INTEGRATING OPTICAL SWITCHING INTO DATA CENTERS USING OPTICAL CROSSCONNECT INTERFACE DEVICES

A Thesis

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

in Partial Fulfillment

Of the Requirements for the Degree

Master of Science

in Electrical and Computer Engineering

by

Dmytro Chenchykov

December 2014

INTEGRATING OPTICAL SWITCHING INTO DATA CENTERS USING OPTICAL

CROSSCONNECT INTERFACE DEVICES

Dmytro Chenchykov

Approved:

Chair of the Committee
Yuhua Chen, Associate Professor
Electrical and Computer Engineering

Committee Members:

Earl J. Charlson, Professor
Electrical and Computer Engineering

Pauline Markenscoff, Associate Professor
Electrical and Computer Engineering

Cumaraswamy Vipulanandan, Professor
Civil Engineering

Suresh K. Khator, Associate Dean,
Cullen College of Engineering

Badri Roysam, Professor and Chair
Electrical and Computer Engineering

Acknowledgements

I would like to thank my advisor, Dr. Yuhua Chen, for her continuous guidance and support in this endeavor. I am also grateful to all professors who taught me the essential skills required to complete this project and the ones to come. I thank my lab mates for being great partners in learning, and my family and friends for their support.

This work was supported in part by the National Science Foundation (NSF) under Grants 0923481 and 0926006.

INTEGRATING OPTICAL SWITCHING INTO DATA CENTERS USING OPTICAL CROSSCONNECT INTERFACE DEVICES

An Abstract

of a

Thesis

Presented to

the Faculty of the Department of Electrical and Computer Engineering

University of Houston

in Partial Fulfillment

Of the Requirements for the Degree

Master of Science

in Electrical and Computer Engineering

by

Dmytro Chenchykov

December 2014

Abstract

This thesis describes a hybrid optical/electronic network architecture that overcomes the limitations of using only modified commodity hardware in such networks, and is easy for industry to adopt. Test results show significant performance improvements when the proposed system is added to a highly oversubscribed electronic network and good performance with no detrimental effects when the system is added to a traditional electronic network that is not oversubscribed. This allows for greater oversubscription ratios in the electronic network and therefore cheaper and less complex hardware. The primary contributions of this work include a method of seamlessly integrating optical switching into an existing network using custom hardware, a research platform that allows greater flexibility in the analysis of hybrid optical/electronic networks, and a software platform for simulating such networks.

Table of Contents

List of Figures

1. Introduction

The capabilities of modern computer networks, especially those in data centers, have grown dramatically in the past decade due to seemingly insatiable demand for ever more massively-parallel processing capability and distributed storage. Search engine providers have their eyes set on continually re-indexing the entire Internet; online businesses keep a complete record of customers' purchases, always coming up with new ways to take advantage of the masses of information collected; and scientists continue to deal with ever larger data sets. Thus, data centers, large collections of computers that share support infrastructure such as power, cooling, security, and communication, have become an important, quickly growing, and always in-demand part of the modern technology.

Distributed computing frameworks such as *MapReduce*, Dryad, and many lesser-known proprietary systems have allowed the efficient utilization of thousands of servers performing small pieces of a single large job. Most fundamentally, these frameworks take advantage of the fact that work can be divided into small pieces, distributed among many nodes, each of which performs a small part of the job, and results are later aggregated into a final product. These distribution and aggregation steps often create relatively localized hotspots with very high demand for network bandwidth. Modern packet-switched networking infrastructure has been struggling to keep up with demand without requiring prohibitively complex and expensive hardware. In recent times, much progress has been made by the research community in making network typologies more efficient, which led to massive network performance increases due to the deprecation

of naïve tree-structured networks and the increasing use of Fat-Trees and other significantly more efficient ways of structuring computer networks. However, networking is still a major bottleneck and resource sink in modern data centers – more efficient typologies dictate more complex wiring, making the tasks of design and maintenance increasingly difficult and expensive, requiring literally tons of wiring and thousands of switches [1]. In hopes of solving these problems, the research community is now turning to a new paradigm – optically switched networking, which promises simpler, cheaper, more energy-efficient, and higher-performance architecture for data center networks.

The key idea behind optical switching is to physically set up a direct path for the data to travel from source to destination without intermediate decoding and routing steps. In the current generation of electronic networks, every packet has to be processed by a switching or routing device at every intermediate node. In an optically switched network, a centrally controlled optical switch fabric would continuously measure demand and set up optical paths where they are most needed. A significant amount of research on this topic has been done. Systems proposed have varied in the methods of demand estimation, the presence or absence of an electronically switched overlay for handling light and latency-critical traffic, scheduling algorithms, switching fabric implementations, and topologies.

All research so far has run into the problem of modern hardware not being designed for fast optical switching. Commodity optical switches tend to be slower than the technology allows due to lack of business demand for fast switching – most

currently available commodity optical switching devices are designed and used for long term circuit switching and network configuration. Network interface cards are not generally suitable for optical switching out-of-the-box because proprietary firmware dictates them to take a long time in reconfiguration when a loss of optical link is detected, while optical link teardowns and setups must happen quickly and routinely in an optically switched network.

The quick adoption and resounding business success of Fat-Trees, VL2, and other relatively new network architectures can be attributed to the fact that they did not require businesses to completely replace existing hardware. All existing hardware could be re-used. In fact, a network operator could take 10-year old switches, rearrange the way they are connected, change the way subnets and IP addresses are arranged, add some routing rules, and have a significant improvement in performance. It was simply a matter of working smarter, not harder with existing commodity hardware and the industry was left wondering why no one thought of it sooner. In contrast, optically-switched networks as they have been described in recent research are not practical without drastic hardware changes, including some devices that have not yet been commercialized at best, and purely theoretical components at worst.

This thesis proposes the use of an Optical Crossconnect Interface (OXCI), a novel type of device that would allow data centers to keep as much hardware as possible while obtaining the benefits of an optically switched network. Essentially, the OXCI is an interface add-on that can be connected to existing switches and serve as an interface between a mostly electronic network edge and an optically switched core network. This

add-on device would aggregate packets suitable for transmission over the optical network and communicate with a central switch fabric controller to report bandwidth demand and determine when bursts of queued packets can be sent. This approach allows the system to overcome the inadequacies of multiple pieces of commodity hardware by inserting a single custom interface device. This thesis describes and analyzes a hybrid optical/electronic data center network where an optical circuit switch with a custom controller and OXCI devices at each rack functions to significantly increase the capacity of a data center network.

The rest of this thesis is organized as follows:

- Chapter 2 introduces key concepts relating to data centers and optical networking. An overview of past and current research is provided.
- Chapter 3 describes the proposed network architecture in detail.
- Chapter 4 provides a description of the simulator and model used to test the proposed system, followed by analysis of the simulation results.
- Chapter 5 provides an interpretation of the test results and summarizes the implications of this work.

2. Background

To understand the proposed method, with its benefits, drawbacks, and implementation details, one must understand the context in which this work is presented. To that end, this section provides an overview of the state-of-the-art of data center networking and optically-switched networking research.

2.1. The Data Center

A data center is a large collection of computers that share resources including networking infrastructure, locations, power systems, cooling systems, security, and management. Large groups of computers in data centers often work together to achieve a common goal such as performing large distributed calculations, serving a high-traffic website, or processing extremely large data sets. The history of data centers can in some respects be traced to historic mainframe computers that required entire rooms to house the computer and all support equipment. The mainframes of old eventually evolved into the modern supercomputers, retaining tight coupling and shared memory between processing units. Modern data centers inherited much of the philosophy about support infrastructure such as cooling and power from mainframes, but are very different in their core functionality. Modern data centers are usually not thought of as monolithic computers as mainframes and supercomputers are, but are large collections of individual servers. Thus, the coupling between processing resources is relatively loose, and each processing node is a self-sufficient server, usually having its own power supply, processing units, volatile and nonvolatile storage, and network interfaces. In recent years, data centers have become increasingly important as utilization of computing

power has been shifting from heavy use of local resources such as consumer desktops or servers owned and operated by businesses to cloud computing and computing-as-a-service. Some examples of this trend are the recent shift of users from desktop-based e-mail clients like Outlook to web-based mail and the increasing popularity of services such as Amazon EC2, where users and organizations can rent large pools of computing power on demand, without having to worry about the infrastructure, maintenance, and design of their own computing resources.

2.1.1. Big Picture: The Anatomy of a Data Center

Data centers are characterized by two key concepts: pooling of computing power and sharing of infrastructure. A typical data center is made of thousands of individual servers. Usually, large groups of these servers must be pooled together to achieve a single large task, such as creating an index of the entire public World Wide Web.

Sharing infrastructure is important because it is what allows data centers to be so cost-effective and efficient. All computers in a data center have a common power distribution system, which allows for the efficient use of power and backup power capabilities. Most data centers have battery banks and diesel generators that would allow operations to continue without interruptions in case of power grid failures.

Cooling systems are another infrastructural consideration. When thousands of computers are pooled together in one building and each of them is dissipating hundreds of watts of power, the building will overheat very quickly. To keep the ambient temperature down and the servers operating within temperature specifications, data centers use large cooling units, often termed Computer Room Air Conditioning, or CRAC

units. These machines transfer heat from the surrounding air to a coolant fluid which is pumped outside and cooled by large chiller of drycooler units in a similar manner to normal air conditioning, and expel cooled air back into the data center through some managed airflow system. Air flow management is important because allowing cold air to mix with hot air drastically reduces cooling system efficiency. Methods employed for controlling airflow include pressurizing the area under a raised floor with cooled air and letting the air out through strategically placed vents, or using a system of specially designed air conduits.

Maintenance overhead is also often reduced by the use of a single common performance monitoring and management system for the data center. Operators can monitor the performance of the data center as a whole. Designing the servers to be extremely reliable would be prohibitively expensive. Instead, data center-grade servers are usually not much more reliable than desktop computers, and of thousands of units, some are expected to fail almost daily. In a well-designed data center, a common management and monitoring system quickly detects these failures, taking some automated steps to move the failed machine's workload to another machine and begin resolving the issue, possibly alerting a human operator to physically replace the failed unit [2].

A shared local network is the fabric that ties the data center together and allows many computers to contribute to the completion of a common task. Fast, scalable, well-managed networking infrastructure facilitates efficient management and monitoring of the data canter, large scale data processing, distributed storage, and high reliability.

Recently, there have been many advances in the architecture of data center networks, with ideas about incorporating optical switching into the data center being one of the most prominent research topics is the field.

2.1.2. Typical Workloads: The Purpose of a Data Center

Although servers within data centers can technically perform the same kind of local single-machine computation tasks that desktop computers often perform, the typical workload of data center servers is somewhat different. Most modern data centers are utilized for massive computation jobs such as math-heavy scientific processing – for example, large scale image processing involving distributed Fast Fourier Transform (FFT) calculations. Serving massive, user heavy web-based applications such as Google maps or "Yahoo!" web search is also a very common workload. Perhaps most importantly, data centers are used for large scale data mining and processing, such as building web search indexes, analyzing usage patterns of online shopping sites, and determining the effectiveness of targeted advertising placement. Management tasks, such as backups and virtual machine migration also make up a significant portion of the work done within modern data centers.

One of the most common applications seen within data centers is the *MapReduce* [3] parallel computing framework and similar tools such as Dryad [4]. This kind of application is often the go-to example and test workload for many data center related research studies due to its ubiquity, versatility, and high importance to the biggest players in the industry, such as Google, whose web indexing engine is a *MapReduce* program. *MapReduce* was born from the need to perform often

straightforward operations on extremely large and complex data sets – for example, counting the number of times a word occurs on all crawler-accessible web pages. This kind of task has long been the bread and butter of many Internet giants, and much code has been written to simplify it. The problem itself is extremely simple – simply read documents and increment a counter each time a certain sequence of characters is encountered. In practice, this cannot be done by a single computer because the data set is extremely large. The solution is as simple as the problem – split the data set into smaller chunks, and have each computer count the number of times the word occurs in its assigned chunks. Even if the data set is a petabyte in size, a reasonable estimate for the currently indexable Web, a cluster of 2000 servers can split the data set and have each server process 500GB of information. Given that the data has already been stored on the appropriate local disks within each server, something that can be arranged during the data collection process, assuming processing rates of 40MB/s for each server this task can be completed in under four hours. Having to do many similar large data mining tasks, one will notice that each task shares the same set of "boilerplate" logic. Any program running this kind of job will need perform the following tasks:

1. Split the job into pieces and assign each piece of the work to a server.
2. Have each server compute its intermediate or partial result.
3. Monitor job progress and server performance, gracefully dealing with server failures as they occur.
4. Aggregate intermediate results, computing the final result.
5. Managing many other common book keeping details.

In recognition of these similarities in commonly performed data center work, Google developed the *MapReduce* framework, which in their own words, "hides the messy details of parallelization, fault-tolerance, data distribution, and load balancing in a library" [5]. Tasks numbered 2 and 4 are roughly corresponding to "map" and "reduce" respectively, and form the user logic of the program.

The reality of *MapReduce* is somewhat more complicated and abstract than the simple description above. According to the canonical description of *MapReduce*, the user is responsible for writing two functions. The *Map* function takes a takes a key/value pair as an input and processes it, producing an intermediate result in the form of a list of other key/value pairs, often of different data types and taken from a different value domain than the original input pair. Each execution of the *Reduce* function accepts a single key K_{INTER} from the set of intermediate results and the set of values for that key. The reduce function then processes its data and generates a result, often a single value or some other data set much smaller than the input. Mathematically, given a set of M input key/value pairs,

$$\{(K_i^{INPUT}, V_i^{INPUT}), (K_{i+1}^{INPUT}, V_{i+1}^{INPUT}), \ldots, (K_M^{INPUT}, V_M^{INPUT})\} \quad (1)$$

for each of M inputs, a mapping is performed generating $N.i$ new key/value pairs

$$MAP(K_i^{INPUT}, V_i^{INPUT}) \rightarrow \{(K_{i,j}^{INTER}, V_{i,j}^{INTER}), (K_{i,j+1}^{INTER}, V_{i,j+1}^{INTER}), \ldots, (K_{i,N.i}^{INTER}, V_{i,N.i}^{INTER})\} \quad (2)$$

followed by the reduction mapping executed for each intermediate key,

$$REDUCE(K_j^{INTER}, \{V_i^{INTER}, V_{i+1}^{INTER}, \ldots, V_M^{INTER}\}) \rightarrow V_j^{FINAL}, \quad (3)$$

where V_j^{FINAL} is the final answer for the intermediate key K_j^{INTER}. Let N be the total number of unique intermediate keys generated. The final answer data set can be

generalized to the list of key/value pairs

$$\{(K_j^{INTER}, V_j^{FINAL}), (K_{j+1}^{INTER}, V_{j+1}^{FINAL}), \ldots, (K_N^{INTER}, V_N^{FINAL})\}. \quad (4)$$

Data center processing frameworks, including *MapReduce*, are designed to be fault-tolerant, even to the point of being able to gracefully handle entire racks of servers becoming unavailable in the middle of a job. This is achieved by the master node detecting a failure and moving the affected work to other available nodes.

Along with parallel processing frameworks, data centers employ highly-distributed redundant storage systems like GFS [6]. Such systems ensure the availability of data in case of server or even rack failures. Several copies of every piece of data within the system are distributed throughout the network.

The administration and monitoring done within a data center can often be viewed as a workload in itself. Prime examples of this are virtual machine migration and software updates, which take up significant network bandwidth and CPU time. In the context of this thesis, one should keep in mind that improving the performance of such common management tasks can result in significant performance improvements in the data center as a whole. Thus, such tasks are also of interest in modeling and observing the performance of data centers.

2.1.3. Data Center Networking: What Holds it All Together

Data center networks are connected by a hierarchy of switches. The analogy of a tree is of some use in this context. The smallest “branches” of this network tree are the individual servers within the data center. These server nodes are usually organized in racks, with each rack holding tens of servers. For simplicity, it is assumed that each

server has a single network interface for communication with other nodes. Each rack also contains a network switch that routes traffic within the rack and serves as the gateway for inter-rack communication. This is called a Top-of-Rack, or "ToR switch". Recent developments in data center designs have introduced the concept of a "pod", which can be thought of either as a bigger rack containing hundreds instead of tens of servers, or an aggregation of several racks into a bigger structural unit. In the context of this thesis, terms "pod switch" and "ToR switch" are used interchangeably because the main concern here is the qualitative concept of the level of traffic aggregation above individual servers, but below the network core. ToR switches are commodity switches with tens to hundreds of ports with the lowest bandwidth capability in the network**.** In practical networks, ToR switches are usually connected to another layer of aggregation switches, most often higher performance switches than the ToRs. Most research articles leave out this layer for simplicity. The next level, making up the "root" of the tree is the network core. The core has traditionally been composed of very expensive high-bandwidth switches that route traffic between the aggregation switches. The core switches also act as gateways to wide area networks, and eventually, the Internet backbone.

