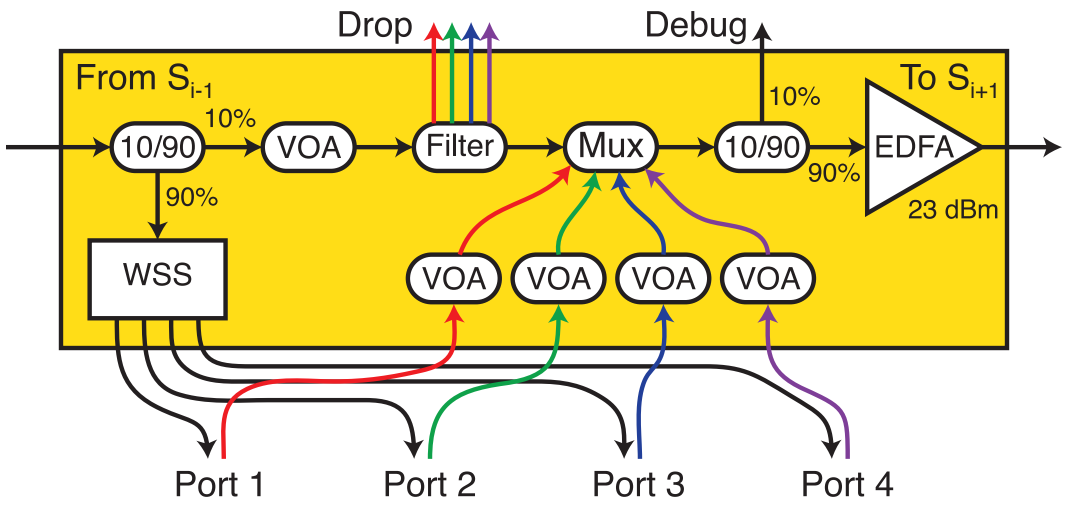


Figure 4.3: A station in the optical ring.

10/90 splitter. In this case, 10% of the power continues on through the station to eventually be combined with the four input wavelengths. First a VOA attenuates the aggregate wavelengths. Second the four wavelengths that are entering the station are removed from the aggregate. This is necessary to prevent a collision. Finally the aggregate is combined with the four input wavelengths. The remaining 90% of the power is sent to a 1×4-port WSS, which selects the wavelength channels sent to the output ports.

Optical amplifiers are used to overcome the 12 dB insertion loss of the WSS. However, the use of optical amplifiers presents its own challenges. One challenge is that amplification adds noise, which in turn increases the bit error rate (BER) of the channel. Although no increase in the BER of the prototype is measured when using a series of six amplifiers, it is possible that adding more amplifiers could increase the BER to noticeable levels. Another challenge is that placing amplifiers in a ring could result in lasing conditions if the round-trip gain exceeds the round-trip loss. This problem is solved by placing a VOA in each station to attenuate the aggregate wavelengths before amplification.

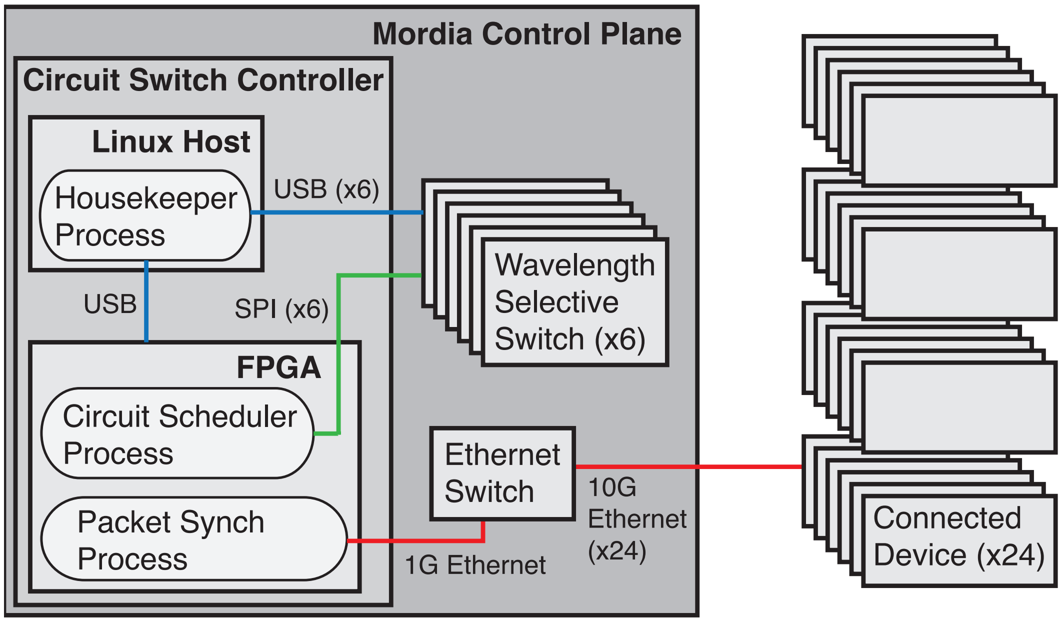


Figure 4.4: Control plane of the 24-port Mordia OCS prototype.

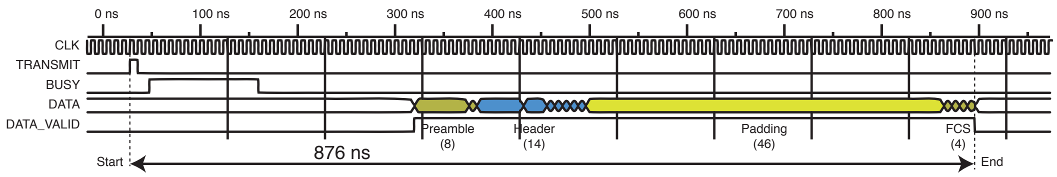


Figure 4.5: Sync packet propagation delay inside the FPGA.

4.3.2 Control Plane

The control plane interacts with the data plane in two essential ways. First, it maintains a schedule: a plan of *when* to reprogram the WSS modules and *what* their input-output port mappings should be. Second, it allows devices connected to the Mordia OCS to become synchronized and learn the schedule. This is necessary so that the connected devices will know *when* to send data and *where* that data will be sent to.

Hardware

The control plane of the 24-port Mordia OCS prototype is shown in Figure 4.4. The control plane consists of the Circuit Switch Controller, six WSS modules, and a 10G Ethernet switch. The Circuit Switch Controller is comprised of a Linux host for non-real-time processes and an FPGA for real-time processes. The Linux host is connected to the FPGA over USB, with the host as the master. The Linux host is also connected to the six WSS modules over six USB cables. The FPGA is connected to the six WSS modules via six 10 MHz SPI ribbon cables. The FPGA is also connected to an Ethernet switch, which is then connected to all 24 connected devices.

Software

The three primary control plane processes are the Housekeeper Process, the Circuit Scheduler Process, and the Packet Synchronization Process.

The Housekeeper Process interfaces with all of the different hardware modules inside of the Mordia prototype over their respective management interfaces. Its primary responsibilities are to (a) configure and supervise the other hardware modules, (b) read the current status of the hardware and publish the results to a website, (c) detect potentially unsafe situations such as overheating or improper laser power levels and then take appropriate action such as shutting down the hardware and notifying the operator.

The Circuit Scheduler Process is responsible for for programming the six WSS modules with the correct input-output port mappings at the right time. Since each WSS module has four output ports, the process sends 4 bytes of data with each port/channel assignment encoded as a single byte. These 32-bits of data are transmitted over a SPI interface with the FPGA as the master and the WSS module as the slave. The WSS module supports a maximum SPI transfer rate of 10 MHz. The prototype transmits at a slower rate of 5.33 MHz due to the length of the connecting ribbon cables. This corresponds to a programming time of 6 μs. This programming happens while the optical circuits are stable and can carry traffic; the reconfiguration does not occur until explicitly triggered.

The Packet Synchronization Process is responsible for transmitting synchronization packets at very precise times in order to notify connected devices of the current and future input-output OCS port mappings. There are two types of synchronization packets: setup and teardown. A teardown packet signals that the current optical circuits are about to be broken. A setup packet signals that new optical circuits have just been established. Each synchronization packet also includes relevant input-output port mapping information.

Synchronization packets are implemented as a custom Layer 2 Ethernet protocol.[3] The FPGA transmits the synchronization packets using 1G Ethernet. An Ethernet switch then broadcasts the synchronization packets in parallel to all 24 connected devices using 10G Ethernet.

The synchronization packets do not arrive at the connected devices instantaneously. Figure 4.5 from the Xilinx ISIM Verilog cycle-accurate hardware simulator shows that the packet generation process takes 876 ns from start to finish. Note that this delay could be reduced to 576 ns with more complicated logic to write directly to the Ethernet MAC. Additionally this delay could be reduced to 57.6 ns by using 10G directly instead of 1G. In addition to the propagation delay of 876 ns, the Ethernet switch adds a minimum of 67.2 ns of additional latency because of store-and-forward when converting from 1G to 10G. Finally the synchronization packet arrives at a connected device and undergoes another 67.2 ns delay for deserialization. The entire end-to-end propagation delay is close to 1 μs.

[3]It may be possible to use the IEEE 1588 Precision Time Protocol.

Table 4.2: Bill of materials

Component	Make/Model
Ethernet switch	Cisco Nexus 5596UP
fiber amplifier	OEQuest EDFA-GB-23
FPGA	Digilent Atlys (Xilinx Spartan-6)
Linux host	HP DL380 G7
optical spectrum analyzer (OSA)	Anritsu CMA-5000a
SFP+ modules	Eoptolink EOLP-1696-14-XX
VOA	Kotura K100-3410
WDM mux/demux	Auxora LGX1-5-7DWDM/BPS,100G
WSS	Nistica Full Fledge 100

4.3.3 Implementation

Figures 4.6 and 4.7 show the rack containing the Mordia prototype. At the top is a Linux host running the Housekeeper process. Underneath are six fiber amplifiers. Each amplifier belongs to a separate station. Underneath the amplifiers is the OSA used to observe the individual wavelength channels if necessary. Underneath the OSA is a slow, mechanical OCS used to select which debug port from each of the six stations is connected to the OSA. The bottom of the rack contains four sliding shelves. The top shelf contains the FPGA from the Circuit Switch Controller, the six VOAs that are inside the ring, and some power supplies. The other three shelves are identical and each contain the optical components for two stations. Table 4.2 gives the make and model of each component.