2.2. Optical Networking

The use of lasers and fiber optic cables to transmit data originally developed as a way of achieving communication over long distances. In computer networking, optical components have provided the building blocks of long haul and extremely high-bandwidth Internet backbone networks. Recently, optical networking technologies have

been quickly making their way into corporate networks and data centers, and there has been significant research effort into making these technologies more suitable for use in non-backbone networks. Currently, the most common use for optics in the data center is high-bandwidth data transmission between electronic switches, due to the fact that data signals are getting too fast for electronic cables to handle efficiently over long distances. Research has been focusing on using optical switches to unload electronic switches and routers, making high data-rate transmissions more efficient.

2.2.1. Switching Mechanisms

Three primary switching mechanisms within networks have been identified and studied by the research community. These mechanisms are packet switching, circuit switching, and burst switching. Theoretically, each of these can be implemented in the optical or electronic domain, although optical packet switching is not currently practical, and electronic circuit switching along with electronic bust switching are generally not seen as useful.

The traditional method most commonly used by the networking industry is electronic packet switching (EPS). Packet switching is defined as having packet-level granularity, where a packet is usually around 1500 bytes of data. This means that a separate routing decision can be made about every individual packet traversing the network at every node it passes. In electronic packet switching, this means that the packet's header is interpreted by the router and the next hop of the packet is determined by a routing algorithm based on the packet's final destination and the previous information the router has about the state of the network. If a packet to be

routed through EPS is transmitted as an optical signal, which is often the case, it must be converted to an electronic signal for processing and possibly back to an optical signal for transmission to the next-hop node. This requires expensive transceivers and a significant amount of power. Even if a packet is transmitted electronically, the high bit rates used by modern electronic Ethernet also require a lot of power and complex cable construction to ensure signal integrity. One of the ultimate goals of modern networking research is optical packet switching – a theoretical method in which routing decisions can be made about each packet individually at each node without converting the packet to an electronic signal. However, this has not been achieved with current technologies, so it is necessary to consider other switching methods that allow the packets to be kept in the optical domain.

Optical burst switching has been the subject of much research emphasizing its potential benefits to long-haul backbone infrastructure. In optical burst switching, packets are queued until a large burst of packets with the same destination is collected. When the burst is ready to be sent, a control packet traverses the network, instructing core nodes to set up direct optical paths in anticipation of the burst's arrival. The burst itself is sent shortly after the control packet, such that by the time the burst arrives at any node, the node would have already processed the control packet and set up the appropriate optical path for the burst. Optical burst switching in its current incarnations involves significant scheduling complexity and extensive use of wavelength division multiplexing, as well as the possible necessity of wavelength conversion. Generally, optical burst switching networks deal with the need of bursts to travel through several

purely optical core nodes without optical-electronic-optical conversion. Implementation of true optical burst switching is currently seen as too complex and expensive for data center implementation. However, optical burst switching in simpler forms may also be seen as a subset of optical circuit switching where the circuit lifetime is relatively short.

Optical circuit switching has been traditionally thought of as the ability to set up a direct optical path between two nodes for a relatively long time. In the trivial case this could involve a technician physically connecting two nodes to each other with an optical cable. More realistically, in modern networks, this corresponds to automated switching hardware configuring a direct optical path within the network either because some client has purchased the use of a direct optical link between specific machines, or because a network administrator (human or machine) has decided that such a direct link would be beneficial to performance. Traditionally, optical circuits have had lifetimes counted in days, months, or even years. However, more recently, optical link durations have been getting shorter. Ongoing research into data center networking has utilized optical circuits with lifetimes as short as tens of microseconds. There is no longer a hard semantic distinction between optical circuit switching and optical burst switching, but burst switching most often implies control packets going ahead of the burst through a complex core network with the possibility of wavelength conversion, while fast circuit switching most often refers to the case of a relatively simple optical core. The OCS core is viewed as a single optical switch and not multiple switching nodes. It is reconfigured by a single central controller based on a centralized control algorithm in which the controller schedules and sets up optical circuits between nodes based on its perceptions

of present and future traffic demand. Such fast optical circuit switching is the subject of this thesis.

2.2.2. Switching Hardware

Optical switching is essentially the act of setting up a direct optical light path between two optical transceivers. Several devices are available for achieving this. The most common device is a MEMS (Micro ElectroMechanical System) switch. These devices contain several ports and a set of physical mechanically movable mirrors that can route the laser beam being used for communication from an input port to any output port.

Another common switching device is the wavelength-selective switch (WSS). This device is a reconfigurable optical bandpass filter that can separate a range of wavelengths from an incoming signal containing many wavelengths from many sources multiplexed into a single fiber. A controller can use this device to select which data is going to which destination in much the same way this is possible with a MEMS switch.

An important consideration in this area is the amount of time it takes to reconfigure a switch. WSS switches are generally orders of magnitude faster to reconfigure than MEMS switches, but they can only operate on limited sets of wavelengths, while MEMS switches are completely wavelength-agnostic and can generally switch signals carrying vast amounts of data on as many as 100 wavelengths.

2.3. Current Research in Optical Data Center Networking

The research community has been focusing on finding ways to use optical components to augment electronic networks. Of particular interest is the idea of

offloading traffic-heavy slowly changing flows from the electronic switch to optical circuits. This is most often achieved by adding an optical switching device to the core of the network and finding some way to monitor demand for high-traffic connections.

2.3.1. Helios

Helios [10] is one of the first prototype networks combining electronic packet switching with optical circuit switching to improve data center networking efficiency. It achieves this by adding an optical switching fabric on top of an oversubscribed electronic network. The research objective in Helios was to figure out a good way to manage the allocation of the optical switching resources and measure the performance improvement attained, comparing the cost and complexity of the novel system to an analogous traditional system. The key observation motivating Helios has been the fact that in current data center networks the designer must either go to great expense to

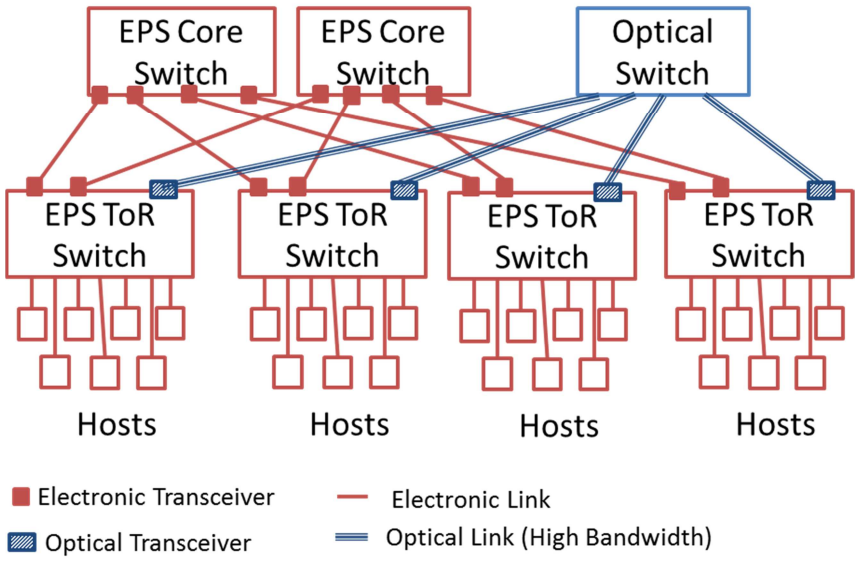


Figure 1: The Helios network topology.

build a complex network able to handle worst-case traffic scenarios and allow most of the network to remain idle under most normal workloads, or run the risk of workloads being bottlenecked by inadequate network bandwidth. Rather than statically provisioning for the worst case, Helios provides a means of augmenting the network with a pool of bandwidth that can be allocated where it is deemed necessary and constantly reconfigured based on changing traffic patterns.

The Helios network is architected as a two-level multi-rooted tree. There is a level of top-of-rack switches aggregating traffic from individual hosts, and a core level, composed of a mix of optical and electronic switches, as shown in Figure 1. Note that there is more than one possible path through the core between each pair of top-of-rack switches – each top-of-rack switch has connections to multiple optical switches. Most importantly, the network is designed to allow the strengths of the optical switches to compensate for the weaknesses of the electronic switches and vice-versa. When high-bandwidth connections are necessary, the optical switches are configured to accommodate this need by taking on the heaviest flows of traffic. When traffic is bursty and granularity in switching is necessary, the electronic switch handles this less bandwidth-intensive demand. In the design of such a network there is a tradeoff to be made between the resources invested in the electronic part of the network and the resources invested into the optical part – a decision to be made based on the anticipated workload characteristics of the data center. The example network in Figure 1 delivers full bisection bandwidth. However, only half of this bandwidth is usable at the packet granularity, and the other half must be allocated by the optical switch manager

based on traffic demand measurements.

The Helios prototype that was constructed and tested consists of 24 rack-mount servers used as hosts on the network, several commercial electronic packet switches, and a 64-port Glimmerglass optical circuit switch. The optical and electronic switches were subdivided into smaller switch topologies as necessary. The optical switch was divided into up to 5 virtual 4-port switches, and the ToR switches were actually not independent, but subdivisions of a single optical switch. This allowed greater flexibility in configuring such a small-scale topology, making the prototype more realistic and allowing more efficient use of available hardware resources.

The software implementation of Helios consisted of three primary components. The circuit switch manger (CSM) was the unmodified control software of the Glimmerglass switch. It provided an interface for managing the configuration of the optical switch, but did not provide any feedback about when the switch started or completed reconfiguration. The Pod Switch Manager (PSM) was a user-level program running on each ToR switch (or pod switch, to use Helios terminology). Its primary functions included initializing the switch hardware, managing the flow table for the network, and communicating with the optical topology manager. The PSM was responsible for multiplexing traffic between the optical network and the electronic network. It achieved this with the help of Link Aggregation Groups (LAGs). A link aggregation group is a way of combining multiple network connections in parallel – for example, when a single pair of pod switches utilizes multiple optical links for communication. The PSM on each pod switch kept a LAG for every other pod switch in

the network. Whenever an optical link was set up to another pod switch, the PSM would add the newly set up link to the LAG of the corresponding pod switch. Whenever an optical circuit was disconnected it would be removed from the corresponding LAG.

The most complex and important part of Helios is the topology manager (TM). It is a software program running on a central management server, responsible for estimating traffic demand within the network based on data gathered from each PSM, and continuously recalculating the optimal switch configuration for the network based on demand. Once the optimal configuration is calculated, the TM notifies the PSM of a reconfiguration event and sends commands to the CSM to achieve the switch reconfiguration.

The control loop run by the topology manager consists of six steps:

1. Measurement of the traffic matrix: this is achieved by polling each PSM for flow rate data.
2. Demand estimation: in this step, the TM takes flow rate data collected from the PSMs and calculates the max-min fair bandwidth allocation for each TCP flow. This is a measure of what the flow rate of each flow would be if it were traversing an ideal non-oversubscribed packet switch. In other words, this is a measure of how much bandwidth the flow would use if it could have unlimited bandwidth.
3. Optimal topology computation: in this step, the TM computes the optimal network topology given the latest demand estimate. The problem formulated mathematically is a set of graphs, one graph per optical virtual switch, matching

source-destination sets of vertices with edges to achieve the maximum total edge weight, where edge weights are assigned from the estimated demand metrics computed in the demand estimation step.

4. Notify down: the TM notifies all PSMs to stop sending packets to the optical switch while it is being reconfigured in order to avoid dropping packets by sending them into a vacuum.
5. Change topology: the TM sends the reconfiguration commands to the CSM and waits for the reconfiguration to complete.
6. Notify up: The TM notifies the PSMs of the new topology and signals them to add the newly configured links to LAGs in accordance with the newly configured topology. The PSMs now start sending packets – through the optical network when they are going to a destination to which an optical path is available and through the electronic network otherwise.

The Helios team initially wanted to test the prototype using an implementation of *MapReduce*, since it is a real application very commonly used in modern data centers. However *MapReduce* only produced an aggregate throughput of at most 50 Gbps, which was not enough to achieve the goal of stressing the network to its limits. To solve this problem, the Helios team decided to use synthetic traffic generators, which are able to supply arbitrary amounts of traffic because they are not constrained by CPU and I/O speeds like real applications. The algorithms are parameterized by a *stability* variable, which indicates the period of time the generator generated a uniform traffic pattern before moving to the next iteration. Three synthetic traffic generation algorithms were

used:

- Pod-Level Stride (PStride): Each host in a source pod sends one TCP flow to each host in a destination pod, rotating destination pods after every stability period.
- Host-Level Stride (Hstride): Each host sends 6 flows to another host. The hosts rotate after each stability period, allowing for a gradual shift of traffic from one pod to the next.
- Random: Each host sends 6 flows to another random host in a different pod. Note that hotspots occur when multiple sources are sending to the same destination.

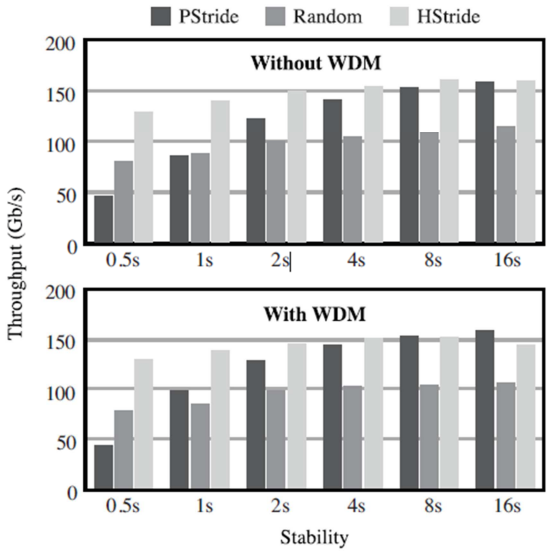


Figure 2: Test results from [Far10].

The results of these synthetic traffic trials are shown in Figure 2. It is clear from these results that the performance of Helios relies heavily on the characteristics of the workload – specifically, on the presence of long, sustained flows between pod switches. Bulk file transfers and virtual machine migration are ideal jobs for Helios, but

performance from other workloads where the traffic pattern changes slowly may also benefit.

Another important factor for the performance of Helios is the directionality of flows traversing optical switches. Since symmetric flows in which the traffic volumes going in both directions are similar but not common in practice, Helios links should be unidirectional. This implies that in a TCP flow, the bulk the data transfer travels through an optical link, while lower-volume acknowledgment and control traffic goes through the electronic packet switches. The traditional network is a non-oversubscribed electronic network used as an upper bound of performance. The goal of Helios is not to outperform the fully provisioned electronic network, but to provide similar performance with significantly less cost and complexity.

Some of the most important information gleaned from Helios has been the discovery of a set of inadequacies in the available hardware and corresponding control software when used for hybrid networking, admittedly, a purpose none of the tools were designed for. Most of the problems observed are relatively standard engineering challenges which can be solved in the design of hardware. Thus, one of the biggest contributions of Helios has been the presentation of new possible use cases for network hardware vendors, and especially optical switch vendors to consider. For example, it is noted in [10] that switching speed in the prototype was limited not by the laws of physics governing MEMS switching elements, but sub-optimal switch control software. Another example of this is a set of specific optimizations which improve the operation of devices when used for their intended purpose within purely electronic networks, but

hinder the operation of a hybrid network – these features include electronic dispersion compensation, link-down event handling, debouncing, and Ethernet's assumption that all links are bidirectional. All of these features can be modified to function properly within hybrid networks or at least not interfere with their operation, and Helios is the prototype which first shed light on this set of issues for equipment vendors and the research community to consider.

2.3.2. C-Through

C-Through [12] is another test platform for a hybrid electronic/optical switching architecture. Like the previously discussed Helios, it incorporates a traditional electronic packet-switched network alongside an emulated optical switching fabric. This testbed does not use real optical components, emulating the optical switch by constantly reconfiguring a traditional electronic packet switch to deliver packets only between pairs of “currently connected” nodes and preventing communication during simulated reconfiguration times, as would happen with a real optical switch. Simple testing shows that this provides reasonably realistic behavior in terms of restricting communication the same way a real optical switch would, but does avoid many problems that only come to light when using real optical components – for example, properly handling link-down events seen by network interface cards while a real optical switch reconfigures connections. Note that this design decision also provides the benefit of bidirectional connections without much additional effort. Bidirectional communication can be achieved relatively easily when using most types of switches, especially wavelength-agnostic ones like MEMS, by using circulators or multiplexing each data direction onto a

different wavelength, but it does not come so easily on other switching technologies, such as the WSS-based switches discussed in the next section. The application traffic tested is all TCP traffic and relies heavily on very short round trip delay between nodes to facilitate TCP ramp-up, meaning that low-latency bidirectional communication between any pair of nodes connected by an optical link is required. ACK packets must get back to the sender right away, and cannot be allowed to wait in a queue because the return connection is not currently available or are awaiting a decision on whether they should go through the optical or electronic network.