The Helios prototype uses 10G SFP+ optical transceivers that comply with the 10GBASE-LR specification for long-range (10 km) transport. But instead of using the standard wavelength of 1310 nm, four different wavelengths on the CWDM grid are used: 1270 nm, 1290 nm, 1310 nm, and 1330 nm. The Mordia prototype also uses 10G SFP+ optical transceivers[4] following the 10GBASE-LR specification, but uses the following wavelengths from the ITU-T G.694.1 DWDM 100 GHz frequency grid [74]: 15-18, 23-26, 31-34, 39-42, 47-50, 55-58. These DWDM SFP+ modules have a specified power budget of 14 dB. The output power is measured to be between 1.5 dBm and 3.5 dBm, and the receive sensitivity is measured to be approximately -17 dBm. This 14 dB power budget combined with

[4]http://www.sfp-xfp.com/products/dwdm-sfp-plus.html

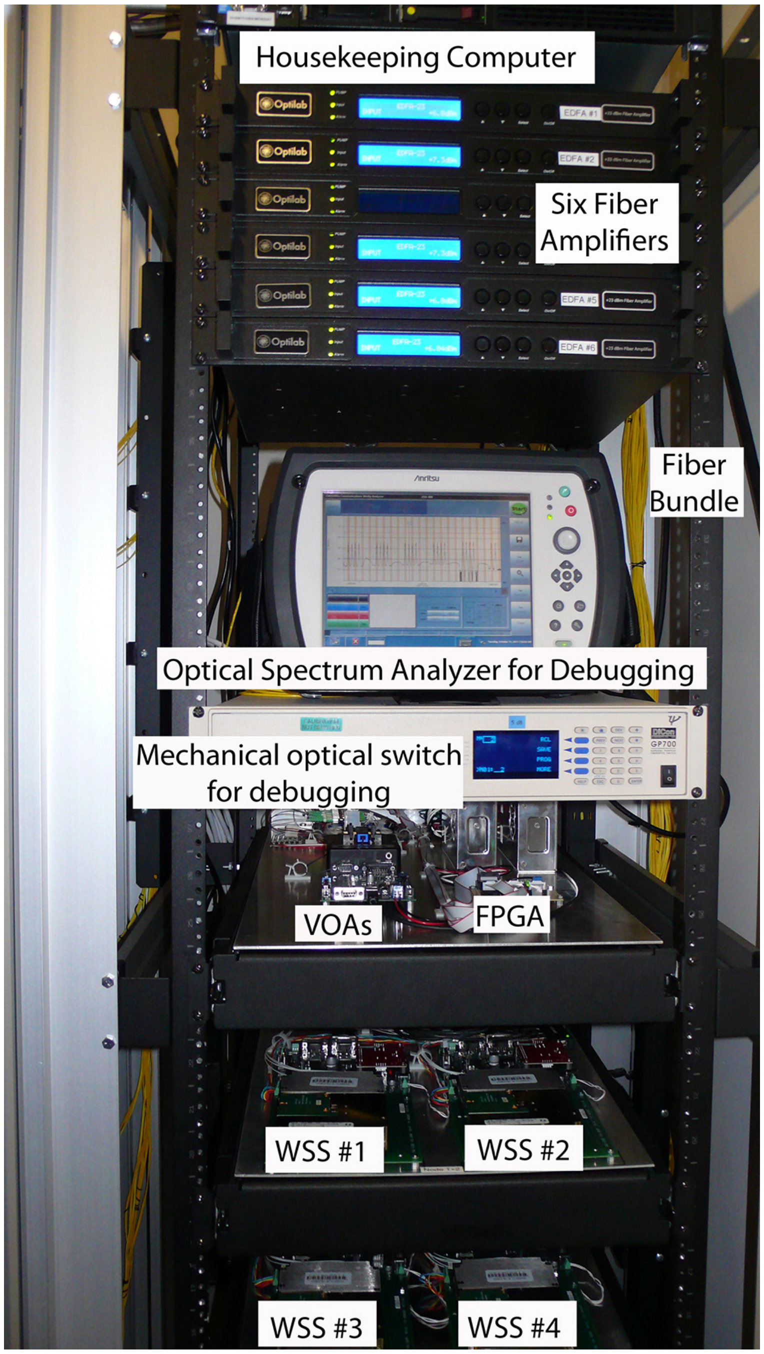


Figure 4.6: Mordia components (front view).

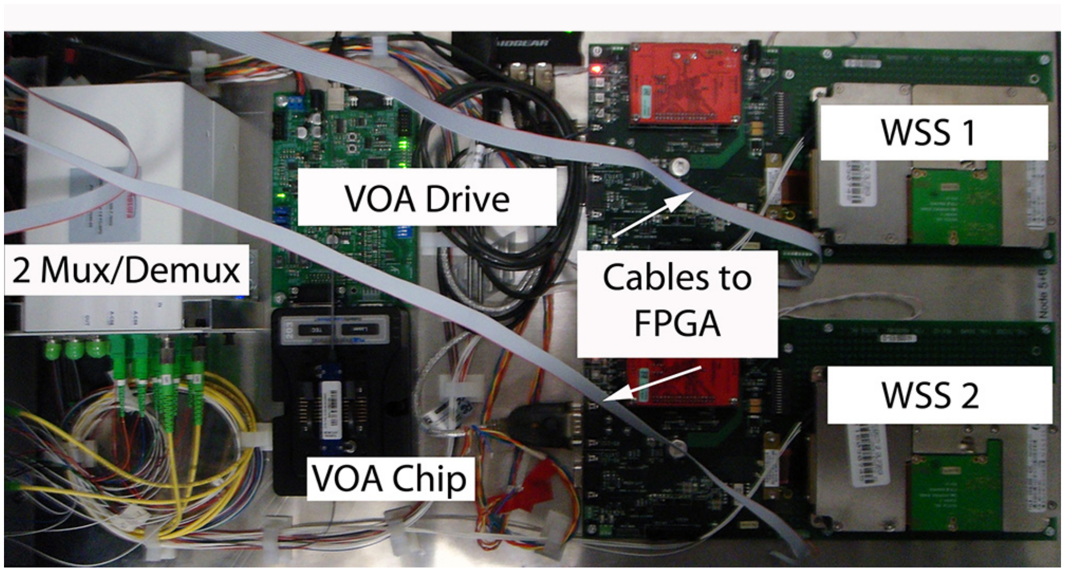


Figure 4.7: Mordia components (overhead view).

the 12 dB insertion loss of the WSS means that it is necessary to use optical amplification.

Although the 2D MEMS switch in Nistica's WSS is very fast, the interface uses an I2C bus protocol and takes milliseconds to program. The author worked with Nistica to develop a faster interface. This faster interface uses the SPI bus protocol and takes only microseconds to program. The physical control of the 2D MEMS chip is modified to minimize switching time. Other aspects of switch performance, such as a smooth switching transition, were sacrificed for faster switching speed.

The VOA has 8 channels and comes in a zero-insertion force (ZIF) package. Electrically, these VOAs are diodes, meaning that the amount of optical attenuation is controlled by the amount of current through each diode. A custom current-driver board controls the amount of optical attenuation. Each board consists of a microcontroller and a set of digital-to-analog converters (DACs). The microcontroller sets the value for each DAC and communicates this value to the Linux host via USB. One DAC maintains the temperature of the VOA chip so that the attenuation setting does not drift. The other DACs set the attenuation values.

In addition to a Xilinx Spartan-6 XCSLX45 FPGA, the FPGA board also contains an Exar XR21V1410 USB-Serial bridge, and a Marvell Alaska 88E1111 Tri-Mode Ethernet PHY. The ribbon cables connecting the FPGA board to the six WSS modules are custom made from the same gauge wire as used in old floppy-drive cables.

The Housekeeper Process is 2,600 lines of Python and 1,900 lines of PHP. The FPGA implementation is 5,000 lines of Verilog. The synthesized design occupies 826 Slices and 10 BlockRAMs, and runs at 125 MHz.

4.3.4 Scalability

The Mordia prototype uses a single ring with 24 wavelength channels to create a 24×24-port OCS. Since the C-band contains 44 DWDM channels, it is straightforward to scale the prototype to 44 ports.

Increasing the number of wavelengths on a single ring beyond 44 is more difficult. Instead of using 100 GHz spacing, it is possible to use 50 GHz, 25 GHz, or even 12.5 GHz spacing. Each smaller increment doubles the number of channels. SFP+ modules with lasers on the 50 GHz grid are commercially available meaning that it is straightforward to scale to 88 ports. However the technology to support 10G, 40G, and 100G Ethernet over narrower DWDM channels might not yet be commercially available or might not be cost competitive in a data center environment. An alternative could be to keep the 100 GHz spacing but to extend into the L-band. This would allow a doubling of the number of channels, but would make amplification more difficult.

Another way to scale beyond 88 ports is to use multiple stacked rings, with each ring reusing the same wavelength channels, as shown in Figure 4.8. Multiple independent rings can be stacked to increase the total port count. Each of the k rings has N ports. Every input port $jN + i$, where $j \in \{0..k-1\}$ and $i \in \{1..N\}$, is bundled together into a $k \times k$ ring-selection OCS before being sent to its default ring. This allows an input normally destined for one ring to arrive at a different ring. For example, an 8×8 ring-selection OCS would allow the construction of a $8 \times 88 = 704$-port OCS. It is important that all inputs assigned to the same

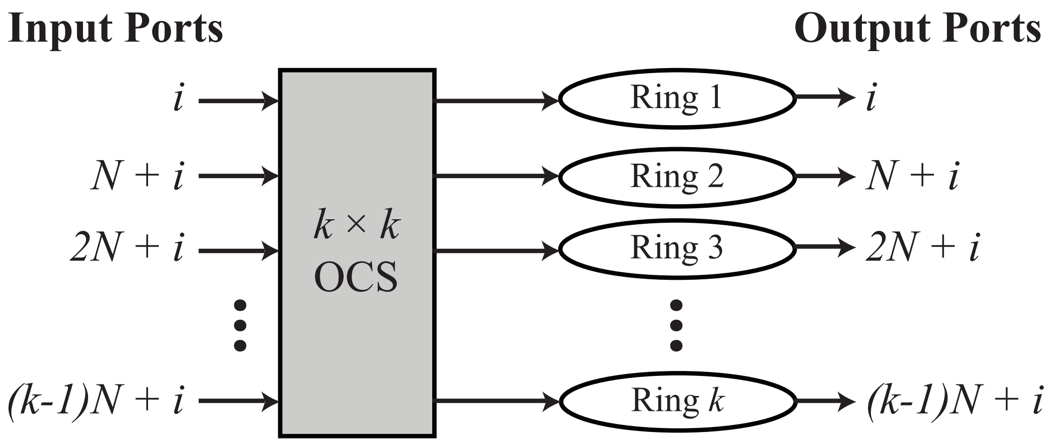


Figure 4.8: Stacked rings can increase total port count.

wavelength channel are connected to the same ring-selection OCS or else there could be a collision within a particular ring. The ring-selection OCS is only used for the input ports; the output ports are directly connected to the individual rings.

While the single-ring architecture is fully non-blocking, the stacked-ring architecture is blocking, meaning that not all input-output port mappings are possible. Fundamentally the challenge comes from reusing a finite number of wavelength channels across a larger number of switch ports. One possible solution to this problem is to introduce another degree of freedom by using tunable lasers that can transmit on any wavelength channel rather than on a specific channel. This should restore the fully non-blocking property of the OCS at the cost of additional optical and algorithmic complexity. Even without tunable lasers, the blocking version of the stacked-ring OCS can still be useful for realistic communication patterns such as all-to-all simply by using a round-robin circuit schedule.

4.3.5 Analysis

This section extends the analysis from section 3.7.1. The dwell time, T_0, is defined as the amount of time that a circuit is stable before beginning the process of tearing down that circuit and setting up a new circuit. A time slot, T, is defined as

$$T \doteq T_0 + T_1 + T_2 + T_3 \tag{4.1}$$

B	sender's minimum buffer size
D	duty cycle
F	flow size
L	link rate
N	number of switch ports
T_0	dwell time
T_1	programming time
T_2	switching time
T_3	receiver electronics initialization time
T	time slot

The duty cycle, D, is the percentage of time that an input port on the OCS is connected to some output port and is defined as

$$D \doteq \frac{T_0 + T_1}{T_0 + T_1 + T_2 + T_3} < 1 \tag{4.2}$$

The duty cycle is a useful metric because it places an upper bound on the amount of data that an OCS can forward. For example, an electronic packet switch with a 10G port can forward at most 10 Gb/s of data. But a circuit switch with $D = 80\%$ can forward at most $D \times 10\text{G} = 8$ Gb/s of data.

Given an $N \times N$ OCS, to provide full connectivity, the OCS must cycle through a minimum of $N - 1$ configurations. It is assumed that the circuit switch transitions through exactly $N - 1$ configurations and that this sequence of configurations allows every input port to connect to every output port exactly once. This ensures throughput fairness.

D and N define the capacity of the circuit connecting input port A to output port B and is given by

$$\text{capacity}(D, N) = \frac{D}{N-1} < \frac{1}{N-1} \tag{4.3}$$

For example, if $D = 80\%$ and $N = 24$, then the $A \rightarrow B$ capacity is 3.48% of the link capacity. A 10G Ethernet link would not be able to forward faster than 384 Mb/s on average. More specifically, when the $A \rightarrow B$ circuit is established it will be able to forward at 10G, but when the circuit is broken it will have a capacity of zero.

Consider an OCS configuration with an $A \rightarrow B$ circuit and a $B \rightarrow A$ circuit. Then the round-trip time (RTT) of the devices connected to bidirectional ports

A and B will be equivalent to devices that are directly connected with a cable. Now consider the case where a $B \to A$ circuit exists in one configuration, and an $A \to B$ circuit exists in the next configuration. A packet sent into port A destined to port B will need to wait $(N-1)T$ until the $B \to A$ circuit is established to allow a reply. Therefore the OCS adds a minimum of 0 to the RTT and a maximum of $(N-1)T$. In order to provide consistent performance for all connected devices, the best sequence of configurations is to increment through destination ports in a round-robin manner. For the first time slot, each input port i connects to output port $i+1 \mod N$. On the second time slot, each input port i connects to output port $i+2 \mod N$, and so on. After $N-1$ time slots the sequence repeats. This ensures latency fairness.

Unlike an EPS, an OCS does not have the ability to buffer packets, so the connected devices must assume this responsibility. How large do these additional packet buffers need to be? In the best case, a connected device would have the ability to forward or generate packets for the right destination at the right time, i.e., when the circuit is established. In this case no buffering is needed. In realistic applications, the time that a packet to a particular destination is generated is independent of the configuration of the OCS. In this case, each packet will need to be buffered for up to $(N-2)T$. Given link rate L and flow size $F = \frac{L}{N-1}$, the additional buffering B required is given by

$$B = F(N-1)(N-2)T = L(N-2)T \tag{4.4}$$

For example, let $N = 24$, $L = 10\text{G}$ and $T = 100\mu\text{s}$. Then $B = 22$ Mbits. It is important to note that this is the amount of buffering required *per port.* For example, an EPS TOR switch with four uplink ports connected to four separate OCS will require four times as much buffering.

In general, given an $N \times N$-port circuit switch, one should use the smallest dwell time, T_0, that does not result in an unacceptably low duty cycle. The reason is that increasing the slot time, T, will also increase the average RTT and the size of the buffers required per connected device. This increased T does not increase the delivered throughput as long as the offered load is less than the duty cycle.

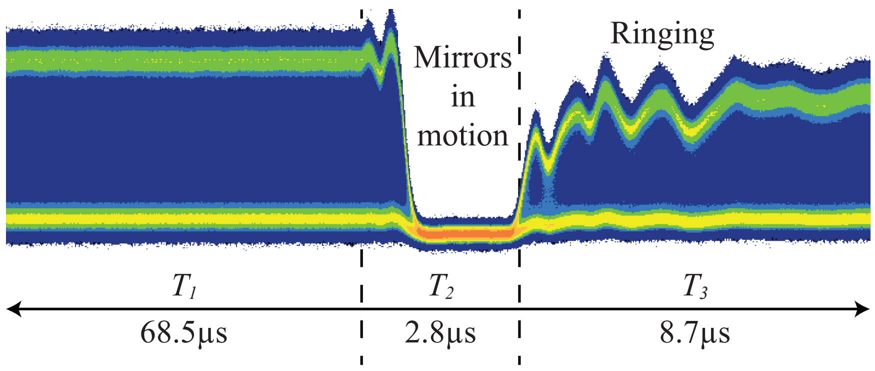


Figure 4.9: Oscilloscope trace of a WSS during reconfiguration.

4.4 Evaluation

4.4.1 Wavelength-Selective Switch Timings

In this section, the characteristics of the WSS are evaluated independently of the Mordia prototype. Two different wavelength channels are multiplexed into the input of the WSS. First, the WSS is configured to route one wavelength channel to output port 1, then is reconfigured to route a second wavelength channel to output port 1. Output port 1 is connected to a high speed optical/electrical converter and then to an oscilloscope to measure the signal over time.

Figure 4.9 shows the oscilloscope trace of the 10G Ethernet Layer 1 modulated optical signal. The solid horizontal bars above and below are times that the signal is 0 (low) or 1 (high). The shaded portion between the solid horizontal bars shows transitions between 0 and 1. On a 100 picosecond time scale, this solid blue area shows the individual 0-to-1 and 1-to-0 transitions for the 10 Gb/s signal.

Shown on the trace are measurements of T_1, T_2, and T_3 from Section 4.3.5. Parts of T_1 and T_3 are not shown. These measurements indicate that $T_1 = 68.5\ \mu$s and $T_2 = 2.8\ \mu$s. It is not possible to measure T_3 directly from the WSS outside of the Mordia prototype[5]. During T_2, the mirrors are in motion and no light seen on the output port. During T_3, the mirrors have finished moving, but mechanical vibrations cause a ringing effect that manifests as a decaying sinusoidal waveform. No effort is made to optimize the voltage signal that moved the mirrors in order to

[5]In section 4.4.3, $T_2 + T_3 = 11.5\ \mu$s is measured, and then T_2 is subtracted to determine T_3.

Table 4.3: Performance of Mordia compared to Helios.

	Helios	**Mordia**	**Speedup**
T_1	5 ms	68.5 μs	73x
T_2	12 ms	2.8 μs	4,286x
T_3	15 ms	8.7 μs	1,724x

reduce the ringing. From conversations with Nistica, it is possible to improve T_1, T_2, and T_3. Table 4.3 compares the performance of the WSS used in the Mordia prototype to the OCS used in Helios.

4.4.2 Synchronization Packet Jitter

The Packet Synchronization Process is implemented in hardware in order to minimize latency and jitter. The synchronization packets travel through an unloaded 10G Ethernet switch. The transmission delay should be close to 1 μs.

The jitter from the point of view of a Linux host directly connected to Mordia is measured. Myricom's Sniffer10G line-rate tcpdump is used to capture a total of 1,922,507 synchronization packets across 3 hosts simultaneously. The differences between consecutive timestamps are computed. T is set to 106 μs. The synchronization packets are transmitted 6 μs apart. A limitation of the line-rate tcpdump experimental setup is that timestamps are recorded with only microsecond precision. The hypothesis is that packets should arrive with timestamp deltas of either 6$\pm$1 μs or 100$\pm$1 μs.

Figure 4.10 shows the results on a semi-log plot, normalized so that the area under the curve sums to 1. 99.31% of the synchronization packets arrived at their expected times. 0.47% of the packets arrived with a timestamp delta of zero. 0.06% of the packets arrived with a timestamp delta between 522 μs and 624 μs. These outliers are attributed to the non-real-time behavior of the Linux kernel. A more accurate characterization could use a hardware-based network measurement tool such as a 10G NetFPGA or a tool from Ixia.

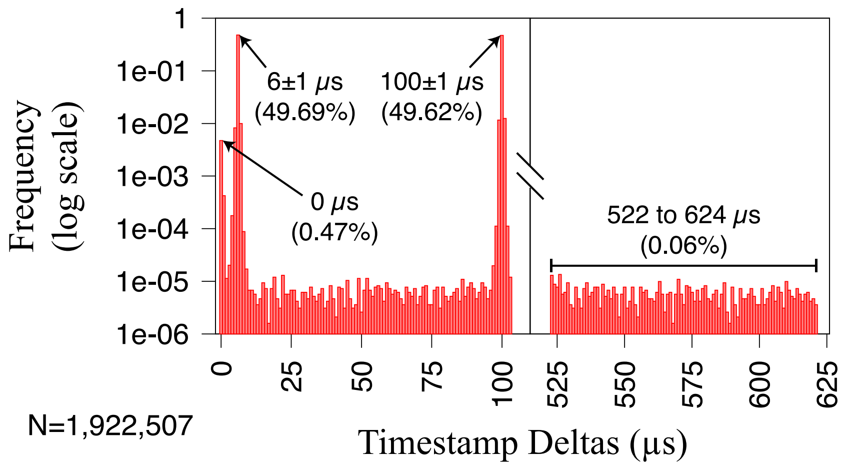


Figure 4.10: Synchronization packets received by connected devices.

4.4.3 Measurements of $T_2 + T_3$

For this experiment, four hosts are connected to the Mordia OCS prototype. One host is connected to input port 1 and transmits minimum-sized Ethernet frames at 10 Gb/s. Each frame takes 67.2 ns to transmit. For this experiment, the synchronization packets are ignored and the data packets are transmitted even when it is known that the OCS is undergoing a reconfiguration. Normal operation would not transmit during these times. Myricom's user-level kernel-bypass API is used for packet capture and transmission. Unfortunately, the non-real-time nature of the Linux kernel means that rarely the sending host will process an interrupt and will not send a run of packets. These instances are filtered out of the study. Three receiving hosts are connected to output ports 2, 3, and 4, respectively. Each receiving host uses Myricom's Sniffer10G line-rate tcpdump packet capture utility to capture the minimum-sized Ethernet frames sent by the sender. After the experiment, the packet traces are merged and analyzed. The non-real-time Linux kernel also introduces measurement errors in the three receivers.