As with any hybrid network, C-Through must measure the traffic demand for each possible optical switch configuration and provision optical links accordingly. Unlike Helios, this demand estimation is achieved at the host. The size of TCP per-flow buffers used by the Linux kernel is increased from its default value around 128KB to a value on the order of 300MB. A user-space management application periodically queries buffer utilization for each flow using the *netstat* command and adds together the numbers for all flows communicating with the same destination rack. The total utilization of buffers within rack A whose destination hosts are in rack B is a measure of the demand for a direct optical connection between racks A and B. In general, if this number gets high enough relative to the demand between all other possible connections, then the connection will be established until some other connection obtains a high-enough demand metric to replace it. Note that while an optical connection is active, that connection's demand metric is being quickly depleted because the queues generating the demand are being emptied through a high-bandwidth direct connection.

Mathematically, Wang et. al. formulate this as a maximum-weight perfect matching problem [12]. The cross-rack traffic demand matrix is viewed as a graph, where each top-of-rack switch is represented by a vertex, and each optical link is represented by an edge. The weight of any vertex is the total demand metric for the hypothetical optical connection it represents. The goal of the optical configuration manager is to find the mapping such that each vertex is connected to one other vertex by a single edge and the total weight of all edges utilized is the maximum possible. That is, the configuration pushed to the switch should have the maximum aggregate demand matrix of all possible configurations at the time step being considered. C-Through uses Edmonds' [11] algorithm to solve this problem, observing that it runs in polynomial time and can recompute the schedule for a thousand racks in several hundred milliseconds. While for optical links lasting hundreds of milliseconds, this is adequate, faster switching requires a much faster way of computing desired optical configurations, but the system would be more forgiving of sub-optimal configurations.

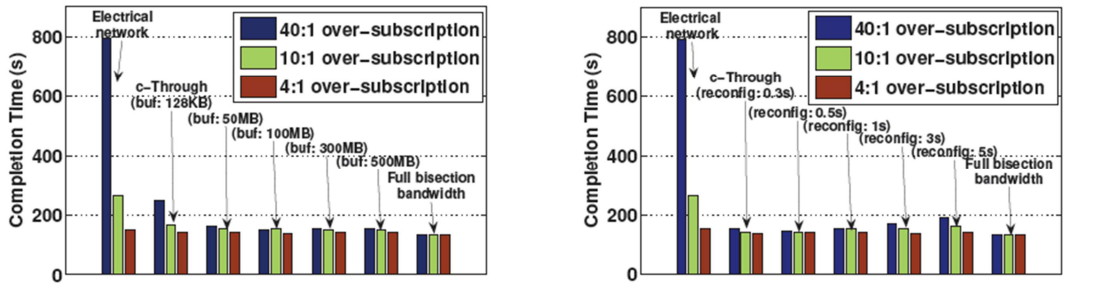
One practical problem that C-Through runs into is the need to multiplex traffic between optical and electronic networks. How does the system decide which packets should go to the optical switch and which go to the electronic one? What if some packets associated with a particular connection travel through the optical network, but later the connection becomes unavailable? Traditionally, different network interface devices connected to the same server would have different IP addresses, or at the very least, different MAC addresses, making it difficult to switch interfaces once a connection has been established. C-Through solves this problem by having only one physical

network interface at each host, but splitting traffic into two virtual interfaces, each of which tags the traffic for use with a different VLAN – either the optical “VLAN-c” or the electronic “VLAN-e”. The ToR switch is able to properly route traffic based on the VLAN tag, and the higher layers of the network stack need not be concerned about which path a packet traverses. The management application on each server keeps track of the optical path availability information it receives from the central configuration controller, and configures its own host's network stack to multiplex packets between the two VLANs as appropriate: optical if possible, and electronic if necessary.

C-Through puts significant effort into making sure the traffic patterns used for testing closely match those in real data centers. This is very important to C-Through because the primary goal of the project is proving that hybrid EPS/OCS network architectures are feasible in today's data centers with minimal to no modifications to hardware and applications, and would deliver significant performance gains or cost savings not on synthetically generated ideal-case traffic, but on real workloads and applications running in modern data centers. The experiments performed track and analyze the performance of three real applications: virtual machine migration, *MapReduce*, and the MPI Fast Fourier Transform algorithm. These applications were chosen because they are commonly performed workloads with varying degrees of suitability for optical switching. Virtual machine transfers are the ideal workload for an optically switched network because they involve long duration, bursty, highly concentrated, high-bandwidth, one-to-one data transfers. *MapReduce* represents distributed computing frameworks – one of the most commonly used classes of

applications in modern data centers. The traffic patterns generated by distributed computing algorithms exhibit periods of bursty one-to-many and many-to-one traffic [7, 8], creating 'hostspots' of relatively latency-insensitive traffic which could potentially benefit from the addition of a high-bandwidth optical switch to the network. The third application, MPI FFT, was chosen as an adversarial case because it lacks centralization [12].

The virtual machine migration test simulates a scenario in which all of the virtual machines in a single rack are to be migrated to different racks, as would typically happen when an administrator needs to shut down an entire rack for maintenance. As expected, the performance of virtual machine migration was drastically improved by the addition of an optical network when the electronic network was oversubscribed (as is usually the case in data centers). In fact, an electronic network with a 40:1 oversubscription ratio augmented by an optical switch achieved performance comparable to that of a network with full bisection bandwidth without the added cost of complexity associated with full bisection bandwidth. This means that augmenting a



(a) Varying socket buffer size (reconfigure: 1s) (b) Varying reconfigure interval (buffer size: 100MB)

Figure 3: Test results of Hadoop sort on C-Through, copied from [Wan10].

data center network with C-Through would allow for much higher oversubscription ratios in the electronic network without significant loss of performance during times of high demand, significantly reducing the cost of network hardware.

While workloads involving virtual machine migration are run in the data center only occasionally, parallel computing frameworks like *MapReduce* take up most of the working life of many data centers of big data giants like Amazon and Google.

The test results of C-Through running such workloads show much promise. In the case of 40:1 oversubscription, a purely-electronic network was crippled by a Hadoop Sort (Hadoop is an implementation of *MapReduce*). When this network was augmented with C-Through, it showed results that outperformed purely-electronic networks with significantly lower oversubscription, as shown in Figure 3.

The MPI FFT algorithm looks like a difficult case for optical switching because it requires tight synchronization and a significant amount of all-to-all communication. To compute the FFT of a large matrix, a central node divides the original matrix into sub-matrices and transmits each sub-matrix to a worker node. At run time, the algorithm involves many exchanges of intermediate results between all nodes – a pattern seemingly not suited for the slow switching times of an optical network. However, much to their surprise, the C-Through team noted significant improvements to the performance of the FFT computation.

As predicted, the reconfiguration time of the optical switch had significant effect on performance. When the switch was reconfigured every 0.3 seconds, the addition of the optical switch improved performance significantly. With longer reconfiguration

intervals, performance was still improved significantly, but not nearly as much as with short reconfiguration intervals. This is logical, because a shorter configuration interval more readily facilitates a fast system-wide synchronization which necessarily involves

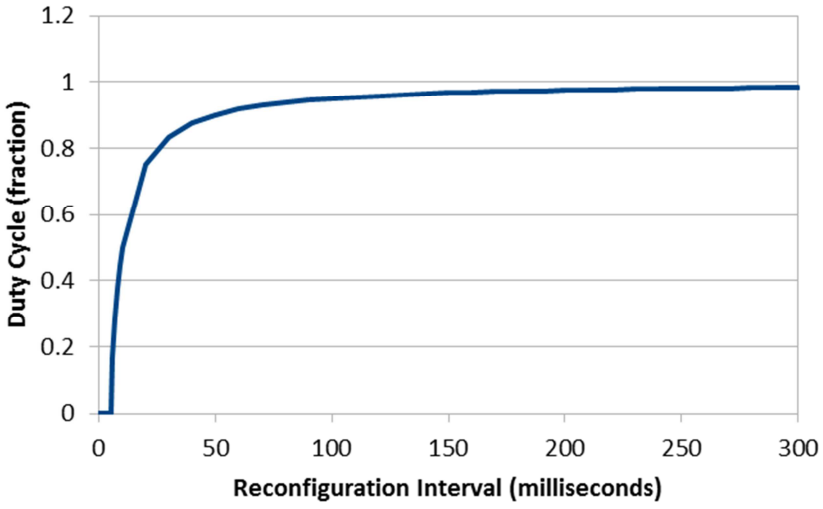


Figure 4: Diminishing returns of decreasing t_{stable}.

all-to-all traffic.

The tradeoff that must be made when decreasing the reconfiguration time is the duty cycle of the network. This is the percentage of total time that the switch is stable and can be used for data transmission. This is expressed specifically in [9] as

$$D = \frac{t_{stable}}{t_{setup}+t_{stable}}, \tag{5}$$

where t_{setup} is the amount of time it takes to reconfigure the switch and t_{stable} is the amount of time the switch remains stable before being reconfigured again.

Figure 4 shows the duty cycle of an optical network as a function of the reconfiguration interval $T = t_{setup} + t_{stable}$. Note that this relationship is a hyperbolic curve with the horizontal asymptote at unity (100% duty cycle). After T gets to be about

ten times bigger than the switching time, the returns from further increasing t_{stable} diminish significantly. This coupled with the fact that performance of tightly synchronized applications like FFT calculation benefits significantly from shorter reconfiguration intervals, leads to the conclusion that reconfiguration intervals should in general be kept as short as possible, given adequate duty cycle. It is not surprising that better granularity is good for application performance – recall that the ultimate goal in this research endeavor is completely optical switching at packet granularity, and the theme of most current research in this field is proving feasibility while balancing tradeoffs to achieve optimal performance.

In hybrid network research, one must always be aware of cases where it is tempting or even necessary to "compare apples to oranges". The C-Through test results are a good example of such a scenario. The conclusions drawn from the previously discussed performance analysis are valid and no doubt promising, but there is an alternative way of looking at the test results that one must keep in mind. The addition of an optical network in parallel with an existing electronic network will, except perhaps under certain adversarial workloads, improve performance because no matter how well or how poorly the optical network is utilized, it still adds to the total available bandwidth of the network. Consider the native bit rate of the purely electronic network, designated as R_{eps}, as well as the bit rate of an optical links, designated R_{ocs}, in the trivial case of C-Through with an infinite reconfiguration time. This is equivalent to a purely electronic network in which the bit rate of all links except one is R_{eps}, and one link has a bit rate of $R_{eps} + R_{ocs}$. Given that the fast link is not completely unused, it is

not necessary to run any experiments to see that performance will improve when compared to the original electronic network. Perhaps this would have been a better base case for comparison with non-trivial C-Through. Another example would be an N-node hybrid network in which the optical links simply traverse all nodes in a round-robin fashion. It can be seen intuitively that such a network would outperform an unaugmented electronic network with link rates equal to R_{eps} for each link. Given a workload of latency-agnostic all-to-all traffic, such a network would have performance equivalent to a purely electronic network with all links operating at $R_{eps} + R_{obs}$. If the workload is such that each node is only interested in communicating with one other node, the equivalent link rate would be $R_{eps} + \frac{R_{obs}}{N}$. To reiterate, this simply means that the exact magnitude of the improvement is to some extent open to interpretation, but it does not cast doubt on the fact that performance does improve. In general, a C-Through network with 40:1 EPS oversubscription still significantly outperforms a pure EPS network with only 10:1 oversubscription under most workloads.

2.3.3. Mordia

The Mordia system [9], created at the University of California: San Diego, is an improvement on previous work in the fast optical switching field. The most important contribution of Mordia is the use of switching times in microseconds rather than tens of milliseconds as seen in previous work. The work argues that the assumption that each optical circuit must exist for close to a hundred milliseconds in order to justify several milliseconds of switch setup time is not necessary and proposes a new network built around an optical switch that takes only around 10 microseconds per reconfiguration.

The key component of the hardware testbed used is a novel optical switch that uses a ring of wavelength selective switches along with several other components to achieve switching times in the microseconds. The current system supports a total of twenty four wavelengths for communication in a unidirectional ring topology. Each node on the ring can inject a single wavelength. The wavelength that any given node injects into the ring cannot change – it is determined by the laser wavelength of the SFP+ interface module of that particular node. Each node on the ring injects a different wavelength using a passive add/drop multiplexer. The same multiplexer is also used to filter out (drop) the wavelength injected by the current node in order to prevent the same signal from traveling more than one lap around the ring and interfering with subsequent transmissions. On the ingress side, each node has a passive optical coupler with one input port and two output ports. This splitter takes in all of the optical signal power traversing the ring and splits it into two fibers. One fiber continues the ring, while the other fiber goes into a wavelength selective switch. Each node can configure its wavelength selective switch to receive any one wavelength from the ring. Technically, this topology can support unicast, multicast, broadcast, or loopback signal transmission, but the implementation of the Mordia system is interested only in unicast transmissions (single transmitter and single receiver). The authors describe this system as a "broadcast and select network in which each fixed wavelength signal from each transmitter port can be routed to any receiver port," [13] noting that each of the SFP+ transceivers utilized in such a system must be able to receive any wavelength used for transmission. Additionally, since each coupler drops a percentage of total power from the ring to the

corresponding node's WSS, there must be amplifiers along the ring to keep up the signal power. In the practical implementation constructed by the group at UCSD, the network is actually split into six stations, each serving four nodes instead of having a completely separate set of equipment for each node. Each set of four nodes shares a four-channel WSS, a four-wavelength (bandpass) add/drop multiplexer, a coupler, and an amplifier, along with some variable optical attenuators used for fine tuning signal power for each node. There are two aspects of measuring the switching performance of such a system. First, the switching time was measured using a test signal and an oscilloscope. It was found that switch reconfiguration actually begins about 3 μs after the trigger signal is sent to the switch. This is followed by about 2.25 μs of reconfiguration time, which is followed by 6-7 μs of ringing. It was noted in [9] that communication may start before the ringing stops completely. Secondly, there is the time that the phase-locked loop in the network interface card takes to lock on to the signal – something that needs to happen before communication can be established. In order to take both parts of the switching time into account, Farrington, et. al. set up a test where the switch was put between a sender PC transmitting sequence-numbered Ethernet frames and three receiver nodes, and the switching time was determined based on the number of dropped frames during each switch event. After the test was conducted over a sample of a million Ethernet frames with 705 switch events, a mean switching time of 11.5 μs with standard deviation of 2.5 μs was obtained.

The other major contribution of the Mordia system is a novel scheduling algorithm called Traffic Matrix Scheduling (TMS). While previous hybrid network

implementations performed “hotspot scheduling” methods in which relatively slow and complex algorithms were used to identify only the pairs of nodes where the extra bandwidth provided by the optical switch was needed most, [13] argues that with microsecond switching, serious attention must be paid to the computation time of the schedule because schedule updates must now happen fast enough to keep up with a 10 μs switching time. Such fast switching also allows the assumption that the majority of traffic can be transmitted over the optical network, not only the highest-demand flows. The key idea is that the scheduling algorithm aims to connect each host to every other host within a set scheduling cycle. For example, in a hypothetical network where the scheduler needs to provide communication between eight nodes, the scheduler will interconnect the nodes to each other in a round-robin fashion, with the amount of time allotted to each configuration being proportional to a measure of total demand for that configuration. The process repeats indefinitely with a fixed period, with the scheduler computing the schedule for the next cycle while the current cycle is being run. The first step of the algorithm is the acquisition of a traffic demand matrix (TDM). This is a matrix that represents the current total traffic demand between each pair of nodes. In practice, this could be the number of bytes buffered for each source-destination pair, or could be obtained from some other source – the algorithm is independent of the specific source of demand estimation data. Once the TDM is obtained, it is scaled into a bandwidth allocation matrix (BAM), which represents the fraction of the total bandwidth 'desired' by each source-destination pair, such that the sum of the values in each row of the matrix is 1. Once the BAM is obtained it is decomposed such that

$$BAM = \sum_i^k c_i P_i, \quad (6)$$

where $c_i$is the fraction of total cycle time allocated to each possible switch assignment matrix (permutation matrix) P_i. Note that the set of matrices $P_i$is pre-determined by the network topology: every node must be connected to every other node at some point in the schedule. There exist well-studied algorithms for achieving this decomposition, which run in polynomial time or better.

The primary argument used by the Mordia team for the requirement of millisecond switching and the infeasibility of TMS without such fast switching times is the queue size required to support it. The equation given is

$$B = R(N-1)T, \quad (7)$$

where B is the queue size required, in bits, N is the total number of ToR nodes, and T is the total slot duration which is also seen in the duty cycle calculation as $t_{setup} + t_{stable}$. Given the typical setup of a network like the previously-discussed C-Through or Helios, $R = 9Gbps; N = 64; T = 100[ms]$, yields $B = 7.1GB$, which is too big to be practical. If T is decreased by a factor of 1000, the queue memory required also decreases proportionally, and 7.1MB queues per port are easy to handle. While convincing, the importance of this argument should not be overstated – modern MEMS systems have switching times on the order of 1 ms, allowing for $T = 20\ ms \rightarrow B = 140MB$, which is not unreasonable. Additionally, this calculation takes into account full link utilization for an indefinite period of time. In such cases, EPS networks run out of queue space and start dropping packets as well. In practice, the general assumption is that networks are rarely used at capacity. TMS does not necessarily require the use of an expensive

custom-built WSS switching system such as the one in the Mordia prototype, but the fastest switches available are strongly preferred.