The Mordia OCS is programmed using a static round-robin schedule. On time slot 1, input port 1 is connected to output port 2. On time slot 2, input port 1 is connected to output port 3. On time slot 3, input port 1 is connected to output port 4. Next this sequence repeats. For example, on time slot 5, input

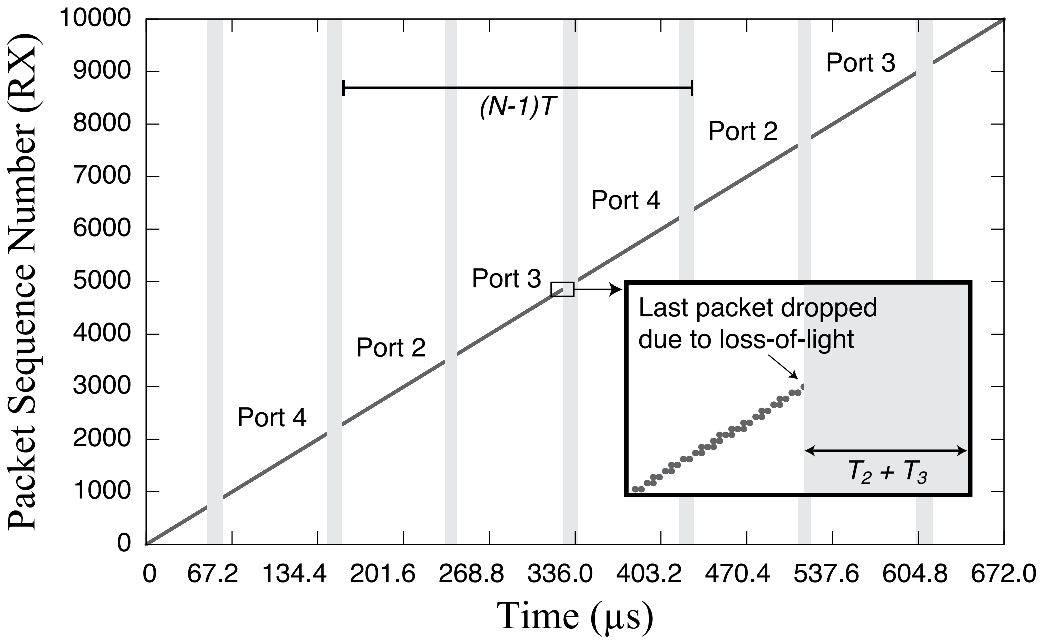


Figure 4.11: Equal-length timeslots.

port 1 is connected to output port 2 again.

Figure 4.11 shows a portion of the merged packet trace, i.e. sequence numbers of packets received by the three receivers. The gaps between runs of packets show measurements of T_2+T_3. The x-axis is time and the y-axis is packet sequence number. There are gaps in the trace where no packets are received because they were dropped while either the WSS was switching or the receiver electronics were initializing. These gaps are highlighted as gray vertical strips. An inset highlights the discrete arrival times of the packets. The last packet received by the NIC often gets dropped because it is cut off in the middle. In this case the PHY indicates an error to the MAC. The circuit scheduler is configured for $T = 106$ μs. This can be seen by the equal-length widths in Figure 4.11.

From the merged trace of approximately 1,000,000 packets, 705 gaps in packet transmission are extracted. The length of each gap is a measurement of T_2+T_3. Figure 4.13 shows the resulting histogram. The data fits a normal distribution with a mean of 11.55 μs and a standard deviation of 2.36 μs. Given a measurement of $T_2 = 2.8\mu$s from Section 4.4.1, T_3 is approximated to be 8.7 μs.

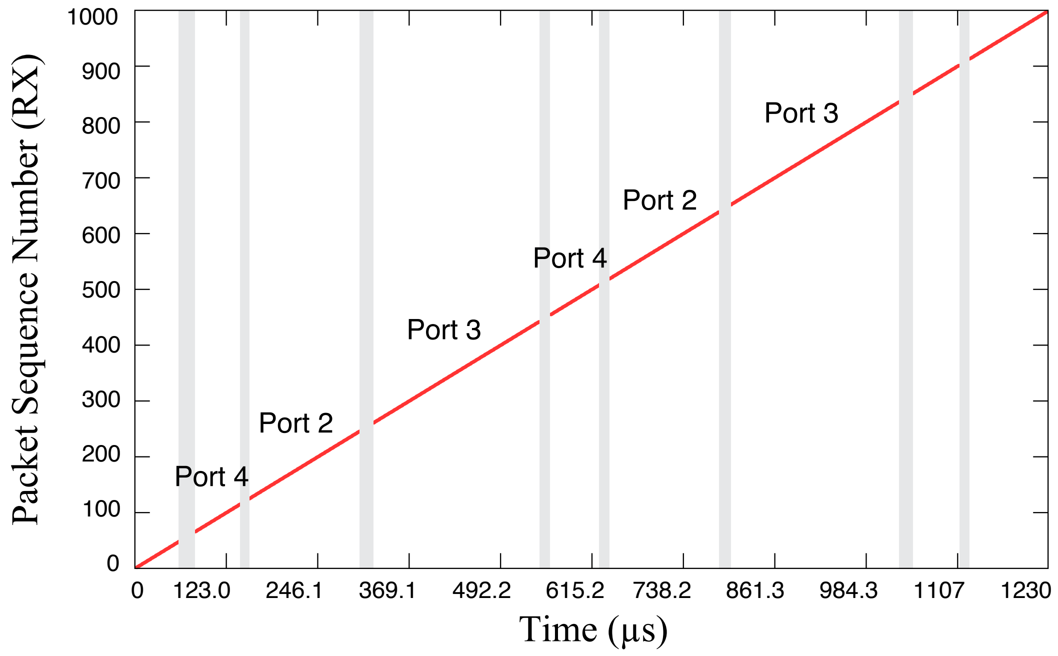


Figure 4.12: Variable-length timeslots.

Figure 4.12 shows a measurement with regular-sized Ethernet frames (1500 bytes) and variable-length timeslots (80 μs, 160 μs, and 240 μs), with the same round-robin schedule.

4.4.4 Duty Cycle Measurements

With $T = 106$ μs and a $T_2 + T_3 = 11.5$ μs, the duty cycle defined in (4.2) is equal to 89.15%. Therefore, it is hypothesized that 89.15% of the transmitted Ethernet frames will be captured. From 997,917 transmitted packets, only 871,731 packets were captured, yielding a measured duty cycle of 87.35%. In other words, there are approximately 18,000 additional missing packets. These additional missing packets are attributed primarily to periods where the sender or receiver is interrupted by the non-real-time Linux kernel.

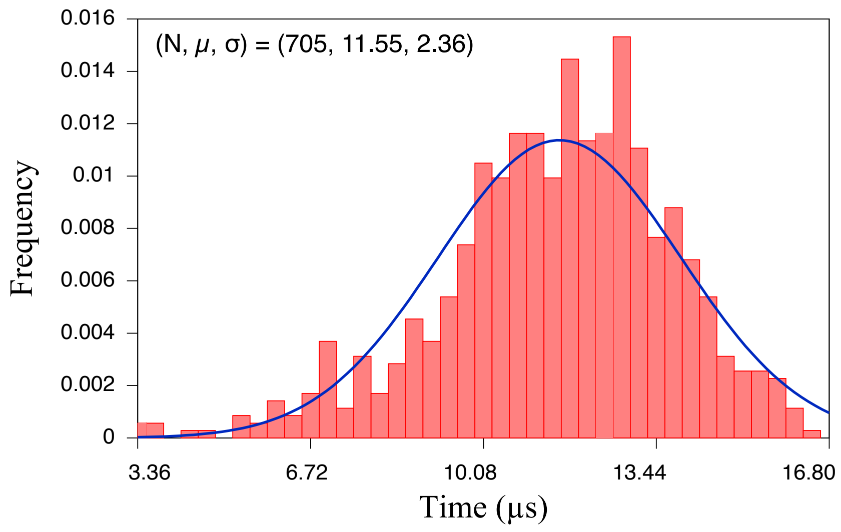


Figure 4.13: Histogram of $T_2 + T_3$ using 705 samples.

4.5 Traffic Matrix Scheduling

Previous hybrid networks [64, 75, 62, 3, 71] have all used similar circuit scheduling frameworks, called hotspot scheduling (HSS) in this dissertation. In HSS, (a) the inter-pod traffic matrix is measured, (b) the traffic demand matrix is estimated, (c) hotspots are identified, and (d) a centralized scheduler establishes physical circuits between pods to automatically offload traffic from the congested packet-switched network onto the circuit-switched network. The remaining traffic is routed over the packet-switched network. One of HSS's shortcomings is that expensive algorithms need to be run before each switch reconfiguration. Also, if a hotspot is not large enough to saturate a circuit, then the excess circuit capacity goes to waste because the circuit cannot be shared. These two limitations make HSS static and inflexible.

First-generation HSS only scratches the surface of what is possible with circuit switching in the data center. This section describes a generalization of HSS called traffic matrix scheduling (TMS), where most or even all bulk traffic is routed over circuits. TMS rapidly time-shares circuits across many destinations at microsecond time scales. In fact, TMS reduces to HSS as a special case. One reason TMS has never been proposed before is that its implementation is impractical using the millisecond time scales of previous hybrid network prototypes. It is only with microsecond time-scale circuit switches that TMS becomes possible. TMS makes circuit switching much more efficient than with first-generation HSS because much more of the network traffic can now be offloaded to circuits.

Whereas HSS couples scheduling and switching together, TMS decouples them. Like HSS, TMS uses expensive algorithms to construct a circuit switch schedule. But unlike HSS, once a schedule has been constructed, it is then implemented in hardware at microsecond time scales. This separation of scheduling and switching is what allows TMS to schedule a larger amount of traffic than HSS despite the fact that the scheduling algorithms are not any faster. In a sense, TMS does a much better job of amortizing the cost of the scheduling algorithms than HSS.

While the circuit switch is busy rapidly setting up and tearing down circuits between pods, hosts in these pods are observing the state of the circuit switch to know when to transmit packets. In other words, the hosts are using time-division multiple access (TDMA) over standard packet-based protocols such as Ethernet [76]. Hosts are required to implement variable-length timeslot TDMA for compatibility with this formulation of traffic matrix scheduling. An alternate formulation could use fixed-length timeslots. However, little will be said about host TDMA in this section and instead the focus is placed on the design and implementation of the traffic matrix scheduling algorithm.

4.5.1 All-to-All Example

This section walks through an example of TMS. Consider eight pods running Hadoop and generating a perfectly uniform all-to-all communication pattern. Fig. 4.14 (a) shows the pods physically connected to the same core circuit switch; Fig. 4.14 (b) shows them logically connected as a full mesh. Fig. 4.14 (c) shows the inter-pod traffic demand matrix with sources as rows, destinations as columns, and values as fractions of the total link rate. The diagonal is not zero because hosts send to other hosts in the same pod. Although this intra-pod traffic does not transit the core circuit switch, it is still accounted for in the traffic demand matrix.

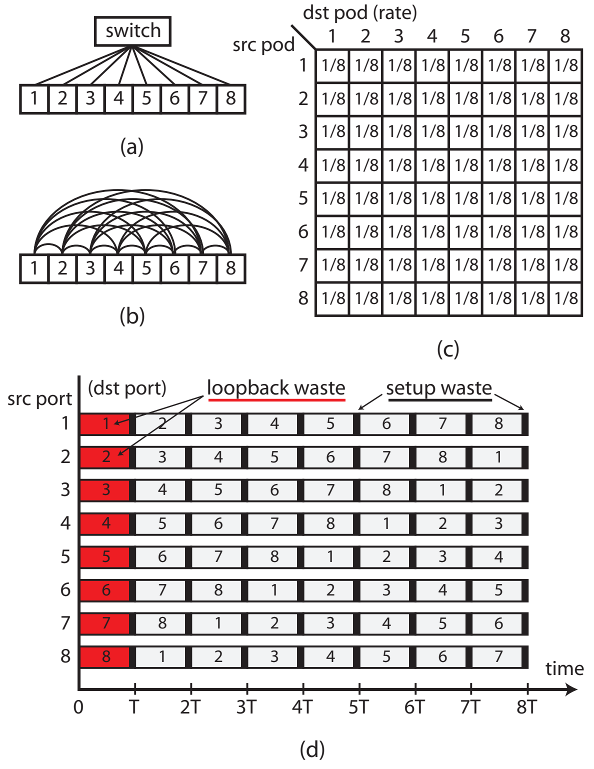


Figure 4.14: Eight pods with an all-to-all communication pattern.

This matrix is the desired transmission rate of the hosts; it is the responsibility of the network to satisfy this demand.

With HSS, each pod would require 7 physical uplinks to implement the logical topology in Fig. 4.14 (b) even though each physical link would be heavily underutilized (1/8 of the link). But with TMS a single uplink is time-division multiplexed to create 7 virtual (logical) links, each with 1/8 of the capacity of one uplink, meaning that the entire traffic demand can be routed with a single physical uplink, just like a traditional electronic packet switch.

The Gantt chart in Fig. 4.14 (d) shows a circuit switch schedule that partitions time into 8 equal-duration time slots. Over the course of the schedule, each source port will connect to each destination port for exactly 1/8 of the total time, thus implementing the logical full mesh topology in Fig. 4.14 (b) and allowing all of the traffic to be routed. The schedule then repeats. A circuit switch schedule has two sources of waste. First, loopback traffic does not leave the pod and transit the circuit switch, so any circuit switch loopback assignments are wasted, such as the assignment from $t = 0$ to $t = T$. Second, the circuit switch takes a non-negligible amount of time to switch and setup new circuits (t_{setup}), which are represented as black bars at the end of each time slot. No traffic can transit the circuit switch during this time. Reducing loopback waste requires careful scheduling whereas reducing setup waste requires using faster switching technologies.

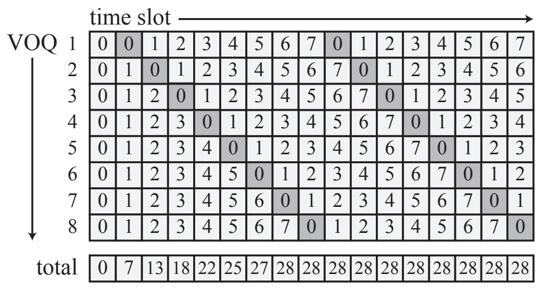
Duty Cycle and Effective Link Rate

The time during which the circuit switch is being reconfigured is called t_{setup} and the time during which the circuit switch is stable and can carry traffic is called t_{stable}. The duty cycle, D, for a circuit switch schedule with equal-length time slots, t_{slot}, is given by

$$D = \frac{t_{\text{stable}}}{t_{\text{setup}} + t_{\text{stable}}} = \frac{t_{\text{stable}}}{t_{\text{slot}}} \tag{4.5}$$

The duty cycle is important because it determines the effective link rate of the circuit switch. For example, if the nominal link rate, L, is 10 Gb/s and $D = \frac{90\mu s}{10\mu s+90\mu s} = 90\%$, then the effective link rate, $L_{\text{effective}} = 9$ Gb/s. This means that the circuit switch would not actually be able to carry the full traffic demand matrix given in Fig. 4.14 (c) but only 90% of it.

There are three ways to increase $L_{\text{effective}}$. First, one can reduce t_{setup} by using a faster switch. Second, one can increase t_{stable} at the cost of increased host buffer requirements. Third, one can increase the nominal link rate, L. This third option is especially attractive given technologies such as WDM that can place tens of 10 Gb/s channels on the same physical link. If the traffic demand is less than $L_{\text{effective}}$, then 100% of the traffic can be routed over the circuit-switched network. But once the traffic demand exceeds $L_{\text{effective}}$, the circuit switch has become a bottleneck and it is necessary to route some traffic over the electronic packet-switched network or suffer a performance penalty.



VOQ	time slot →																
1	0	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
2	0	1	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6
3	0	1	2	0	1	2	3	4	5	6	7	0	1	2	3	4	5
4	0	1	2	3	0	1	2	3	4	5	6	7	0	1	2	3	4
5	0	1	2	3	4	0	1	2	3	4	5	6	7	0	1	2	3
6	0	1	2	3	4	5	0	1	2	3	4	5	6	7	0	1	2
7	0	1	2	3	4	5	6	0	1	2	3	4	5	6	7	0	1
8	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0
total	0	7	13	18	22	25	27	28	28	28	28	28	28	28	28	28	28

Figure 4.15: Virtual output queue (VOQ) buffer occupancies.

Host Buffer Requirements

Traffic destined to hosts in the same pod is called pod-local traffic. Since pod-local traffic does not transit the circuit switch, it is possible to treat this traffic as a special case in the analysis of host buffer requirements. However, such analysis is outside the scope of this dissertation, and it is planned to extend this analysis to treat pod-local traffic specially in future work.

Each host in each pod maintains a set of N virtual output queues (VOQs)[6], one for every destination pod, including the host's own pod. All traffic generated by a host first goes to one of these queues. There can be additional queueing disciplines within a host to support arbitrary traffic engineering policies, but ultimately it is assumed that traffic waits in these VOQs before transmission. When a circuit is established, all traffic destined for that particular destination is drained from the respective queue. Fig. 4.15 shows the buffer occupancies of these VOQs of a single host in pod 1 from a cold start, in units of T. In less than one complete scheduling period a host has filled its VOQs to the steady-state level, in this case $28T$. A particular queue fills at a rate given by the traffic demand matrix until a circuit is established to the appropriate destination, then the queue is drained completely at precisely the time for the next switch reconfiguration, so that there are no standing queues. It is important to note that this example is for completely uniform all-to-all traffic; for other traffic demand matrices, the buffers will fill at different rates, but the circuit switch schedule should be chosen so that the queues are completely drained each scheduling period.

[6]Virtual output queues for input-queued switches were first described in [77].

$$\overset{\textbf{TDM}}{\begin{bmatrix} 0.375 & 0.951 & 0.732 & 0.598 \\ 0.156 & 0.156 & 0.058 & 0.866 \\ 0.601 & 0.708 & 0.021 & 0.970 \\ 0.832 & 0.212 & 0.182 & 0.183 \end{bmatrix}} \longrightarrow \texttt{Sinkhorn()} \longrightarrow \overset{\textbf{BAM}}{\begin{bmatrix} 0.094 & 0.286 & 0.521 & 0.100 \\ 0.144 & 0.173 & 0.152 & 0.531 \\ 0.279 & 0.394 & 0.027 & 0.299 \\ 0.483 & 0.147 & 0.299 & 0.071 \end{bmatrix}} \longrightarrow \texttt{Birkhoff()} \longrightarrow \begin{array}{l} \qquad\quad \textbf{Schedule} \\ 0.394\,P_1 + 0.144\,P_2 + 0.100\,P_3 + \\ 0.089\,P_4 + 0.079\,P_5 + 0.067\,P_6 + \\ 0.050\,P_7 + 0.043\,P_8 + 0.027\,P_9 + \\ 0.009\,P_{10} \end{array}$$

Figure 4.16: Example run of the TMS scheduling algorithm.

For an all-to-all workload with N pods, an effective link rate of $L_{\text{effective}}$ bits per second, and uniform time slots of duration t_{slot} seconds, the amount of buffering required by each host in bits is given by

$$B = L_{\text{effective}}(N-1)t_{\text{slot}} \tag{4.6}$$

From (4.6) it can immediately be seen why the t_{setup} of 27 ms reported by Helios makes TMS impractical for slower millisecond-scale circuit switches. For 24 pods with 10 Gigabit Ethernet, choosing $T = 270$ ms to achieve a 90% duty cycle yields $B = 7.23$ GB per host. Currently this is an impractical amount of DRAM to dedicate to packet buffering. Since $L_{\text{effective}}$ and N are both likely to increase for future data centers, the only way to make TMS practical is to decrease T by using microsecond-scale switching. Setting $T = 100$ μs yields $B = 2.74$ MB of buffering per host, which is much more practical. Therefore, TMS requires microsecond-scale (or faster) circuit switching.

4.5.2 Traffic Matrix Scheduling Algorithm

This section describes the TMS algorithm for arbitrary traffic demand matrices. Fig. 4.16 shows an example run of the scheduling algorithm on a random TDM. To summarize, in phase 1, the Sinkhorn algorithm scales an inadmissible traffic demand matrix (TDM) into a bandwidth allocation matrix (BAM). In phase 2, the Birkhoff-von Neumann algorithm decomposes the BAM into a circuit switch schedule: a convex combination of permutation matrices (P_i) that sum to the BAM (see equation 4.7).

The TMS algorithm is divided into two phases. In phase 1, the traffic demand matrix (TDM) is scaled into a bandwidth allocation matrix (BAM). A TDM represents the amount of traffic, in terms of percent of circuit switch link rate, that the hosts in a source pod wish to transmit to the hosts in a destination pod. A BAM represents the percentage of bandwidth in a circuit switch and how it is allocated between input-output port pairs over time. In a sense, the BAM is typically "larger" than the TDM since it is not often that the hosts completely drive the network at full utilization. This notion of larger is captured mathematically by the theory of stochastic matrices, even though there is nothing random about the TDM and BAM. If no pod wishes to send more than its link rate (its row sum is less than or equal to 1) and no pod wishes to receive more than its link rate (its column sum is less than or equal to 1), then the TDM is said to be both admissible and doubly substochastic. The BAM is called doubly stochastic because it has the further constraint that its row sums and column sums are exactly equal to 1. Scaling the TDM into a BAM in some sense preserves the demands of the senders and receivers, while also satisfying the constraints of the circuit switch. A matrix scaling algorithm such as Sinkhorn (1964) [78] can be used for this purpose, even when the TDM is not admissible.