As of this writing, there has been no published implementation of TMS as described in [13] in an optically-switched network. Farrington et al only provide a general description of the method in order to present a feasible scheduling algorithm for use with a microsecond-switched optical network, but do not test it – the Mordia prototype as of the latest paper about it does not implement any demand estimation, which is the required input for any kind of scheduling algorithm. The traffic used for the published round of testing is synthetic all-to-all traffic, which results in a trivial switching schedule in which each possible switch configuration gets an equal amount of time. The idea itself, however, appears feasible and warrants further research, including simulation and prototype testing.

To prototype the system, the Mordia team used commodity servers to emulate ToR switches. This involved modifying the NIC drivers to ignore link-down events seen by the card in order to keep the system ready to transmit during switch reconfiguration. To achieve the necessary queuing behavior, they modified the operating system's network stack and developed a custom queuing discipline which managed a separate queue for every possible destination ToR switch. Whenever each node received synchronization data indicating which destination node it was connected to, it would drain the corresponding queue, while enqueueing all packets destined for any other nodes. The switch would then reconfigure itself, during which time all packets coming from the TCP/IP stack would be enqueued, until the next connection is established and

synchronized, and the cycle would repeat.

Originally the synchronization of ToRs to the optical switch schedule was to be achieved by moving to the next schedule slot whenever a link-down event is detected by the NIC. Due to limitations imposed by using commodity servers running non-real-time operating systems and the fact that the NIC firmware was closed-source and could not be modified, this could not be implemented. Instead, a separate all-electronic control plane (and therefore separate set of NICs) was used to send synchronization packets from the switch controller to each host. Good synchronization, accurate within 1 μs was generally achieved, but sometimes packets were dropped due to synchronization "misses" caused by the non-real-time nature of the Linux operating system. About 1% of sync messages were received at the wrong time, causing delays and a 0.5% overall drop rate. Another hardware challenge turned out to be the 23 μs delay of the NIC, which forced packet transmission to stop 23 μs before link reconfiguration started. In general, there were many hardware challenges that would have been solved through the use of custom intermediate hardware which could be designed specifically to facilitate OCS.

The Mordia system fared well in a performance evaluation using constant-rate all-to-all synthetic traffic. The goal of the experiment was to see how Mordia compares to traditional EPS in UDP and TCP data rates under ideal conditions. Given a 300 μs per channel schedule slot duration (that is, $t_{stable} = 300[us]$), UDP performance within 4.6% of EPS and TCP performance within 12.1% of EPS was achieved. The law of diminishing returns starts applying to this setup for circuit duration greater than 200-

250 μs, but below that, network performance is significantly reduced – with TCP performance falling by 66% with a slot length of 61 μs. Latency was not mentioned in the paper, but assuming that the network is utilized below full capacity, a latency bound is easy to estimate:

$$L_{max} = T * N. \quad (8)$$

For example, $(300[\mu s/channel])(23[channels]) = 6.9[ms]$ in the unfavorable case of a packet arriving in the queue just after the time slot for its destination has ended. In this case, the latency seems acceptable for most applications. However, latency may pose scalability problems for networks with a large number of ToR nodes and optical switches with high port counts. This becomes another compelling argument for using fast switches, like the novel WSS-based design presented by Mordia, in OCS/hybrid networks. Mordia claims the ability to create a high port count optical switch by stacking rings and gives an example of stacking eight 88-channel rings to obtain a 704 channel switch. However, note what this does to the queuing delay bound for packets: $(300[us/channel])(704[channels]) = 211.2[ms]$, not including any other delays the packet may encounter and with network demand remaining below the link rate. Many data center applications have been shown to work well with comparable or higher latencies [12], but such potential delays warrant the extra effort involved in reducing them, for example, including a low-bandwidth EPS network alongside the OCS network to alleviate this problem for packets that require lower latency, as was done in previous OCS/hybrid network implementations.

3. Proposed System

This thesis work proposes a hybrid electronic/optical network architecture in which unmodified traditional electronic network hardware can be augmented by the insertion of an optical switch with controller hardware and an optical crossconnect interface (OXCI) device at every rack. The OXCI will sit between the ToR switch and the core network, multiplexing traffic between electronic core and the optical core switch, while collecting traffic demand data to allow the optical switch controller to make appropriate switching decisions.

3.1. Motivation

The primary motivation for this work is the fact that simply modifying existing commodity hardware is becoming inadequate as a method of implementing a hybrid data center network. First of all, commercial data centers cannot forgo the use of closed-source hardware and software which cannot be modified in the ways required to make a hybrid network functional. Secondly, hybrid networks require fast switching times for traffic granularity. Recently, major improvements have been made in switching element design. Technology is now reaching a point at which scheduling algorithms and control loops running on general-purpose computing hardware are unable to keep up with switching times and manage the optical switch efficiently. Third, as the basic concept has been proven, it is getting harder for researchers to try new algorithms and network topologies in an environment where all readily available components are closed-source and all but impossible to modify. A separate piece of hardware to manage the interaction between the optical and electronic networks is the next logical step in

this field of research. Initial commercialization of this technology can occur with separate hardware for the optical section of the network as described in this work. Once the technology has been proven and industry is willing to use it on a larger scale, switch manufacturers will likely be motivated to integrate already existing OXCI functionality into their new hardware designs.

3.2. Proposed Network Architecture

To overcome the limitations and challenges described above, a new data center network architecture is proposed, incorporating the previously described idea of a high-bandwidth optical core augmenting a heavily-oversubscribed electronic core network. This system adds hardware to the hybrid network that would prevent any kernel and hardware modifications to the servers, electronic switches, and network interface cards,

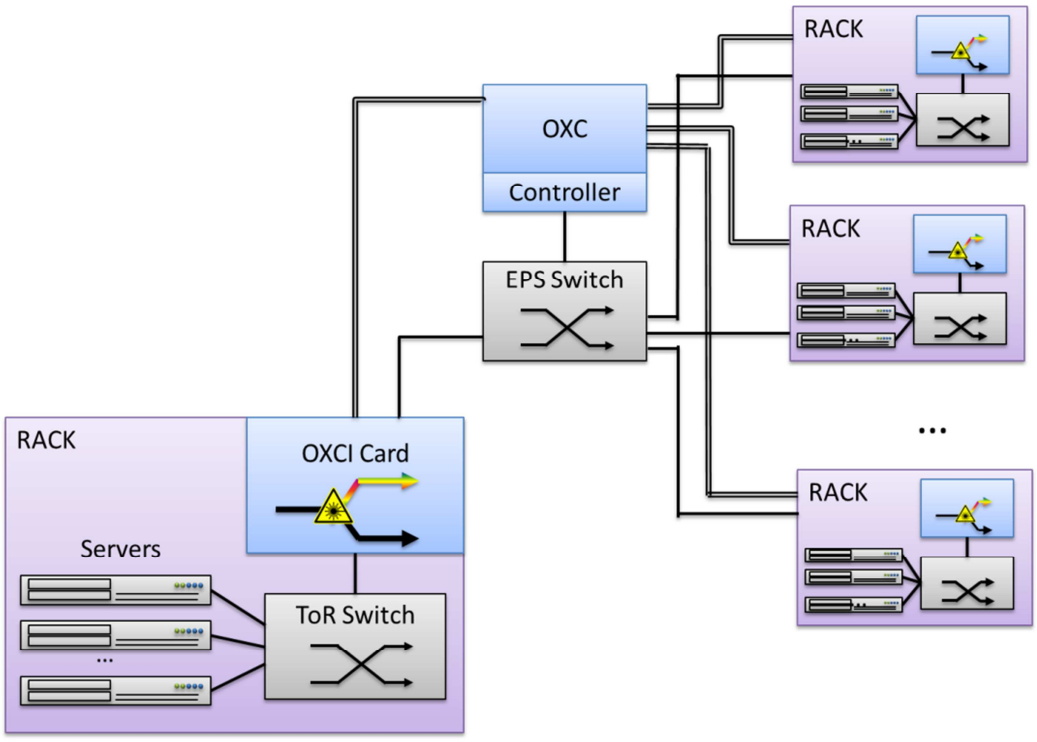


Figure 5: Proposed network architecture.

while still providing necessary data collection, queuing, and scheduling to implement a hybrid network and manage the optical switch.

3.2.1. System Architecture Overview

The proposed network shares significant similarity with both traditional data center networks and previous optical circuit switching proposals. The key differences are the introduction of custom electronic management and support hardware purpose-built for this application and a much cleaner interface between optical and electronic sections of the network, which make the added optical network completely transparent to standard data center components.

The overall architecture of the proposed network is shown in

Figure 5. All of the traditional electronic network components are present within the proposed network with unchanged roles and configuration. Servers and the applications running on them are the primary users of the network, providing the workload. Currently, data center applications tend to be written to take advantage of existing network topologies [5], and future applications can and should be optimized for future network architectures. However, it is desirable that the current generation of applications and servers be able to achieve good performance on a new network architecture without modifications. In view of this, the proposed architecture is designed to require no modifications to the applications or system software running on the servers. In fact, the performance of the proposed network is evaluated with the primary goal of good performance on workloads that model the unmodified behavior of applications currently used in data centers.

Electronic Top-of-Rack (ToR) switches are also kept in their original state within the proposed network. As in traditional networks, they provide communication within the rack and serve as the gateway for communication with destinations that are not on the rack-local network. The only difference is in the way they are connected to the other ToR switches. Instead of connecting directly to an electronic core that facilitates communication between racks in the data center, each top-of-rack switch is connected to an Optical Crossconnect Interface (OXCI), which performs management functions related to interfacing with the optical section of the network.

The Optical Crossconnect interface is the novel aspect of this research. It is a hardware device that sits between the top-of-rack switch and the two core networks. Its primary job is to multiplex traffic between the electronic and optical network, while transmitting and receiving high-bandwidth traffic to and from the high-bandwidth optically switched network. Although this thesis work presents a specific implementation of this device as an example, the key innovation is the addition of a separate device that handles the functions specific to hybrid optical/electronic networks while allowing all pre-existing network hardware and software to operate without any modifications or even knowledge of the hybrid nature of the core data center network.

The electronic network core remains largely the same as in traditional networks. Instead of being connected directly to ToRs, it is now connected to the OXCI devices, which decide which traffic traverses the electronic core. As in traditional networks, it can be composed of either a small number of large switches or many smaller commodity switches arranged in a multi-rooted tree utilizing multipath routing. The

benefit of new hybrid network architectures, including the architecture proposed in this work, is the possibility of significantly increasing the oversubscription factor of the electronic network without losing performance due to the addition of an optical network to relieve congestion of the electronic network by bandwidth-heavy flows. It is desirable to allow the optical network's control plane to monitor congestion within the electronic network in order to efficiently manage the amount of traffic it routes to the electronic network. Fortunately, the facilities for such congestion monitoring already exist within modern electronic switching hardware and can be taken advantage of. One new task to be performed by the electronic core is the routing of control and synchronization packets for the optical network. Existing quality of service (QoS) features of electronic switches can be used to give these most important packets a high priority.

The optical switch is the hardware component that ties the optical part of the core network together, providing direct connections between nodes. There are many architecture and switching technology options for implementing such a device. Many different topologies can be built from different basic elements that vary in switching speed and reliability. While the proposed architecture is technically independent of specific optical switch type, this work assumes sub-millisecond and millisecond switching times, such as the ones achieved in the Mordia testbed [9] or the ones available with the fastest modern MEMS switching elements [14].

The optical switch is managed by a central optical network controller. While OXCI devices can be seen metaphorically as the "nerve centers" of the network, this

central controller is the brain. It is the central entity that collects information from all OXCI devices, making network-level decisions and distributing scheduling and other information to the rest of the network. This device directly controls the optical switch, implementing the decisions it has made.

The proposed network architecture is described solely in the context of IPv4 and Layer-3 switching at the core for simplicity, but can be extended to IPv6 with some additional hardware resource use to handle the longer IP addresses.

3.3. Optical Switch Controller

Most past research has used general-purpose computers for evaluating demand data and calculating the optical switch schedule while using separate FPGA-based hardware to control the switching elements. Since switch controllers from most previous systems already were based on high-performance field-programmable gate array (FPGA) reconfigurable logic hardware that was being underutilized, and schedule computation time is becoming increasingly more important, the proposed system merges management functionality into the optical switch controller. Specifically, the optical switch controller performs the following tasks:

- Gather real time demand data from all OXCI devices.
- Compute the optical switch schedule.
- Send forward scheduling data to all OXCI nodes.
- Interface with optical switch hardware and carry out the switching schedule.
- Act as a clock master for timing synchronization.

3.3.1. Data Collection

The optical switch controller gathers real time demand data from all OXCI devices. To avoid the extra latency associated with polling, all OXCI devices periodically send demand statistics to the optical switch controller. The controller is assigned an IP address on the electronic network. Any packet with the controller's IP address as the destination is assumed to be a control packet.

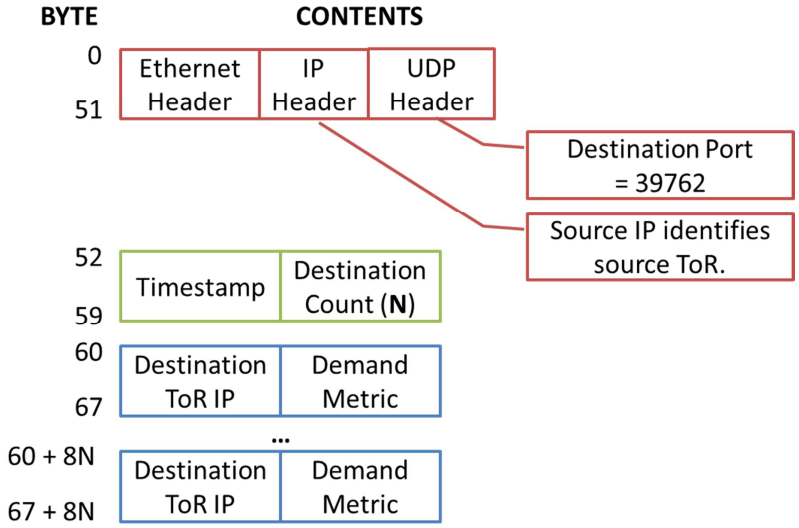


Figure 6: Traffic Demand Report packet format; headers not to scale.

The packet format for Traffic Demand Report (TDR) packets is specified in

Figure 6. The TDR packet can contain traffic demand data for up to 180 destinations, with each listing taking up 8 bytes for a total of 1440 bytes which, with headers, fills the standard 1500-byte Maximum Transmission Unit of Ethernet (normal frame).

To facilitate scalability, ToRs do not report the demand metric for every possible destination. Instead, only the destinations whose demand metrics meet the reporting threshold are reported.

The report for each scheduling cycle is generated immediately after the ToR node receives the upcoming optical switch schedule. The switch controller waits until a reporting deadline has passed and begins to process the demand data it has collected

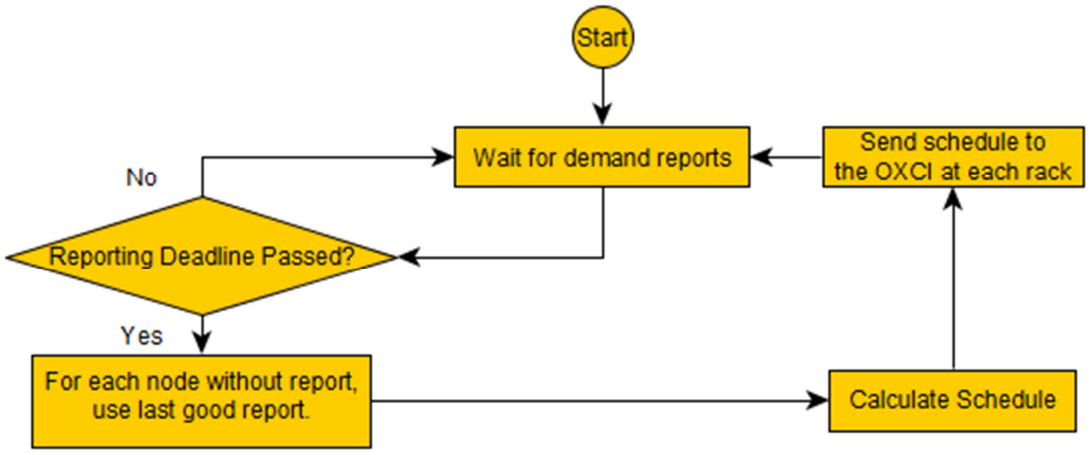


Figure 7: Demand report collection.

from the network. If data from a node does not arrive before the deadline, it is considered dropped, and the demand data from the last report received from the node is used in the current schedule computation. This process is shown in Figure 7.

3.3.2. Optical Switch Schedule Computation

The optical switch controller must use demand data collected from the OXCI devices to compute an optical switch schedule. Unlike Helios and C-Through, the proposed network does not schedule the optical switch to constantly provide bandwidth to the few flows that need it most. Instead, the proposed system follows Mordia's approach of having a schedule computation cycle in which every possible set of connections is established in a cycle. Unlike Morida, which does not retain a parallel electronic core network, the proposed system is able to make up for scheduling

mistakes with the help of the electronic network, so the scheduling algorithm can be simpler, and therefore significantly faster.

One of the schedule computation algorithms that the proposed network can use is Traffic Matrix Scheduling (TMS), the algorithm described in [9] and summarized in Section 2.3.3 of this thesis. Recall from Section 2.3.3 that traffic demand data is aggregated into a traffic demand matrix (TDM), which is scaled to form the bandwidth allocation matrix (BAM) such that its row sums and column sums are equal to 1. The BAM is then decomposed such that

$$BAM = \sum_i^k c_i P_i, \tag{9}$$

where each P_i represents a switch configuration and the constant c_i represents the fraction of the scheduling cycle's time spent in that configuration. As suggested by [9], Sinkhorn's Algorithm [15] can be implemented for matrix scaling and von Neuman's Algorithm [16] for decomposing the bandwidth allocation matrix to obtain the optimal list of c_i, which is the resulting optical schedule, to be calculated every scheduling cycle. Since these algorithms can form the basis of the scheduler, it is most convenient to implement them to utilize application-specific coprocessors in a field programmable gate array or even application specific integrated circuit.

The proposed network takes another step in reducing the complexity of the scheduling algorithm, further reducing computation time. While there can be up to $N!$ connection permutations possible with an NxN optical switch, it is sufficient to select a subset of N of these to connect every pair of nodes. If a constant subset of N permutations is selected to be traversed in N scheduling cycles in a constant order (the

order makes no difference), the only variable that remains is the amount of time allotted to each schedule slot. The amount of time in a fixed-length schedule cycle can be allocated proportionally to the sum of all source nodes' demands for each

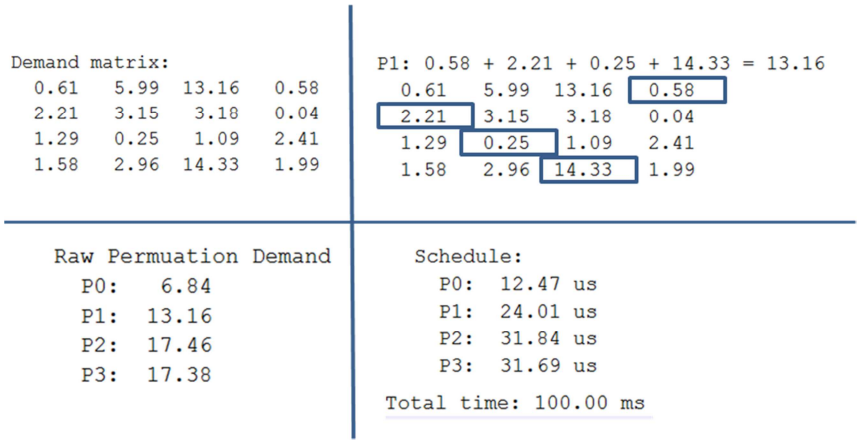


Figure 8: Demand-proportional schedule calculation.

permutation. An example of this process is given in Figure 8. A randomly-generated demand matrix (top left) is given where each row represents a possible source node for an optical circuit, and each column represents a destination. Note the zero-based numbering in this instance. The value of the entry for each source-destination pair is the demand metric for a connection to the destination as reported by the source. The top right frame shows the demand matrix with boxes around entries corresponding to permutation $P1$, where node zero is connected to node one, node one to node two, and so on. The total demand for each switch permutation is calculated in this manner, giving a result in arbitrary units. Finally, the demand numbers for each permutation are normalized to the total amount of available time in the scheduling cycle. This algorithm has linear computation complexity, meaning it can be used when controlling even extremely fast switches.

3.3.3. Schedule Distribution

The optical switch controller must communicate the upcoming schedule to the OXCI nodes attached to each rack. As with demand data collection, for ease of processing, the protocol is kept as simple as possible. Since all OXCI nodes and the optical switch controller are synchronized in time, and schedule computation takes a deterministic amount of time, each OXCI device knows exactly when the switch controller is ready with the schedule.

The switch control node sends the schedule to all nodes as soon as it is available, utilizing a broadcast IP address that includes the core-side interfaces of all OXCI devices on the network. Note that the set of all possible boolean matrices P_i describing optical

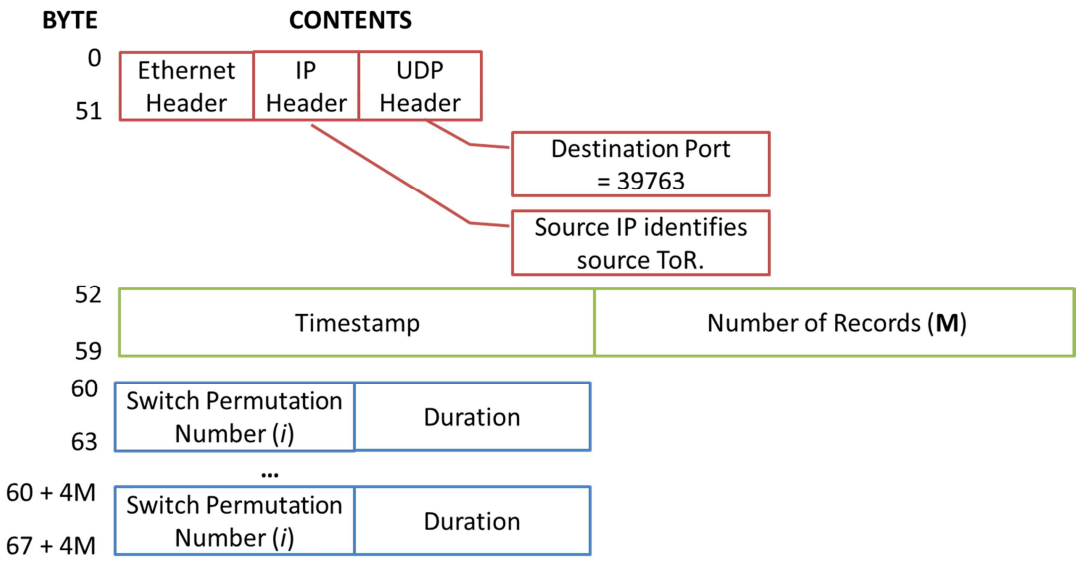


Figure 9: Schedule Notification packet format.

switch configurations is known, so specific matrices need not be transmitted. Thus, the packet format for schedule transmission can be kept small. In the proposed architecture, the schedule notification packet is formatted as described in Figure 9.

This packet contains a list of permutation/duration pairs that compose the schedule. Both of these quantities are 16-bit values. A schedule notification packet that fits within the standard Ethernet MTU can contain up to 360 records. The unit of the specified duration is timer ticks, where the duration of a timer tick, T_f is the fundamental unit of time measure within the network's timing system. The exact value of this parameter is determined by system parameters such as achievable time synchronization accuracy and total scheduling cycle length. If a particular permutation is to last longer than $65535\ T_f$, its permutation number can be repeated more than once in adjacent records.

3.3.4. Optical Switch Schedule Fulfillment

The optical switch controller must interface with optical switch hardware and carry out the switching schedule. The act of carrying out the computed switch schedule must be abstracted away from the specifics of switching technology as much as possible. This is achieved by designing an abstraction layer and switch driver interface into the switch controller that would allow any switching technology and switch topology to be used without affecting any of the higher-level algorithms.

The schedule computed by the switch controller is a list of configurations over time, but does not indicate what signals need to be sent to the optical switch hardware. To actually interface with the switch, the controller needs switch driver hardware that will power the optical switching components and provide a means of transmitting control signals to the raw optical switches, such as GPIO pins or a full featured communication bus like SPI or I2C. The controller also needs knowledge of the

communication and control protocol implemented by the fundamental building blocks of the switch. This can be as simple as the respective meanings of zero and one states on a GPIO line or as complex as a set of configuration registers to be written. The controller may also need knowledge of the specifics of the internal topology of the

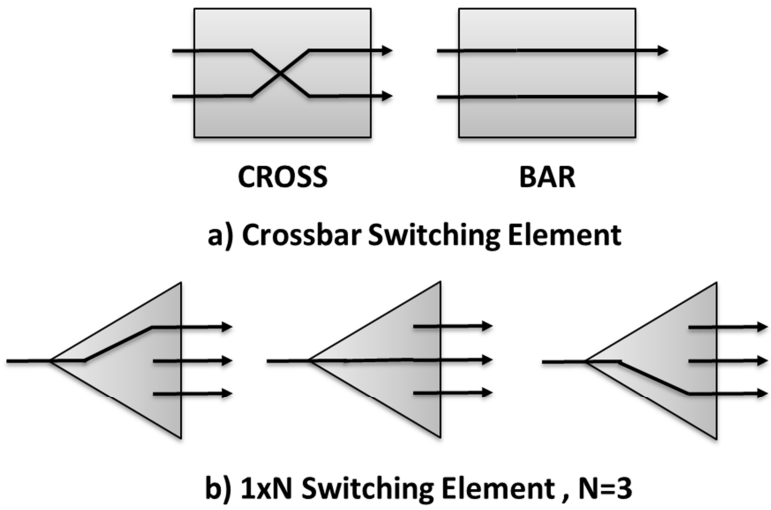


Figure 10: Two types of fundamental switching elements.

optical switch and an algorithm for mapping sets of source-destination pairs to the signals required to achieve the desired configuration. To better grasp the potential complexity of this task, some concrete examples are considered. Figure 10 shows two switching element types commonly used in the construction of larger switches. These and other basic switching elements can be implemented using a variety of technologies [14, 17, 18]. In practice, a large quantity of these basic switching elements can be interconnected to form a switch with a high port count. There are many different topologies for interconnecting such switching elements, some of the most common ones and their characteristics are described in detail in [19].

If one is controlling each switching element directly, the act of achieving a certain input to output port mapping in a non-trivial switching topology, such as a larger version of the Benes network in Figure 11, is far from trivial and requires a specialized

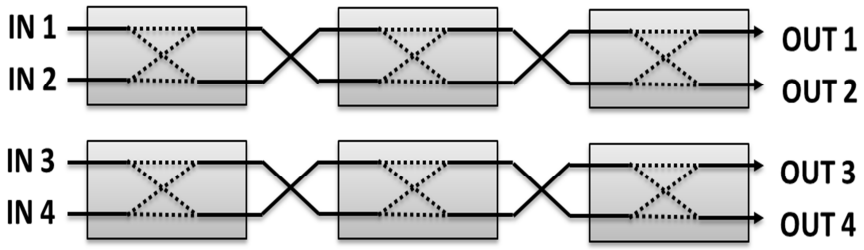


Figure 11: A 4x4 Benes network can be built from 6 crossbar elements.

algorithm. Alternatively, one can use bigger fundamental building blocks, such as the 16x16 integrated optical switch described in [20]. Such high port count integrated modules will probably be the optimal choice when they become commercially available, but interconnecting and controlling them will still not be a trivial task. In light of this

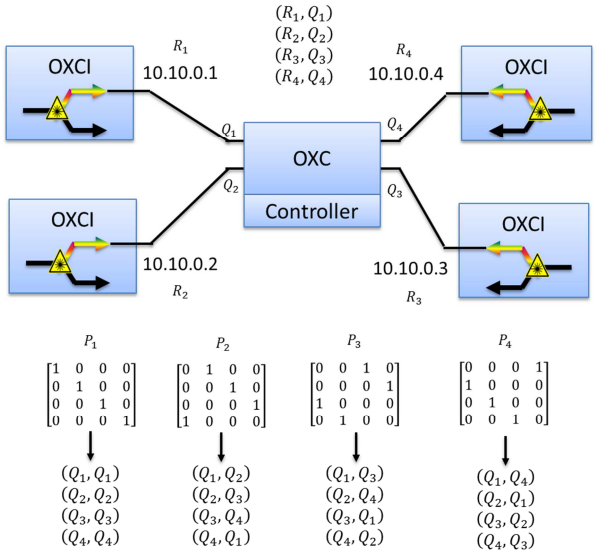


Figure 12: Topology information maintained by the switch controller.

great variety and constant change in optical switching hardware, the network proposed in this paper does not limit itself to any particular type of switching fabric, but provides an interface for the use of interchangeable control hardware and firmware drivers. To describe the implementation of such an interface, begin with the fact that the optical switch controller maintains in memory a list of all possible switch configurations P_i.

Although in discussions of scheduling algorithms, it is easier to think of each P_i as a boolean matrix, it is more practical for them to be stored as ordered source-destination port pairs $\left(Q_i^{src}, Q_i^{dest}\right)$ where Q represents an optical switch port. The optical switch controller also maintains another list of ordered pairs (R_j, Q_j) where R represents a rack-level OXCI node identified by its IP address, and Q represents an optical switch port, identified by its port number. This relationship is illustrated in Figure 12. These lists are populated during system configuration and maintenance, but are not changed during normal operation.

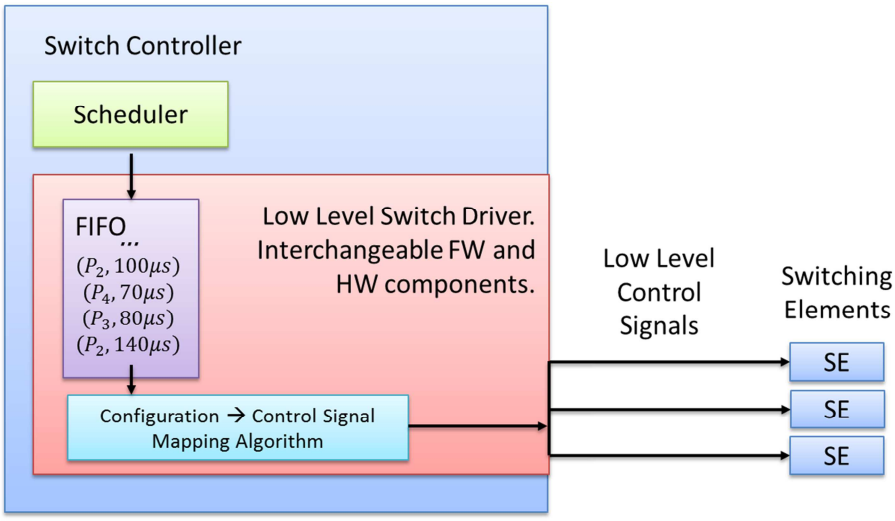


Figure 13: Role of low level switch driver within the optical switch controller.

The set of P_i is defined by the low level switch driver, since it is the only element with knowledge of the switch fabric's physical capabilities. In order to carry out the schedule, the switch controller places configuration/duration pairs (P_i, T_{slice}) into a FIFO maintained by the low-level switch driver, where $P_i$represents the configuration to be executed and T_{slice} is the total time allotted to the schedule slice with configuration P_i, including switch configuration delay. This flow of information is illustrated in Figure 13. Once T_{slice} has elapsed, the low level switch driver moves on to the next entry in the FIFO, or maintains the current configuration if the FIFO is empty. The scheduler may choose to use only a subset of the complete list of configurations. For example, the switch in Figure 12 has loopback capability, represented by P_1. In practice, OXCI devices have no reason to report any demand for loopback traffic since all rack-local traffic would have been already handled by the ToR switches, so the configuration P_1 will not be used during normal operation.

3.3.5. System-Wide Time Synchronization

In order to allow other parts of the network to determine accurately what part of the schedule the switch in currently in, the switch controller must also act as a clock master for timing synchronization between all OXCI devices. This task is key to the operation of the entire system, because any inaccuracy in time synchronization between OXCI nodes and the switch controller can lead to significant and consistent data loss.

Early implementations of hybrid data center networks either switched optical paths too slowly to need precise timing synchronization between nodes [10] or were willing to put up with some timing inconsistency to simplify implementation.