In phase 2, the BAM is decomposed into a circuit switch schedule, which is a convex combination of permutation matrices that sum to the original BAM

$$\mathrm{BAM} = \sum_{i}^{k} c_i P_i \tag{4.7}$$

where $0 \leq i \leq k$, and $k = N^2 - 2N + 2$ [79]. Each permutation matrix, P_i, represents a circuit switch assignment, and each scalar coefficient, c_i, represents a time slot duration, as a percentage of the total schedule duration. One can use a matrix decomposition algorithm such as Birkhoff-von Neumann (1946) [80, 81], also known as BvN, to compute the schedule. Phase 1 is necessary because BvN requires a doubly stochastic matrix as input. In fact, the Birkhoff-von Neumann theorem states that every doubly stochastic matrix has such a decomposition.

Execution Time

The execution time of the TMS algorithm is measured when running on a 2.66 GHz Intel Core 2 processor. The TMS algorithm is tested with dense uniform random input matrices. In the future, it is planned to evaluate with actual data

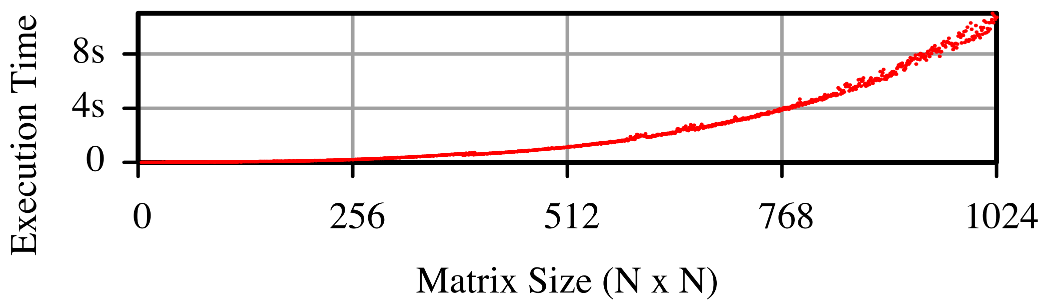


Figure 4.17: Execution time of the Sinkhorn matrix scaling algorithm.

center network traces. Sinkhorn 1964 has a time complexity of $O(N^2)$. Fig. 4.17 shows the measured execution time with input matrices up to size 1024 $\times$ 1024. BvN has a time complexity of $O(N^{4.5})$. Fig. 4.18 shows the measured execution time with small input matrices.

The algorithms were implemented in Python for ease of evaluation. Performance could be expected to improve if implemented on a faster platform such as (a) a multicore x86_64 platform with C code, (b) a GPU accelerator platform such as NVIDIA CUDA, OpenCL, or Microsoft's DirectCompute, (c) a multicore integer platform such as Broadcom's XLP, or (d) an FPGA. Second, there may be faster matrix decomposition algorithms, or faster implementations of BvN. Third, it is not yet known what N should be for real data center networks; N could be 16, 64, or 1024. Finally, dense uniform random matrices were used for evaluation rather than sparse matrices. It is likely that sparse matrices would allow opportunities for performance speedup. For example, the Hopcroft-Karp algorithm [82] used as a building block of BvN performs better on sparse matrices than on dense matrices. However, given that the community does not yet have good data center network traffic models, the evaluation is limited to uniform random traffic. Clearly there is an opportunity to improve the runtime performance of these algorithms.

Longest Time-Slot First Scheduling

Sometimes it may be better not to schedule all traffic over the circuit switch and to simply schedule only the longest time slots. The reason is that the BvN

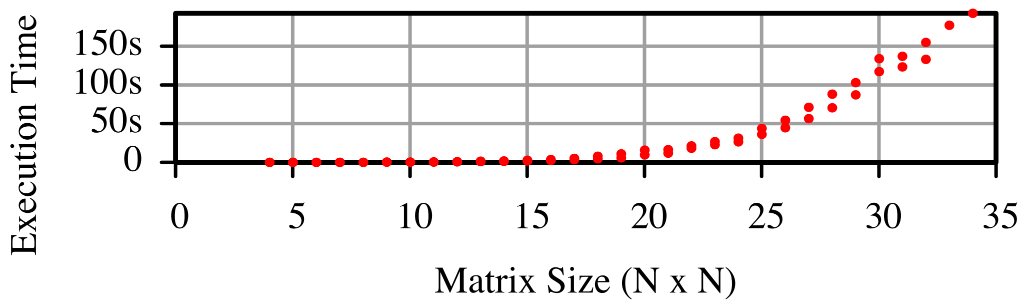


Figure 4.18: Execution time of the BvN matrix decomposition algorithm.

decomposition algorithm will generate time slots of different lengths, some of which can be quite short. Consider the schedule in Fig. 4.16. The shortest time slot is only 0.9% of the entire schedule. With such a short time slot, it is likely that t_{setup} will be so large as a percentage that it would have been better to route that traffic over the packet-switched network.

The greatest benefit comes from scheduling the first n timeslots, where n is chosen based on both the minimum required duty cycle, D, as well as the maximum allowed schedule length, T_{schedule}. The definition of D is extended to variable-length time slots as follows

$$T_{\text{setup}} = nt_{\text{setup}} \tag{4.8}$$

$$D = \frac{T_{\text{stable}}}{T_{\text{setup}} + T_{\text{stable}}} = \frac{T_{\text{stable}}}{T_{\text{schedule}}} \tag{4.9}$$

where $n \leq k$ is the number of time slots in the schedule, and T_{schedule} is the duration of the entire schedule period. This definition allows us to choose to schedule only the first n time slots. Traffic that is not scheduled over the circuit-switched network must transit the packet-switched network. Using the example in Fig. 4.16, Table 4.4 shows the trade offs in choosing the right number of time slots for the schedule. The amount of traffic sent over the circuit-switched network (CSN) vs the packet-switched network (PSN) is related to the duty cycle (D). For this example, $t_{\text{setup}} = 10$ μs and $T_{\text{schedule}} = 1$ ms.

As the n increases, an increasing fraction of the total traffic is routed over the circuit-switched network. In the limit when $n = k$, all traffic is routed over

Table 4.4: Minimum duty cycle limits the number of timeslots.

n	CSN	PSN	D
0	0%	100.0%	N/A
1	39.4%	60.6%	100.0%
2	53.8%	46.2%	98.0%
3	63.8%	36.2%	97.0%
4	72.7%	27.3%	96.0%
5	80.6%	19.4%	95.0%
6	87.3%	12.7%	94.0%
7	92.3%	7.7%	93.0%
8	96.6%	3.4%	92.0%
9	99.3%	0.7%	91.0%
10	100.0%	0%	90.0%

the circuit-switched network. However, the duty cycle decreases with increasing n. This is because T_{schedule} is held constant, so T_{stable} must decrease as T_{setup} increases. For example, if the minimum required duty cycle is 95%, then by setting $n = 5$, 80.6% of the total traffic would be routed over circuit switches. Alternatively, at the cost of increased host buffering, T_{schedule} could be increased in order to increase n to 6 or 7 while keeping the duty cycle at 95%.

4.6 Related Work

This work falls into the same category of data center network research that seeks to solve the data center network capacity and topology problems [11, 12, 35].

Mordia is an extension of the work began with groundbreaking projects such as Helios [3], c-Through [62], Flyways [64], and OSA [71] to explore the potential of deploying optical circuit switch technology in a datacenter environment. Such systems to date have all been examples of elephant hunters. An elephant hunter observes network traffic over time, detects elephant flows, and then changes the network topology (e.g. optically [62, 3, 71] or wirelessly [64]) such that more network capacity is allocated to elephants and overall throughput is maximized.

In contrast, the Mordia prototype operates approximately three orders of magnitude faster than previous work, and as such has significantly smaller periods of network measurement. The control loop must operate in less than 100 μs. With 10G Ethernet, only 82 frames of 1500-octets can be sent during this period. It might be possible to build an elephant hunter-based system at these timescales. An alternative approach is to use a simpler, static schedule that rotates circuit assignments across all input and output ports in a round-robin manner. This provides throughput and latency fairness and is ideal for all-to-all and gather/ scatter workloads.

Optical Burst Switching [67, 83] is a research area exploring alternate ways of scheduling optical links through the Internet between cities. Previous and current techniques require the optical circuits to be setup manually on human timescales. The result is low link utilization. OBS introduces statistical multiplexing where a queue of packets with the same source and destination are assembled into a burst (a much larger packet) and sent through the network together. Like OBS, the Mordia architecture has a separate control plane and data plane. But unlike OBS, the Mordia architecture uses a statically schedule rather than statistical multiplexing.

Time division multiple access is often used in wireless networks to share the channel capacity among multiple senders and receivers. It is also used by a few wired networks such ITU-T G.hn “HomeGrid” LANs and “FlexRay” automotive networks. Its applicability to datacenter packet-switched Ethernet networks was studied in [76]. HSS uses “one loop” to both compute and execute the schedule, whereas TMS uses “two loops”, one for computation and one for execution. Vattikonda et al. [76] pursue a similar two-level approach, with a slower-running central scheduler relying on very fast running TDMA in switches, however in the context of wireline data center networks.

c-Through [62, 63] treats the data center network as a single virtual output queued switch by buffering traffic on hosts and then transmitting via a bufferless crossbar circuit switch. c-Through uses these host buffers to estimate the inter-rack traffic demand matrix, and then relies on a max-weighted assignment on the circuit switch to maximize network throughput. Although operating at a pod-level rather than host/rack-level, Helios [3] also performs demand estimation, and then uses max-weighted assignment to choose circuits. Both of these fully functional hybrid data center network prototypes use HSS. In both systems, the traffic demand matrix is estimated, then communicated to a central point where the max-weighted assignment algorithm is executed, at which point the schedule is distributed back to the circuit switches and top-of-rack/pod switches for execution, all of which take a significant amount of time and reside on the critical path for circuit reconfiguration.

TMS leverages several observations made in Flyways [64, 65], showing that for many realistic deployments, such as MapReduce, there is some degree of stability in the rate of change of the traffic demand matrix. In addition, the traffic demand matrix is often sparse, which is good for the algorithms presented in this paper. Finally, the notion of focusing on the traffic demand matrix as an input to a centralized scheduler is inspired by the heat maps in Flyways.

Bazzaz et al. [72] enumerates limitations of circuit switching, such as the need to support correlated flows within an application in the context of realistic data center workloads. Suggestions include augmenting the central controller with additional application semantics, and using OpenFlow [51] to provide a richer control plane.

This dissertation is not the first to suggest using the Birkhoff-von Neumann decomposition algorithm to compute schedules for input queued switches [84, 85]. Previous work focused on crossbar scheduling for packet switches, whereas this paper focuses on circuit switch scheduling for data center networks with packet buffers distributed among the hosts.