Experiences with Mordia [9] made it clear that any commercial implementation of a hybrid network must be able to keep consistent timing synchronization across the entire network and that this synchronization must be precise to microseconds or better. It is possible to implement timing synchronization using a custom synchronization protocol, but such methods introduce too much cost and complexity for this application, and would alone constitute significant research effort. Fortunately, a technology for achieving the timing synchronization requirements of hybrid data canter networks already exists. Precision Time Protocol, IEEE standard 1588 [21], is a protocol for achieving tight timing synchronization between network-connected devices. Under nearly ideal conditions, it has been reported [22] to achieve accuracy better than 10 ns, and there is research [23] showing that given enterprise-grade "cut-through" switches, such as the ones typically used in data centers, it will achieve sub-millisecond precision even in cases of significant network congestion. PTP support is already available in

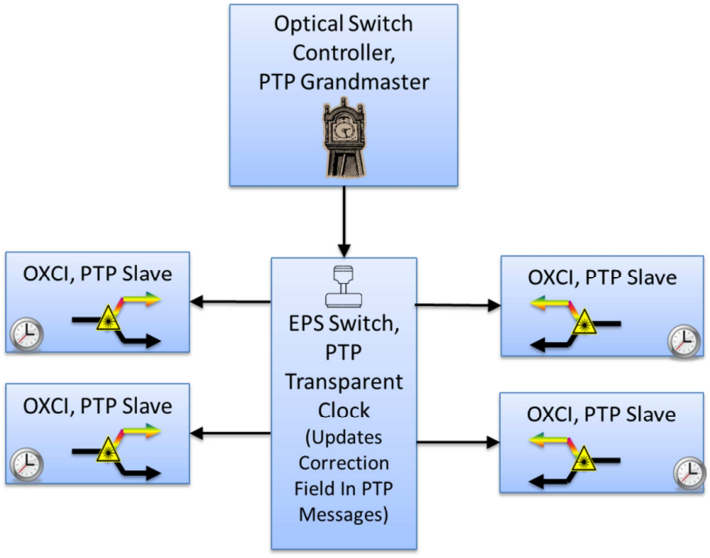


Figure 14: PTP synchronization within the proposed network.

mass-produced switches from manufacturers like Cisco [24] and IBM [25], making it an obvious choice for this application.

The topology of the network from a PTP point of view is shown in

Figure 14. PTP synchronization within the proposed data center network is achieved through the electronic side of the network, with the optical switch controller acting as the “grandmaster clock”, the EPS core switch or switches configured to be transparent clocks, and the OXCI devices acting as slave clocks synchronized to the grandmaster.

3.4. Optical Crossconnect Interface Device (OXCI)

The optical crossconnect interface is placed between the ToR switch and the hybrid network core. It performs the following tasks within the system:

- Queue traffic coming in from the ToR until it can be classified as EPS or OCS.
- Multiplex traffic between the OCS and EPS core networks.
- Manage per-destination queues for the OCS network.
- Compute and report demand metrics for each destination.
- Synchronize timing with optical switch controller.
- Send packets from optical queues, as indicated by the optical switch schedule.
- Facilitate multi-hop OCS delivery.

3.4.1. Ingress Traffic Queuing

The OXCI device must queue traffic coming in from the ToR until it can be

classified as EPS or OCS. This can be achieved using a modest amount of fast memory such as commercially available DDR3. Multiple chips need to be used in parallel to achieve sufficient data rate.

3.4.2. Optical-Electronic Multiplexing

In order for the system to operate efficiently, traffic must be optimally multiplexed between OCS and EPS core networks. In general, slowly-changing flows transmitting large amounts of data should be routed to the optical network, while low-data, latency-sensitive flows should avoid queuing as much as possible and are best left to traverse the electronic network.

This multiplexing is one of the most complex tasks that must be performed within a hybrid network. Ideally, the OXCI device would constantly have knowledge of future traffic patterns and be able to make optimal queuing and multiplexing decisions about every packet before it arrives. In general, such knowledge is impossible to possess but in practice, one can often make a set of safe assumptions about future traffic based on the characteristics of present traffic. For example, it is often the case that if the TCP transmission window continues to grow at the maximum allowed rate for the first several packets, it will continue to grow for some time. There has been research specifically focusing on the prediction of future traffic within a data center environment based on observations of the recent past. For example, the B-Alarm [26] algorithm claims 85% accuracy in predicting traffic bursts within a data center. Another, simpler approach which has already been used in a hybrid data center network [10] is TCP fix point prediction, which predicts the size of a flow's transmit window if it were operating

in an ideal non-oversubscribed environment, an algorithm described in [27]. These algorithms would take significant effort to implement in practice, but would likely give performance more worthy of commercial use. The key observation when considering traffic prediction in the context of multiplexing traffic between the two branches of a hybrid optical/electronic network is simply that flows with constant heavy traffic are better suited for optical transmission, while flows with intermittent traffic are best left traversing the electronic packet-switched network.

The proposed system keeps a list of all destinations potentially reachable through the optical network along with a Circuit Demand Metric (CDM) for each of these destinations. Currently, this is the TCP fixpoint quantity as discussed above, although many other algorithms may be used. The fixpoint algorithm generates a metric that measures a flow's total traffic demand. Only flows that meet a threshold are considered for queuing for the optical network. This reduces fairness because greedier flows are the ones the get the most bandwidth, but it also significantly improves network performance. Although the scheduling algorithm traverses all source-destination combinations and packets can eventually get to any destination using only the optical network, the electronic network is still present and must be utilized optimally. Especially with a fast optical switch, the focus should be on picking the flows that are most likely to benefit from the electronic network.

To optimize the use of the optical network, the OXCI also keeps track of a parameter called the Packet Congestion Metric (PCM). This is an indication of the amount of congestion in the electronic network. In the proposed network, it is the total

throughput handled by the electronic switch.

The decision on whether to send a particular packet to the optical queues or the electronic network is handled at the destination level, not at the flow level. Each destination is periodically classified as either an “optical destination” or “electronic destination”. The decision about whether each destination is an optical or electronic destination is based on whether the CDM of the flow is higher than the CDM threshold, M_{THRESH} , where CDM_{THRESH} is a function of the PCM such that

$$CDM_{THRESH} = A(1 - e^{-m\,PCM}), \tag{10}$$

where A and m are empirically-determined tunable parameters. This relationship between optical-electronic multiplexing and electronic network congestion ensures that the electronic network is never overly congested, even though it can be significantly oversubscribed. This is extremely important because the electronic network carries all control traffic within the hybrid network, and without the proper operation of the electronic network, the optical network cannot be utilized optimally.

3.4.3. Per-Destination Queues

For traffic that has been multiplexed into the optical section of the core network, the OXCI must manage per-destination queues. Possible problems that must be prevented at this point are queue overflows and excessive latency due to a packet waiting in queue longer than one scheduling cycle.

The OXCI maintains an independent queue for each possible destination ToR within the network. At current switching speeds, this is feasible and scales even to relatively large networks. For example, assume the case of a 3000-server cluster

containing 30 servers per rack in 100 racks. With a 20Gbps incoming link rate from the ToR switch to each OXCI, and a 20ms scheduling period (the set of all source-destination combinations scheduled every 20ms), the worst-case optical queue utilization would be $20\ Gbps * 20\ ms/8\frac{\text{bits}}{\text{byte}} = 50MB$ per queue. This could require a total of 5GB of memory for the per-destination optical queues. In a typical use case, the system would not be driven to such extremes, and memory would be dynamically shared for all queues, but the proposed would be able to handle even difficult cases with statically allocated memory.

It is important to ensure that packet latency is kept to a minimum. Packets heading to all destinations that are classified as packet-switched destinations by the multiplexing algorithm are never put into optical queues. Packets remaining in any optical queue that has not been serviced by the optical scheduler are drained into the electronic network as soon as possible. For example, consider the situation where a queue has 10MB of data in it, but has only been allocated time to transmit 8MB during the next optical scheduling cycle. This queue will be partially drained by 2MB into the electronic network to reduce the number of packets that have to wait for more than one scheduling cycle for transmission.

3.4.4. Demand Metric Reporting

The network can be thought of as a control loop, which must have a feedback path for stability and good control. The OXCI provides such feedback in the form of demand metric reports for each destination sent to the optical switch controller. Using this information, the switch controller can calculate the schedule, which is the control

system's output. After each scheduling cycle, the OXCI determines how much optical switch time to request based on the performance of the previous scheduling cycle and how much new data has arrived into the per-destination queues discussed above.

The simplest demand metric usable for the allocation of optical circuits is queue utilization. With this demand estimation scheme, the demand metric for a given destination is simply the number of bytes in the corresponding optical output queue. This approach suffers from several disadvantages. First, it does not take into account the fact that each queue is serviced at a different time. Consider an example where two optical circuit queues are being filled at the same rate and are being completely drained with each scheduling cycle as the electronic/optical multiplexing algorithm attempts to combat latency. The two destinations are serviced at different times, but the demand metric is taken at a single point in time for all destinations. The total throughput is higher for the flow that gets serviced earlier by the scheduler because it consistently has a higher demand metric during the time at which the queue utilization is sampled. To prevent this, the proposed system does not sample the queue utilization directly, but counts the amount of new data that arrived from the ToR switch for the given destination since the last sampling point. This allows the demand metric to account for demand regardless of whether it was met in the current cycle. This is desirable if the assumption that the throughput any flow does not often change rapidly holds true. This is the first term to be used for the calculation of the total demand metric – the input parameterP_{in}.

Although it is undesirable to always force dependence of the total demand

metric on the amount of traffic already serviced or to be serviced during the current cycle, there may be cases where it should be included in the calculation while being able to control the extent of such dependence. Therefore, the concept of already-serviced traffic is re-introduced back into the equation using the parameters$P_{s.eps}$, defined as the amount of throughput already serviced by the EPS network this cycle, and $P_{s.ocs}$, defined as the amount of traffic that can be or has been serviced in the optical time slot allocated to the current flow in the most recent optical switch schedule received. The advantage of ensuring that these have no direct effect on any other terms in the equation is that they can be weighted independently.

While the multiplexing algorithm will try to route any packets remaining in the optical queue to the electronic network, other uses of the electronic network take priority over this queue flushing operation, and it may not always occur, especially when the electronic network is already loaded. When this happens, the OXCI should be able to increase its demand metric in an attempt to prevent any packets from experiencing unusually high latencies. To facilitate this, each packet entering the queue will be time stamped. Each time the demand metric is calculated, the term P_{ql} will serve as an indicator of the age of the oldest packet in the queue. The parameter P_{ql} need not be the packet age itself. In fact, it is preferable for it to generalize to a nonlinear function of the age depending on the emphasis to be placed on latency reduction.

Due to the nature of TCP and its congestion control and avoidance algorithms, the transmission rate of any flow that has been observed to grow in the recent past can often be expected to continue growing until it levels off at a maximum. This behavior

makes the derivative of the input rate parameter very useful in determining the demand metric. The system can make the assumption that if P_{in} has grown in the last cycle, it can be expected to grow at the same rate next cycle. Keeping in mind that the goal of the demand metric is not only to indicate the current demand for an optical circuit at the time it is calculated, but also to predict the demand for the time of the next scheduled optical circuit, the input rate change parameter $P_{\Delta in}$ is included as part of the demand metric calculation.

Finally, the general equation for the demand metric for destination j at OXCI number i is:

$$M_d^{i,j} = W_{in}P_{in} + W_{s,eps}P_{s,eps} + W_{s,ocs}P_{s,ocs} + W_{ql}P_{ql} + W_{\Delta in}P_{\Delta in}. \quad (11)$$

The constants W in this equation represent the weights of the various terms. They are all empirically determined from simulation and experiments, and their optimal values can vary depending on specific work load and network. The complete network can be thought of as a discrete-time control system with the demand metric equation representing one of many transfer functions. The inputs of this control system represent the incoming data rates and other characteristics of the traffic coming in from the ToR switches. The actual optical and electronic network throughputs for each flow can be thought of as the outputs of the system. There are also many different feedback loops. In general, this system is too complex to be modeled analytically, but it can be simulated, prototyped, examined, and most importantly, empirically tuned like any other control system.

3.4.5. Time Synchronization

The OXCI device uses Precision Time Protocol to synchronize to the master clock run by the optical switch controller as described in Section 3.3.5. The time reference allows the OXCI to accurately coordinate the following events with the optical controller and other OXCI devices:

- Transmission of packets into the optical network.
- Sampling of the various components of the demand metric.
- Transmission of the demand metric report to the optical switch controller.

3.4.6. Schedule Fulfillment and Data Transmission

Given adequate time synchronization, the OXCI device knows what state the optical switch is in, and therefore knows which destination it has an optical link to at any time. The OXCI must send packets from optical queues, as indicated by the optical switch schedule: this is the function of the OXCI that all other components are designed to support and optimize.

When the time slot allotted to any given destination comes, the queue for that destination is drained into the optical output port. Significantly, for a certain amount of time, the OXCI is not transmitting any packets at all into the optical network because the switch is in the process of being reconfigured and no connection is actually established. Along with reconfiguration time, there is additional guard time around each period of reconfiguration. The length of this guard time consists of two components: switch reconfiguration uncertainty and time synchronization uncertainty. Many switching

fabric vendors specify their products' switching time with a very significant safety factor. For example, the Helios team discovered [10] that the switching time specified for Glimmerglass is actually much faster than specified, due to the fact that the manufacturer includes a long period of ringing after the switch as part of the switching time, even though the connection is usable for digital communication long before the switch's control loop stops ringing. The trend in hybrid networking research so far has been to overlook manufacturer's switching time specifications, which usually specify switching time for analog signaling purposes, and measure the average time from the breaking of the previous connection to the time the switch stabilizes enough to support a digital communication and call that the "reconfiguration time" of the switch. The actual switching time varies for every switching event and can generally be modeled by a normal distribution for most switches. Due to this, if the OXCI simply waited for the average reconfiguration time between each destination's time slot, it would be guaranteed to lose some packets half of the time. To prevent this, a reconfiguration uncertainty guard time T_{GR} is necessary. This is a set of constants that is experimentally determined for a particular switching fabric. For some optical switching fabrics, this time may be different depending on the specific reconfiguration event (depending on the number of series switching elements being reconfigured, etc.), so this value is best handled by the low-level switch driver, as described in Section 3.3.4. Generally, it should be some multiple of standard deviations of the actual switching time.

The second part of the guard time, namely, the PTP synchronization guard time, T_{GS}, is meant to account for inaccuracy of time synchronization between the optical

switch controller and the OXCI device . This can either be a safe constant given normal conditions, or can be derived from observing the actual quality of the PTP synchronization between the optical switch controller's grandmaster clock and each OXCI device from factors such as jitter observed in PTP synchronization, or an empirical model based on the electronic network congestion as seen in [23].

Finally, the average switch reconfiguration time is referred to as $T_R$and the total silent time between time slots is defined as

$$T_{SILENT} = T_R + T_{GS} + T_{GR} . \quad (12)$$

The proposed system is designed to support, but does not require the dynamic calculation of this time based on specific details of network congestion and optical switch state.

3.4.7. Multi-Hop Delivery

One optional feature of the OXCI is the facilitation of multi-hop delivery. This is an optimization and does not necessarily need to be included in the system, but it is an interesting optimization to examine and may prove useful in practice.

Consider a case of 8 racks with OXCI nodes connected to the hybrid network, designated N1-N8. The schedule is composed of 8 time slots T1-T8 as illustrated in Figure 15. In case a), the system is in time slot T3, where the node N1 is directly connected to N3. Suppose N1 has run out of data to send to N3 but has some time left before the end of the time slot. Also suppose N1 has data to send to N2 but will not reach that state in the scheduling cycle soon because it has just left it. The data can be transmitted from N1 to N3 during time slice T3, and take another hop from N3 to N2 in

SLOT	N1	N2	N3	N4	N5	N6	N7	N8
T1	1	8	7	6	5	4	3	2
T2	2	1	8	7	6	5	4	3
T3	3	2	1	8	7	6	5	4
T4	4	3	2	1	8	7	6	5
T5	5	4	3	2	1	8	7	6
T6	6	5	4	3	2	1	8	7
T7	7	6	5	4	3	2	1	8
T8	8	7	6	5	4	3	2	1

a) N1 can send to N2 via N3

SLOT	N1	N2	N3	N4	N5	N6	N7	N8
T1	1	8	7	6	5	4	3	2
T2	2	1	8	7	6	5	4	3
T3	3	2	1	8	7	6	5	4
T4	4	3	2	1	8	7	6	5
T5	5	4	3	2	1	8	7	6
T6	6	5	4	3	2	1	8	7
T7	7	6	5	4	3	2	1	8
T8	8	7	6	5	4	3	2	1

b) N1 can send to N3 via N5

SLOT	N1	N2	N3	N4	N5	N6	N7	N8
T1	1	2	3	4	5	6	7	8
T2	2	3	4	5	6	7	8	1
T3	3	4	5	6	7	8	1	2
T4	4	5	6	7	8	1	2	3
T5	5	6	7	8	1	2	3	4
T6	6	7	8	1	2	3	4	5
T7	7	8	1	2	3	4	5	6
T8	8	1	2	3	4	5	6	7

c) N1 can send to N6 via N7

Figure 15: Examples of multi-hop transmission in the optical network.

the next time slice T4. This could significantly reduce latency. Cases b) and c) illustrate other possibly useful two-hop routes that save some time. It is possible to enable such an optimization by having the OXCI check the destination of each packet arriving from the optical network.

Although multi-hop delivery in the optical network can save significant latency, it also carries the risk of causing serious problems like out-of-order TCP delivery. The advantages of all such optimizations must weighed against the disadvantages and if they are worth implementing, steps must be taken to mitigate the disadvantages and dangers.

3.5. Performance Monitoring and Evaluation

The primary goal of networking research is to improve the performance of a network. The performance of a data center network can be measured using the following metrics:

- Network throughput
- Packet latency

- Fairness
- Completion time of particular tasks
- Cost efficiency
- Power efficiency

The most often cited quantity in network research is throughput. When looking at low-level components such as transceivers, throughput can be a relatively simple concept, reduced down to bit rate or the user data rate achievable in a single test flow. However, when modeling an entire network, the concept of throughput becomes more complex because there are many flows within the network. For example, many real routers are limited by the number of packets they can process as well as the bit rate of each network interface, so even in a synthetic traffic test, the performance of a real router is often a function of the packet size distribution. When measuring the performance of an entire network at heavy load, one may choose to measure the aggregate network throughput, obtained by summing the throughputs of all flows, or one may choose some single flow of interest and monitor its throughput individually.

In the proposed network, every packet of interest is processed by the OXCI device and source and destination data for each of these packets is read in order to multiplex them between the optical and electronic networks, route them to the proper ports, and collect raw data for demand estimation and scheduling. In this way, throughput statistics are naturally available and easily accessible in a hardware implementation of the proposed system. Since this work concerns itself with simulation of the system, all data is readily available to the simulator and the question becomes

which throughput data should be collected.

Packet latency is the time it takes for a packet to traverse a path from source to destination. This is an important performance metric because many applications require not only that data arrives at a certain rate, but also that data arrives within certain deadlines. There are also applications whose performance depends heavily on how fast traffic can travel between processing nodes. In practice, true one-way latency measurement can be somewhat difficult because it involves time measurements at multiple nodes. It is often measured using synthetic traffic between two time-synchronized nodes or more often, the round trip time of an echo packet, as in the case of ICMP ping or the arrival time of an acknowledgement, as in TCP.

In the case of a hardware implementation of the proposed system, latency will be measured using synthetic traffic. If a more precise result is desired, direct measurement of latency is also possible because OXCI nodes are synchronized using PTP. To perform direct measurement of latency, a sample of packets may be time stamped and hashed when they arrive at the OXCI node, the hashes from multiple nodes compared later, and latencies calculated from timestamps of packets with the same hashes. In the investigation done in this work, the network is simulated, and the simulator has full access to time stamps of all events, so the latency of every packet can be calculated. The latency characteristics of packets within the system at different configurations are used as a key metric in analyzing the performance of the proposed system.

Fairness of scheduling is another metric often used to evaluate the performance

of computer networks. One of the most popular canonical formulas for quantitatively measuring the fairness of some resource allocation is Jain's Fairness Index [28]:

$$f_A(x_1, x_2, \dots, x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2}, \tag{13}$$

where the total amount of some resource is allocated among n clients such that x_i is the amount allocated to the i^{th}client. For example, one can calculate the fairness index f_A of the allocation of bandwidth within a network link among n flows sharing the link. Note that when the allocation is equal for all flows, $f_A = 1$ indicates perfect fairness.

In the proposed system, as in other hybrid networks, it is not given that fairness is necessary or even desired. In fact, the system is designed to give more bandwidth to flows with higher bandwidth demand. Even so, fairness can be evaluated in the analysis of the proposed system and parameters to be tuned if an improvement in fairness is desired can be identified. Alternative definitions of x_i can be used to make this kind of analysis more meaningful in hybrid optical networks – for example, the fraction of a flow's demand met by the network at some point in time can be used as x_i instead of the raw throughput value.

The ultimate test of any computer system is the job it was designed to perform. Although data center networks are built for general purpose computing and perform a variety of tasks, certain tasks and traffic patterns have been established as useful benchmarks. These include *MapReduce* jobs, large file transfers such as VM migration, and highly parallel scientific computations such as parallel FFT computation. The time taken to complete a certain job serves as a reasonable benchmark of overall network performance, given that the test is set up such that network performance is a

bottleneck in system performance.

The proposed system is evaluated using synthetically generated traffic designed to simulate a real workload. The idea is that traffic generators on several hosts are linked such that a traffic generator in a given node does not continue generating packets until it has received a “prerequisite set” of packets from another node or set of multiple nodes. The specifics of the prerequisite relationships between packets determine the degree of synchronization between nodes to be simulated. Many small sets of prerequisite packets between nodes would simulate a tightly synchronized task such as FFT computation, while relatively large responses with bursts from a “master” node as prerequisites may simulate *MapReduce*.

4. System Simulation

The proposed system is evaluated in simulation. Two stages of simulations are performed. First, individual algorithms are prototyped in Python and tested as part of a custom event-based network simulator. This allows the behavior of the algorithms to be analyzed and tuned on a more theoretical level without the distraction of the details present in a more realistic simulator. Synthetic traffic can be generated specifically for exercising any given algorithm in the system. This data is used to gain a general intuition about the behavior of the system. Finally, the system is ported to NS-3, a powerful and highly realistic network simulator and the network is modeled and evaluated in a way that would more closely predict the behavior of a future hardware prototype.

4.1. Prototyping and Initial Simulation

The simulation of the proposed system can be logically divided into a set of key components:

- Traffic Generation
- Demand Metric Calculation
- Schedule Calculation
- Optical/Electronic Multiplexing
- Optical Switch Control

Each of these logical blocks is prototyped separately and tested with synthetic inputs, allowing for the individual characterization of each block's functionality without the possibility of destabilizing interference from other blocks – a problem that cannot be

solved until every block works as desired at the individual level. The components are organized as a Python library and available for inclusion in a basic full-system simulator written in Python. Later, these are ported to C++ classes usable in NS-3.

4.1.1. Traffic Generation

Besides the standard theoretical traffic generators like uniform and Poisson traffic sources, the proposed system is tested with a custom application designed to resemble the kind of network traffic generated by distributed computing applications. The key idea behind this traffic generator is the fact that the progress of a distributed computing job at any given node can depend on the progress of the tasks assigned to other nodes. For example, each node computing a distributed FFT must wait for all

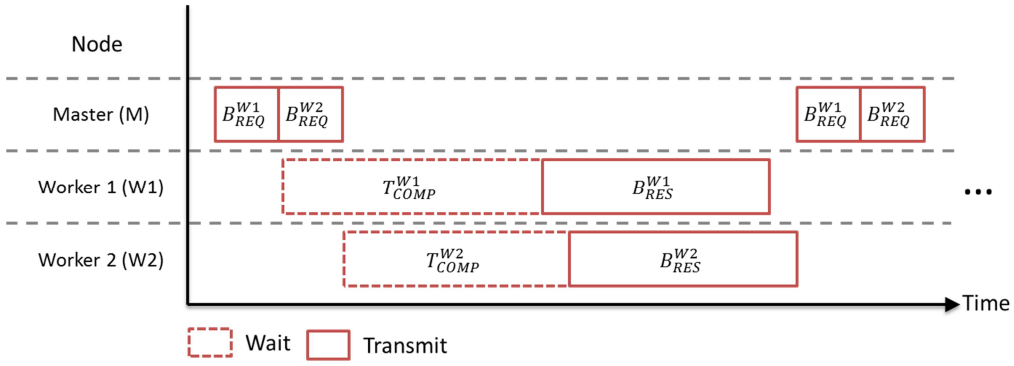


Figure 16: Simulated data center workload.

other nodes to complete their pieces of the calculation before it can move on to the next stage of the job. Many *MapReduce* jobs cannot start the Reduce stage until the total completion of the Map stage. If a job depends on nodes exchanging data very often, it is said to by “tightly synchronized”. If a job only needs a few synchronizations over a long time, it is said to be “loosely synchronized”.

The simulated job is cyclical in nature, with a “request, compute, response” cycle

repeating several times over the course of the job, as shown in Figure 16.The simulated job is divided into N synchronization cycles. Each cycle is defined by the following set of parameters:

- Request length in bytes, B_{REQ}
- Computation time in seconds, T_{COMP}
- Response length in bytes, B_{RES}

Each of these values is an exponentially-distributed random variable, defined in [29] by the probability density function

$$f_x(x) = \begin{cases} \lambda\, e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}, \tag{14}$$

where λ is referred to as the *rate parameter*. For the purposes of this investigation, base values $B_{REQ,MIN}$, $T_{COMP,MIN}$, and $B_{RES,MIN}$ are also used to assure appropriate minimum values. Note that the addition of a base value shifts the mean of the random variable by the base value. For example, given that the mean of an exponential variable is $\frac{1}{\lambda}$ and its variance is $\frac{1}{\lambda^2}$, the computation time random variable T_{COMP} would have a mean of $T_{COMP,MIN} + \frac{1}{\lambda}$ and the variance would remain $\frac{1}{\lambda^2}$. New random variables are generated for each node and each synchronization cycle within the job. Thus, a simulated job is parameterized as follows:

- Number of cycles, N,
- Request length parameters, λ_{REQ} and $B_{REQ,MIN}$,
- Computation time parameters, λ_{CALC} and $T_{CALC,MIN}$,
- Response length parameters, λ_{RES} and $B_{RES,MIN}$.

A tightly synchronized job can be simulated by setting these parameters such that N is a large number, the minimum values are relatively small, and λ is large. Conversely, a loosely synchronized job can be modeled using small values for λ with large $B_{RES,MIN}$. The total duration of the job is determined by the length of each cycle and the total number of cycles in the job.

The progress of a simulated job is tracked in Figure 17. Each line represents a separate run of the simulation with a different scheduling period. The job running on the network with a 100 ms scheduling period completes quickly, while the network with

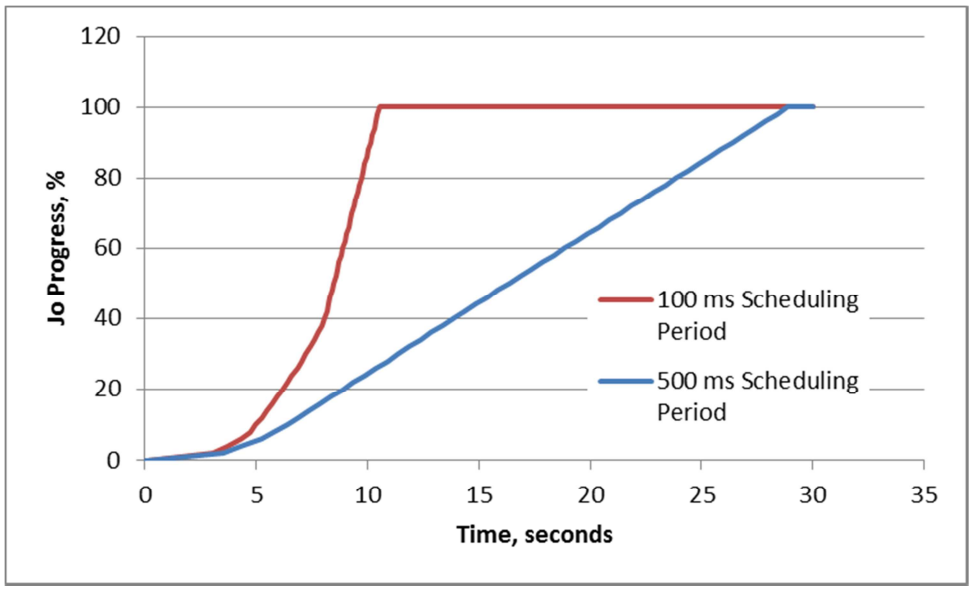


Figure 17: Progress of simulated job with different link rates.

a 500 ms scheduling period backlogs the job because multiple nodes must synchronize with each other by exchanging intermediate results. This is the kind of behavior observed in real data centers as well as other hybrid network research projects such as C-through [12], so it can be concluded that the simulated workload is realistic.

4.1.2. Optical Switch Control

While the proposed network architecture does not force any specific switch topology, technology, or architecture, it is important to test the network with a realistic simulated switch. This section explores one of the more complex cases of using a

Switch	Number of 2x2 Elements	Longest Path	Shortest Path
Benes	$N(2\log_2 N - 1)/2$	$2\log_2 N - 1$	$2\log_2 N - 1$
Batcher-Banyan	$N(3+\log_2 N)(\log_2 N)/4$	$(3+\log_2 N)(\log_2 N)/2$	$(3+\log_2 N)(\log_2 N)/2$
Cantor	$\log_2 N\left(N\log_2 N - \frac{N}{2}\right)$ $+2N(\log_2 N - 1)$	$4\log_2 N - 1$	$4\log_2 N - 1$
Clos(1)	$4\sqrt{2}\,N^{1.5}$	$5\sqrt{2N} - 3$	3
Crossbar	N^2	$2N - 1$	1
Spanke	$2N(N-1)$	$2\log_2 N$	$2\log_2 N$

Figure 18: Vital parameters of various switch types.

MEMS-based Benes network to construct a switch with and the details involved in simulating it with some accuracy. The most important question for the simulator to answer is "given the probability density function of a single switching element's switching time, how long does a switch reconfiguration take in reality?"

Figure 18 shows the characteristics of several switch topologies of an $N \times N$ as reported in [19]. Assuming a network of 32 racks, a 32x32, $N = 32$ optical switch is required. This can be constructed using $\frac{N(2\log_2 N-1)}{2} = \frac{32(2\log_2 32-1)}{2} = 144$ switching elements. One of the advantages of a Benes network is that the number of switching elements traversed in the switch does not vary with configuration. The number of switching elements the signal traverses is always $2\log_2 N - 1 = 2\log_2 32 - 1 = 9$ switching elements.

In order for the signal to be able to travel through the switch, all switching elements must be stabilized. Conversely, data loss will occur if one or more elements is not finished switching by the time data transmission starts. Since the switch is a mechanical device, it is not guaranteed to stabilize at any particular time, but the switching time can be modeled by a probability distribution. While at the time of this writing detailed probability distributions of MEMS switching times are not readily available, the team responsible for the WSS based switch in [13] observed a normal distribution or reconfiguration times. It is reasonable to assume that MEMS switching time can also be modelled by a normal distribution, but the method described here applies regardless of the specific probability distribution. The probability of successful transmission in the optical switch can be calculated using basic statistics.

The random variable $\boldsymbol{T_{SW,i}}$ is defined as the time after the reconfiguration signal is sent that the i^{th} switching element in the chain takes to stabilize in a single switching event. In this example, all switching elements within the switch follow an identical switching time probability distribution. The probability of a successful transmission in a one-element switch is this given by

$$P_{s,i} = P\{\boldsymbol{T_{SW,i}} \leq T_{SILENT}\}, \tag{15}$$

where T_{SILENT} is the time between switch reconfiguration signal transmission and beginning of data transmission into the switch, as defined in chapter 3.2. This is easily extended for an M switching elements in the path being reconfigured by taking the intersection of success probabilities for every switch, expressed as

$$P_{s,total} = P_{s,1} \cap P_{s,i} \cap \ldots \cap P_{s,M}. \quad (16)$$

There is no mechanism by which the value of a switching element's $T_{s,i}$ can affect the value of $T_{s,i}$ for another element, so the *random variables are independent*, which means that

$$P_{s,total} = \prod_{i=1}^{M} P_{s,i}. \quad (17)$$

Since it is assumed that the switching time for each individual switching element has the same probability density function, the *success events are all equally likely* and

$$P_{s,1} = P_{s,i} = \cdots = P_{s,M} = P_s\ . \quad (18)$$

The total probability of success is expressed simply as

$$P_{s,total} = (P_s)^M = (P\{\boldsymbol{T_{SW}} \leq T_{SILENT}\})^M. \quad (19)$$

In the 32x32 Benes network example, M can be assumed to always have the worst-case value of 9, or if further optimization is desired, a table of M-values can be created for every possible configuration change. The probability P_s is found by simply evaluating the probability distribution function of $\boldsymbol{T_{SW}}$ at T_{SILENT}. Note that if the total guard time T_{SILENT} is selected such that a single switching element has a 99% chance of being reconfigured on time, the success rate if 9 elements need to be reconfigured is $(0.99)^9 = 0.914$, leaving the switch with a nearly 9% chance of data loss for every connection in every reconfiguration event.

4.2. Detailed Simulation with NS-3

To obtain realistic results, the proposed network is modeled and simulated in NS-3, a powerful network simulation tool that can provide detailed data about a theoretical network, including simulating the behavior of TCP/IP. The advantage of using

NS-3 is the level of detail and flexibility it is able to provide. Most importantly, this simulator models the behavior of the TCP/IP stack and ARP with good accuracy. This is a complex feature critical to the evaluation of the proposed network architecture because the majority of data center applications depend on TCP/IP for reliable delivery, flow control, and congestion avoidance.

4.2.1. Architecture of the NS-3 Simulator

NS-3 is an event-based simulator that uses object-oriented C++ to model many aspects of computer networking in detail. To understand the advantages, goals, and limitations of the simulation used to evaluate the proposed system, it is helpful to have a general idea of the way NS-3 itself operates. This section presents a description of NS-3 that is pertinent to simulating the network proposed in this thesis, but is simplified for the ease of explanation. Those wishing to gain an accurate and general understanding of NS-3 are referred to the NS-3 manual [30].

NS-3 makes heavy use of the features of C++; especially object oriented design and generic programming (templates). Devices and algorithms in the simulator are represented by C++ objects, and specialization is handled using inheritance – for example, an “echo server application” inherits common attributes from the class “application”. At the highest level a simulated network in NS-3 is composed of node objects connected by communication channel objects.