4.7 Discussion

This chapter presented the design and implementation of the Mordia OCS architecture and 24-port prototype. A performance speedup of three orders of magnitude compared to current commercial OCS offerings is demonstrated. It was described how the Mordia architecture can be scaled to 704 ports. For comparison, the largest current commercially available OCS is 320 ports. The programming time, switching time, and receiver electronics initialization time were measured to be 68.5 μs, 2.8 μs, and 8.7 μs, respectively.

Whether industry will continue to exclusively deploy EPS technology or will eventually migrate to OCS technology is an interesting question with many unknowns. It is hoped that the results from the Mordia project will prove useful to the data center network industry to better understand the relationship between EPS and OCS technology.

Acknowledgements

Chapter 4, in part, has been submitted for publication in a paper titled "Mordia: A Data Center Network Architecture for Microsecond Circuit Switches", by Nathan Farrington, Richard Strong, Alex Forencich, Pang-chen Sun, Tajana Rosing, Yeshaiahu Fainman, Joseph Ford, George Papen, George Porter, and Amin Vahdat.

Chapter 4, in part, reprints material as it appears in the paper titled "Hunting Mice with Microsecond Circuit Switches", published in the Proceedings of the 11th ACM Workshop on Hot Topics in Networks, October 2012, by Nathan Farrington, George Porter, Yeshaiahu Fainman, George Papen, and Amin Vahdat [5].

Chapter 5

Conclusion

This dissertation presented three separate investigations into advancing the state of data center network (DCN) architecture, to provide increased amounts of aggregate bandwidth to hosts, with lower CAPEX and OPEX than is currently possible using traditional DCN architectures of networks of electronic packet switches.

The first investigation (chapter 2) seeks to make FatTree networks practical in the data center by managing cabling complexity. FatTree networks promise vast scalability while relying only on identical (and hence inexpensive) switching elements. Conventional wisdom however says that the cabling complexity burden makes such FatTree architectures infeasible in practice. To tackle the cabling complexity burden, three techniques were used. First, extreme integration was used to place as many switching elements as close together as possible in order to hide the links on PCB traces. Second, redundant components were then removed to reduce CAPEX and OPEX. Third, long cables that remained in the design were further aggregated using higher bitrates (e.g. 10 Gb/s to 40 Gb/s) and then transmitted over very thin optical fiber cable. The result was an architecture for creating a single large FatTree switch that is incrementally deployable and provides enough ports and enough aggregate bandwidth to interconnect an entire data center.

It was during this first investigation that the importance of optics in data center networks was first recognized by the author. When the author later learned of the skin effect for copper cables, it became obvious that the only future data center network architectures will be optical network architectures. Another way of phrasing this insight: if the links must be optical, then what else could be optical? Why not the switches too?

This directly led to the second investigation (chapter 3), namely, adding millisecond-scale optical circuit switches, with WDM superlinks, to the data center network in order to replace a large number of electronic packet switches (EPS) with a fewer number of optical circuit switches (OCS). Since the cost and power consumption of OCS switches is less than EPS switches, the net result is a great reduction in CAPEX and OPEX. The research challenge then is to achieve good performance with a hybrid EPS/OCS DCN, often with the additional constraints of no host modifications, no switch hardware modifications, and an implementation using commodity technologies that were not designed for this application. To summarize chapter 3, very good performance was achieved on the Helios hybrid EPS/OCS prototype, even for adversarial communication patterns, despite the fact that the hardware and software components were not designed for rapid physical reconfiguration. A model of the underlying data path was constructed and measured, and recommendations were made for how to modify future network components to better support hybrid EPS/OCS DCNs.

Later, it was discovered that there is a significant amount of data center traffic that would not benefit from millisecond circuit switching. This raised the questions of whether or not faster circuit switching could improve performance, and if so, then how best to integrate these faster circuit switches into DCN? This led to the third investigation (chapter 4). To help answer these questions, a microsecond-scale OCS called Mordia was designed and implemented, with a switching speed of over 1,000 times faster than Helios. It was discovered that in order to benefit from the faster switching speed, a different circuit scheduling algorithm was required that did not suffer from the lengthy measurement control loop of hotspot scheduling. The result was a new algorithm called traffic matrix scheduling that is able to schedule arbitrary traffic demand matrices completely over the OCS. This discovery required a new definition of "performance", since an OCS could now be as completely efficient as an EPS. It turned out that the limiting factor of traffic matrix scheduling is the minimum required duty cycle. A high minimum duty cycle could limit the number of timeslots that can be scheduled, thus preventing 100% throughput and requiring a separate EPS to offload the remaining traffic. To overcome the problems of a high minimum duty cycle, one could either decrease the OCS switching time to allow for more timeslots in the same amount of schedule duration, or increase the schedule duration at the cost of a larger amount of host packet buffering.

In conclusion, this dissertation presented three different data center network architectures for scaling the network and providing larger amounts of bisection bandwidth, while simultaneously reducing CAPEX and OPEX, and managing cabling complexity. In addition, these three architectures are complementary, and can be combined as desired for any particular data center.

Bibliography

[1] N. Farrington, E. Rubow, and A. Vahdat, "Data Center Switch Architecture in the Age of Merchant Silicon," in *Proceedings of the 17th IEEE Symposium on High Performance Interconnects (HOTI'09).* IEEE, Aug. 2009, pp. 93–102.

[2] A. Vahdat, M. Al-Fares, N. Farrington, R. N. Mysore, G. Porter, and S. Radhakrishnan, "Scale-Out Networking in the Data Center," *IEEE Micro*, vol. 30, no. 4, pp. 29–41, Jul. 2010.

[3] 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 Data Centers," in *Proceedings of the ACM SIGCOMM Conference on Data Communications (SIGCOMM'10)*, Aug. 2010, pp. 339–350.

[4] N. Farrington, Y. Fainman, H. Liu, G. Papen, and A. Vahdat, "Hardware Requirements for Optical Circuit Switched Data Center Networks," in *Proceedings of the Optical Fiber Communication Conference (OFC/NFOEC)*, Los Angeles, Mar. 2011.

[5] N. Farrington, G. Porter, Y. Fainman, G. Papen, and A. Vahdat, "Hunting Mice with Microsecond Circuit Switches," in *Proceedings of the 11th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-XI)*, Oct. 2012.

[6] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica, "A Cost Comparison of Datacenter Network Architectures," in *Proceedings of the 6th ACM International Conference on Emerging Networking Experiments and Technologies (Co-NEXT '10)*, 2010.

[7] D. A. B. Miller and H. M. Ozaktas, "Limit to the Bit-Rate Capacity of Electrical Interconnects from the Aspect Ratio of the System Architecture," *Journal of Parallel and Distributed Computing*, vol. 41, pp. 42–52, 1997.

[8] P. P. Mitra and J. B. Stark, "Nonlinear Limits To the Information Capacity of Optical Fibre Communications," *Nature*, vol. 411, pp. 1027–1030, Jun. 2001.

[9] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” in *USENIX Operating Systems Design and Implementation (OSDI)*, Dec. 2004.

[10] *Cisco Data Center Infrastructure 2.5 Design Guide*, Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706, USA, Dec. 2007, Text Part Number: OL-11565-01. [Online]. Available: http://www.cisco.com/univercd/cc/td/doc/solution/dcidg21.pdf

[11] M. Al-Fares, A. Loukissas, and A. Vahdat, “A Scalable, Commodity, Data Center Network Architecture,” in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*. ACM, Aug. 2008, pp. 63–74.

[12] 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 Data Center Network,” in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communications (SIGCOMM’09)*. New York, NY, USA: ACM, Aug. 2009, pp. 51–62.

[13] C. Clos, “A Study of Non-Blocking Switch Networks,” *Bell System Technical Journal*, vol. 32, no. 5, pp. 406–424, Mar. 1953.

[14] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, D. Hillis, B. C. Kuszmaul, M. A. S. Pierre, D. S. Wells, M. C. Wong, S.-W. Yang, and R. Zak, “The Network Architecture of the Connection Machine CM-5,” *Journal of Parallel and Distributed Computing*, vol. 33, no. 2, pp. 145–158, Mar. 1996.

[15] P. Guerrier and A. Greiner, “A Generic Architecture for On-Chip Packet-Switched Interconnections,” in *Proceedings of the IEEE Design, Automation and Test in Europe (DATE) Conference and Exhibition 2000*. IEEE, 2000, pp. 250–256.

[16] *Cabinets, Racks, Panels, and Associated Equipment*, Consumer Electronics Association Std. CEA-310-E, Dec. 2005.

[17] *Dell PowerEdge Rack Systems*, Dell, Inc., Dec. 2003. [Online]. Available: http://www.dell.com/downloads/global/products/pedge/en/rack_system.pdf

[18] *Telecommunications Infrastructure Standard for Data Centers*, ANSI Std. TIA-942, Rev. 2005, Apr. 2005. [Online]. Available: http://webstore.ansi.org/RecordDetail.aspx?sku=TIA-942:2005

[19] S. van Doorn, *Specification for SFP (Small Formfactor Pluggable) Transceiver*, SFF Committee Std. INF-8074i, Rev. 1.0, May 2001. [Online]. Available: ftp://ftp.seagate.com/sff/INF-8074.PDF

[20] A. Ghiasi, *Specifications for Enhanced 8.5 and 10 Gigabit Small Form Factor Pluggable Module SFP+*, SFF Committee Std. SFF-8431, Rev. 4.1, Jul. 2009. [Online]. Available: ftp://ftp.seagate.com/sff/SFF-8431.PDF

[21] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks.* Morgan Kaufmann, 2004.

[22] The Molex website. [Online]. Available: http://www.molex.com/

[23] L. Geppert, "A Quantum Leap for Photonics," *IEEE Spectrum*, vol. 41, no. 7, pp. 16–17, Jul. 2004.

[24] K. Greene, "Silicon photonics comes to market," *MIT Technology Review*, Aug. 2007. [Online]. Available: http://www.technologyreview.com/Infotech/19261/?a=f

[25] H. Rong, R. Jones, A. Liu, O. Cohen, D. Hak, A. Fang, and M. Paniccia, "A Continuous-Wave Raman Silicon Laser," *Nature*, vol. 433, no. 7027, pp. 725–728, Feb. 2005.

[26] *Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks, Amendment 4: Provider Bridges*, IEEE Std. 802.1ad-2005, May 2006. [Online]. Available: {AmendmenttoIEEEStd.802.1Q-2005}

[27] J. Naous, G. Gibb, S. Bolouki, and N. McKeown, "NetFPGA: Reusable Router Architecture for Experimental Research," in *ACM Workshop on Programmable Routers for Extensible Services of TOmorrow (PRESTO)*, Aug. 2008.

[28] R. N. 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 2009 ACM SIGCOMM Conference on Data Communication*, Aug. 2009.

[29] *Arista 7100S Series Data Center Switches*, Arista Networks. [Online]. Available: http://www.aristanetworks.com/en/7100_datasheet.pdf

[30] *Woven Systems Multi-Chassis EFX Ethernet Fabric Switches*, Woven Systems. [Online]. Available: http://www.wovensystems.com/pdfs/products/Woven_EFX_Series.pdf

[31] *Sun Datacenter Switch 3456.* [Online]. Available: http://www.sun.com/products/networking/datacenter/ds3456/datasheet.pdf

[32] *InfiniScale III*, Mellanox Technologies. [Online]. Available: http://www.mellanox.com/related-docs/prod_silicon/PB_InfiniScale_III.pdf

[33] J. R. Hamilton, "An Architecture for Modular Data Centers," in *CIDR*, 2007, pp. 306–313. [Online]. Available: http://dblp.uni-trier.de/rec/bibtex/conf/cidr/Hamilton07

[34] K. V. Vishwanath, A. Greenberg, and D. A. Reed, "Modular Data Centers: How to Design Them?" in *Proceedings of the 1st ACM Workshop on Large-Scale System and Application Performance (LSAP '09)*. New York, NY, USA: ACM, 2009, pp. 3–10.

[35] 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 Data Centers," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communications*. New York, NY, USA: ACM, Aug. 2009, pp. 63–74.

[36] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers," in *SIGCOMM*, 2008.

[37] "Voltaire Vantage 8500 Switch." [Online]. Available: http://www.voltaire.com/Products/Ethernet/voltaire_vantage_8500

[38] "Arista 7148SX Switch." [Online]. Available: http://www.aristanetworks.com/en/7100_Series_SFPSwitches

[39] "International Technology Roadmap for Semiconductors," 2007.

[40] R. F. Pierret, *Semiconductor Device Fundamentals*. Addison Wesley, 1996.

[41] A. V. Krishnamoorthy, "The Intimate Integration of Photonics and Electronics," in *Advances in Information Optics and Photonics*. SPIE, 2008.

[42] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. G. Beausoleil, and J. H. Ahn, "Corona: System Implications of Emerging Nanophotonic Technology," in *Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA'08)*. New York, NY, USA: ACM, 2008, pp. 153–164.

[43] M. F. Tung, "An Introduction to MEMS Optical Switches," 2001. [Online]. Available: http://eng-2k-web.engineering.cornell.edu/engrc350/ingenuity/Tung_MF_issue_1.pdf

[44] L. Y. Lin, E. L. Goldstein, and R. W. Tkach, "Free-space micromachined optical switches for optical networking," *Selected Topics in Quantum Electronics, IEEE Journal of*, vol. 5, no. 1, pp. 4–9, 1999. [Online]. Available: http://dx.doi.org/10.1109/2944.748098

[45] "Calient Networks." [Online]. Available: http://www.calient.net

[46] R. Ramaswami, K. Sivarajan, and G. Sasaki, *Optical Networks: A Practical Perspective.* Morgan Kaufmann, 2009.

[47] L. Aronson, B. Lemoff, L. Buckman, and D. Dolfi, "Low-cost multimode WDM for local area networks up to 10 Gb/s," *IEEE Photonics Technology Letters*, vol. 10, no. 10, pp. 1489–1491, 1998.

[48] Y. Lu, M. Wang, B. Prabhakar, and F. Bonomi, "ElephantTrap: A Low Cost Device for Identifying Large Flows," in *Proceedings of the 15th IEEE Symposium on High Performance Interconnects (HOTI'07)*, 2007.

[49] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *NSDI 2010 (to appear).* San Jose, Calif.: USEnix, 2010.

[50] J. Edmonds, "Paths, trees and flowers," *Canadian Journal on Mathematics*, pp. 449–467, 1965.

[51] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communications Review*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[52] J. C. Mogul and P. Congdon, "Hey, You Darned Counters! Get Off My ASIC!" in *Proceedings of the 1st ACM Workshop on Hot Topics in Software Defined Networking (HotSDN '12)*, Aug. 2012.

[53] P. B. Chu, S.-S. Lee, and S. Park, "MEMS: The Path to Large Optical Crossconnects," *IEEE Communications Magazine*, pp. 80–87, Mar. 2002.

[54] T. Yamamoto, J. Yamaguchi, N. Takeuchi, A. Shimizu, E. Higurashi, R. Sawada, , and Y. Uenishi, "A Three-dimensional MEMS Optical Switching Module Having 100 Input and 100 Output Ports," *IEEE Photonic Technology Letters*, pp. 1360–1362, Oct. 2003.

[55] L. Yoder, W. Duncan, E. M. Koontz, J. So, T. Bartlett, B. Lee, B. Sawyers, D. A. Powell, and P. Rancuret, "DLP Technology: Applications in Optical Networking," *Proceedings of the SPIE*, vol. 4457, pp. 54–61, Nov. 2001.

[56] *MAX3798: 1.0625Gbps to 10.32Gbps, Integrated, Low-power SFP+ Limiting Amplifier and VCSEL Driver*, MAxim Integrated Products, Inc., 2008. [Online]. Available: http://datasheets.maximintegrated.com/en/ds/MAX3798.pdf

[57] Y. Ohtomo, H. Kamitsuna, H. Katsurai, K. Nishimura, M. Nogawa, M. Nakamura, S. Nishihara, T. Kurosaki, T. Ito, and A. Okada, "High-speed Circuit

Technology for 10-Gb/s Optical Burst-mode Transmission," in *Proceedings of the Optical Fiber Communication Conference (OFC/NFOEC)*, 2010, paper OWX1.

[58] P. Newman, G. Minshall, and T. Lyon, "IP switching—ATM under IP," *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp. 117–129, 1998.

[59] K. J. Barker, A. Benner, R. Hoare, A. Hoisie, A. K. Jones, D. J. Kerbyson, D. Li, R. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. Stunkel, and P. A. Walker, "On the Feasibility of Optical Circuit Switching for High Performance Computing Systems," in *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing (SC'05)*, Nov. 2005.

[60] L. Schares, X. J. Zhang, R. Wagle, D. Rajan, P. Selo, S. P. Chang, J. Giles, K. Hildrum, D. Kuchta, J. Wolf, and E. Schenfeld, "A Reconfigurable Interconnect Fabric with Optical Circuit Switch and Software Optimizer for Stream Computing Systems," in *Proceedings of the Optical Fiber Communication Conference (OFC/NFOEC)*, 2009.

[61] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "SPAIN: COTS Data-Center Ethernet for Multipathing over Arbitrary Topologies," in *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI '10)*, Apr. 2010.

[62] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. Ryan, "c-Through: Part-time Optics in Data Centers," in *Proceedings of the ACM SIGCOMM 2010 Conference on Data Communication*, 2010, pp. 327–338.

[63] G. Wang, D. G. Andersen, M. Kaminsky, M. Kozuch, T. S. E. Ng, K. Papagiannaki, M. Glick, and L. Mummert, "Your Data Center Is a Router: The Case for Reconfigurable Optical Circuit Switched Paths," in *Proceedings of the 8th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-VIII)*, Oct. 2009.

[64] D. Halperin, S. Kandula, J. Padhye, P. Bahl, and D. Wetherall, "Augmenting Data Center Networks with Multi-Gigabit Wireless Links," in *Proceedings of the ACM SIGCOMM 2011 Conference on Data Communications (SIGCOMM'11)*, Aug. 2011, pp. 38–49.

[65] S. Kandula, J. Padhye, and P. Bahl, "Flyways To De-Congest Data Center Networks," in *Proceedings of the 8th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-VIII)*, Oct. 2009.

[66] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "MDCube: A High Performance Network Structure for Modular Data Center Interconnection," in *Proceedings*

of the 5th international conference on Emerging networking experiments and technologies (CoNEXT '09), 2009, pp. 25–36.

[67] J. S. Turner, "Terabit Burst Switching," *Journal of High Speed Networking*, vol. 8, no. 1, pp. 3–16, 1999.

[68] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling Internet Routers Using Optics," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'03).* New York, NY, USA: ACM, Aug. 2003, pp. 189–200.

[69] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High-speed Switch Scheduling for Local-area Networks," *ACM Transactions on Computer Systems*, 1993.

[70] N. McKeown, "The iSLIP Scheduling Algorithm for Input-queued Switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188–201, 1999.

[71] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, and X. Wen, "OSA: An Optical Switching Architecture for Data Center Networks and Unprecedented Flexibility," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI'12)*, 2012.

[72] H. Bazzaz, M. Tewari, G. Wang, G. Porter, T. S. E. Ng, D. Andersen, M. Kaminsky, M. Kozuch, and A. Vahdat, "Switching the Optical Divide: Fundamental Challenges for Hybrid Electrical/Optical Datacenter Networks," in *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC'11)*, 2011.

[73] T. A. Strasser and J. L. Wagener, "Wavelength-Selective Switches for ROADM Applications," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 16, pp. 1150–1157, 2010.

[74] "Spectral grids for WDM applications: DWDM frequency grid," ITU-T, 2012. [Online]. Available: http://www.itu.int/rec/T-REC-G.694.1-201202-P

[75] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng, "Mirror Mirror on the Ceiling: Flexible Wireless Links for Data Centers," in *Procedings of the ACM SIGCOMM Conference on Data Communication*, 2012.

[76] B. C. Vattikonda, G. Porter, A. Vahdat, and A. C. Snoeren, "Practical TDMA for Datacenter Ethernet," in *Proceedings of the 7th ACM European Conference on Computer Systems (EuroSys'12)*, 2012, pp. 225–238.

[77] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *Communications, IEEE Transactions on*, vol. 47, no. 8, pp. 1260–1267, 1999.

[78] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," *The Annals of Mathematical Statistics*, vol. 35, no. 2, pp. 876–879, 1964.

[79] L. Lovász and M. D. Plummer, *Matching Theory.* North-Holland, 1986.

[80] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumán Rev. Ser. A*, vol. 5, pp. 147–151, 1946.

[81] J. von Neumann, "A certain zero-sum two-person game equivalent to the optimal assignment problem," *Contributions to the Theory of Games*, vol. 2, pp. 5–12, 1953.

[82] J. Hopcroft and R. Karp, "An $N^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs," *SIAM Journal on Computing*, vol. 2, no. 4, pp. 225–231, 1973.

[83] C. Qiao and M. Yoo, "Optical Burst Switching (OBS) - A New Paradigm for an Optical Internet," *Journal of High Speed Networks*, vol. 8, no. 1, 1999.

[84] C. Chang, W. Chen, and H. Huang, "On service guarantees for input-buffered crossbar switches: A capacity decomposition approach by Birkhoff and von Neumann," in *Quality of Service, 1999. IWQoS'99. 1999 Seventh International Workshop on.* IEEE, 1999, pp. 79–86.

[85] C. Chang, D. Lee, and Y. Jou, "Load balanced Birkhoff–von Neumann switches, part I: One-stage buffering," *Computer Communications*, vol. 25, no. 6, pp. 611–622, 2002.