A communication channel in NS-3 is meant to represent a physical communication medium, such as a point-to-point copper or fiber optic link, or a wireless communication channel. As a general rule, communication channels are characterized

by their inability to do any logical processing on the data they are handling. That is, they cannot make decisions based on the data traversing them. However, they can delay data, and possibly even introduce errors into the data, while transporting it from source to destination.

Ethernet networks in NS-3 are often represented by carrier-sense multiple access (CSMA) channels which model an environment in which only one device can transmit at any time. This makes the CMSA channel a fitting model for Ethernet switches, since modern Ethernet switches can receive data coming from multiple ports at the same time and queue it, later sequentially sending it out of a single port to a single destination to avoid data loss due to collision. In an NS-3 simulation, the buffering is actually done at the node until the CSMA channel becomes available, but this behavior is in many ways indistinguishable from buffering in an Ethernet switch. For example, a traditional Ethernet network with a switch at the center and hosts connected in a star topology can be reasonably represented in simulation using a CSMA communication channel object. CSMA communication channels have data rates and delay attributes which represent the corresponding real quantities: the data rate is the rate at which transmission can take place, while the link delay is the amount of time data takes to traverse the channel in a manner analogous to traversing a real electromagnetic transmission line such as copper wire or optical cable. Note that a CSMA communication channel itself makes no routing or queuing decisions like a real Ethernet switch, but it can reproduce behavior similar enough to a real Ethernet switch in most cases, and as it happens, it can model a ToR switch accurately enough for the

purposes of simulating the proposed network architecture.

Another important communication channel in NS-3 is the point-to-point link. This is a fundamental theoretical communication link that allows full-duplex communication between two network entities. This is analogous to simply connecting two nodes with a cable. This communication channel is also governed by transmission rate and link delay

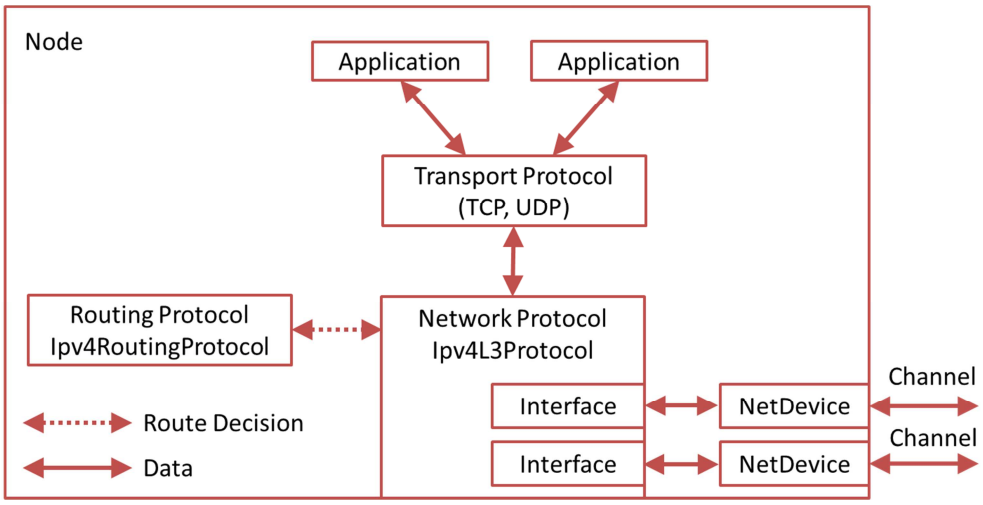


Figure 19: The structure of a node in NS-3.

attributes, but can only be used to connect two nodes.

The Node object in NS-3 represents any entity that can process data traversing the network, modify it, and make decisions based on the contents of packets and frames. Nodes can serve as the sources and final destinations of data in the network as well as intermediate decision makers that route data through the network. Thus a node can represent a server on the network running applications, a network-layer router making decisions about the path data takes in the network, or a custom data-handling device like the proposed OXCI.

Figure 19 shows the composition of a node object in NS-3 and the relationships

between its components. Any node can act as a router, much the same way that a computer with multiple network interfaces can route packets between networks. Each node is connected to the other nodes in the network by one or more communication channels.

The interface to the communication channel is a NetDevice matching that communication channel. For example, CSMA and point-to-point channels use CSMANetDevice and PointToPointNetDevice objects, respectively. A NetDevice object in NS-3 corresponds to a network interface card (NIC) or router port in physical hardware. NetDevice objects take care of link-layer considerations such as physical (MAC) address checking, scheduling packet arrivals on the other end of the channel based on the link rate, delay, and packet size, and queuing data until the channel required for transmission becomes available.

A node contains a network-layer protocol object, in the case of the simulations performed in this work, this happens to be a model of Internet Protocol Version 4 (IPV4). This object handles routing using function calls to a routing protocol object (Ipv4RoutingProtocol) and multiplexing between transport protocols in the case of local delivery. The transport protocol models, in this case TCP and UDP, handle communication with applications installed on the node.

An application is a piece of software that generates and processes traffic. Applications useful in simulation can be simple statistics-based traffic generators, which generate traffic according to some statistical model or models of real application-level protocols such as FTP. The packets traversing the simulated network have real payloads

and can therefore be used to make decisions, even decisions about management of the network itself.

4.2.2. NS-3 Model Implementation

In modeling the proposed network, a topology must be developed and generated in NS-3. The topology of the hybrid section of the proposed system can be described as a star with two centers, or “bipyramid”, because it has two cores – an optical core composed of a fast optical circuit switch and optical switch controller, and an electronic core composed of a traditional electronic switch. Each simulated rack contains a top-of-rack switch connected to the core network via an OXCI device and some number of servers, usually between 20 and 60 in practice, fewer in simulations.

The topology to be generated is shown in Figure 20, where the “rack” nodes include the ToR switch and all hosts in the rack. The simulator is written to take the number of racks in the network and the number of hosts per rack as parameters and automatically generate a topology with these characteristics. The simulated nodes must have information to use for routing packets through the network. Normally, NS-3 can automatically populate routing tables in each node, but the topology of the proposed network is constantly changing and the fact that the network has an optical core switch in parallel with the standard electronic core router makes the standard automatic routing table management scheme ineffective. Instead, the simulated network uses only static routing, set up during topology creation and modified as needed on optical switch reconfiguration events.

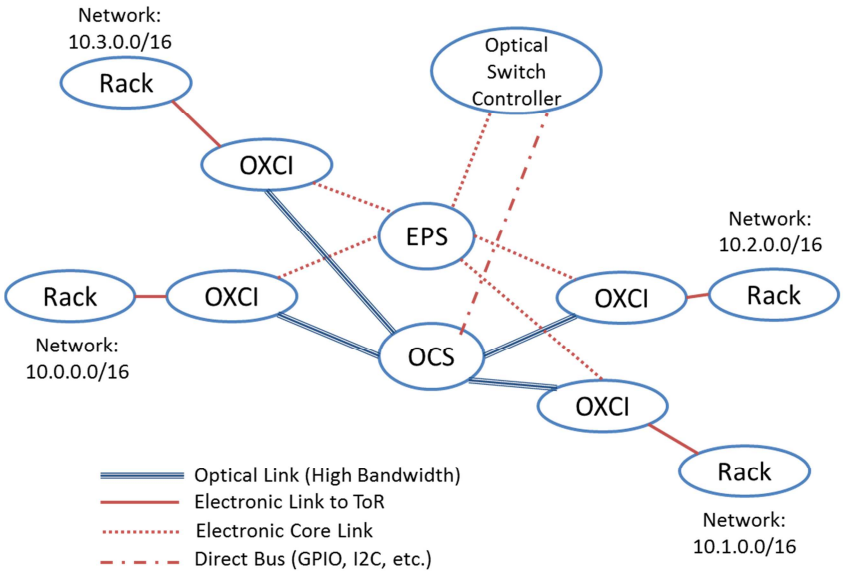


Figure 20: Network Topology with four racks.

In order to model this system, custom modules must be added to the core of NS-3. First, the optical switch is modeled as a *CommunicationChannel* object, and optical transceivers are modeled as *NetDevices*. The functions of the OXCI and optical switch controller are modelled as Applications running on NS-3 Node objects.

The *OpticalSwitchChannel* object is loosely based on the model for a point-to-point link, but instead of connecting two devices to each other, it acts as an *N* x *N* optical switch. Up to *N* network devices can be connected to the switch, such that the transmitter of the node is connected to one end of the switch, and the receiver is connected to the other end.

Figure 21 shows the logical model for the connections between the optical switch and four OXCI nodes. The arrows indicate direction of data travel. In the simulations, the switch can be configured in four different permutations, designated P_n possible for this switch. Permutation P_0 is a loopback connection, so the transmitter of

each node is connected to its own receiver, P_1 indicates the connection scheme $N1 \rightarrow N2; N2 \rightarrow N3; N3 \rightarrow N4; N4 \rightarrow N1$, and so on. In general, if N ports are labeled Q_0 through Q_N, and the transmitter from port Q_S is connected to the receiver at port Q_D, the destination port number D for any source port number S while the switch is configured in permutation number X is expressed by

$$D = (S + P) mod(N). \tag{20}$$

In the NS-3 implementation, the *OpticalSwitchChannel* upon receiving a packet checks the source port the packet is coming from, and based on the current permutation number X, settable through a function call, figures out the destination port number D which serves as the array index of a pointer to the NetDevice connected to port D, and forwards the packet to that *NetDevice*.

Technically, reconfiguring the switch is as simple as setting the variable representing the current permutation number, but the switch must also simulate the time it takes for the components within a real switch to be physically reconfigured. After

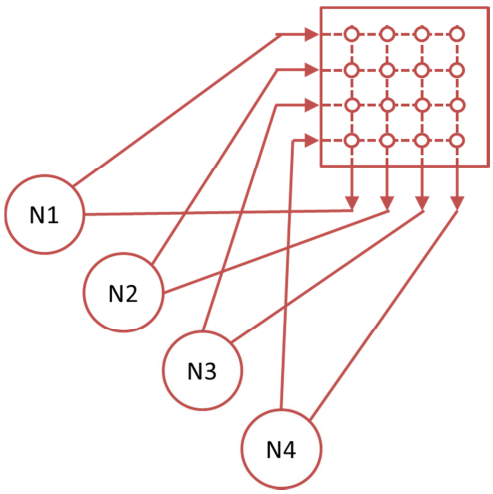


Figure 21: Four OXCI nodes connected to a 4x4 optical switch.

the reconfiguration function is called, the destination port number is set, but all packets attempting to traverse the switch are dropped until the simulated reconfiguration time elapses. The generation of this time interval was described in detail in Section 4.1.2. To summarize, a number of instances of the random variable $\boldsymbol{T_{SW}}$ are generated, and the biggest of these is used as the total switching time.

Another custom component in the simulator is the *OpticalSwitchNetDevice*. This component simulates not only the optical transceiver, but a large part of the OXCI functionality – namely, queuing and demand monitoring. This simulation object houses a queue for every OXCI destination reachable through the optical switch and interfaces directly with the OXCI application running on the node to multiplex traffic between electronic and optical networks, report data to the application, and drain the appropriate per-destination queues based on the known current configuration of the optical switch.

Along with low-level modifications, there are three custom Applications simulating vital network functions. The *OXCIApplication* object in NS-3 models the logic and functionality associated with most of the OXCI functionality described in Section 3.4. An instance of this application runs on every OXCI node.

The switch controller is also modeled by an application running on a dedicated node connected to the electronic network. This application performs switch control tasks as described in Section 3.3. It maintains a pointer to the *OpticalSwitchChannel* object and reconfigures it as needed through direct function calls.

4.2.3. Experimental Setup

There is no limit on the data rates or the number of hosts being simulated, and the simulator is written to be as general as possible. However, the computing resources available for simulations are not unlimited, and there are tradeoffs to be made between the number of simulated computers and simulated data rates and the amount of data points that can be taken. The experiments conducted in this work focus on exploring the qualitative effects of parameter changes rather than specific magnitudes of metrics in highly specific situations. One must keep in mind that the goal is to characterize the behavior of the system rather than predict a detailed outcome in a specific situation. Since the experimental results come from an event-based numerical simulator, increasing all data rates in the system simply amounts to multiplying intermediate quantities by a constant, and in most cases, obtaining the same qualitative behavior.

The experimental setup consists of 12 simulated servers connected to four racks. Each node has a traffic generator application running on it. This traffic generator is designed to simulate the network traffic generated by a typical distributed computing workload within a data center. The traffic pattern is characterized by many-to-one and one-to-many behavior with most nodes not changing their communication partners often, as described in Section 4.1.1. Recall the network topology from Figure 20. The link rates vary throughout the network. The optical simulated link rates are set to 100Mbps, while the link rate (and therefore oversubscription ratio) of the electronic core is varied from extremely low (500kbps) to on par with the optical link rate (100Mbps). The nodes within each rack are interconnected using a simulated Carrier Sense Multiple Access

channel which models the ToR switch. The link rate from the ToR switch to the OXCI devices is also 100Mbps to provide the ability to test the system under heavy traffic through the core. In a real system, it is straightforward to aggregate traffic from many servers into a single high-bandwidth link, but it is much more difficult to create a high-port-count and high-bandwidth core. Due to this, the core EPS switch in the simulated network is simulated with low link rates. The ability of a simpler hybrid core to replace a high-performance electronic core is tested.

4.3. Simulation Results and Analysis

The most important goal of the proposed system is to allow a complex, high-performance electronic core network to be replaced by a low-performance electronic core supplemented by a managed optical circuit switch without significantly reducing the performance of the network. The ability of the proposed system to do this effectively can be measured by the completion time of a simulated distributed computing workload. This job starts with one-to-many "request" transfers from several servers assigned to be job master nodes to other servers assigned to be workers. Once a worker has received all of the request data it waits for some time to simulate processing and it begins sending response data. The master waits until it has finished receiving responses from all workers and begins another request/response cycle. In the simulations performed, there are three masters communicating with nine worker nodes arranged throughout the network. The workload completion time is measured from first packet sent to the time all three masters complete their full workloads. Each workload consists of 50 cycles. Requests are set to be around 8 kilobytes of data, while compute

wait times are around 100 milliseconds and responses are about 120 kilobytes of data. The specific values for each cycle are random in order to facilitate a more realistic

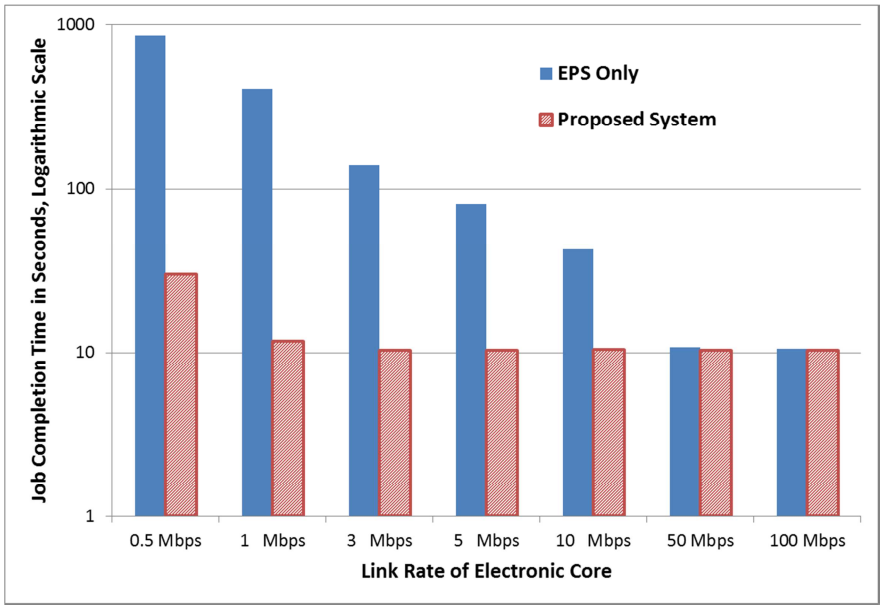


Figure 22: Time to complete simulated job for various electronic core link rates, a comparison of plain EPS to the hybrid network.

simulation. The random number generator can be seeded with a constant value, so results are reproducible. The job completion times for various EPS network link rates with and without the optical components added by the proposed system are shown in Figure 22. The scheduling cycle period of the system is kept a constant 50 ms for this experiment. The proposed system still relies on the electronic core for control traffic and transmission of TCP acknowlegements; an extremely low-speed electronic network will adversely affect its performance. The hybrid network significantly outperforms a pure EPS network with even extremely low EPS link rates, and nearly matches the performance of pure EPS once network bandwidth is sufficient to no longer bottleneck the simulated workload.

One important concern for hybrid networks is the possibility of transmission into the optical switch starting before it has been fully reconfigured. The optical switch in the simulations is modeled to realistically reflect the random behavior of a real switch, as described in Section 4.1.2. Experiments are performed to determine the effect of the relationship between the guard time and the averagre switching time on data loss and job completion time. The switching time can use any statistical model that fits the specific phyical properties of the switch being modeled. In this particular simulation, the

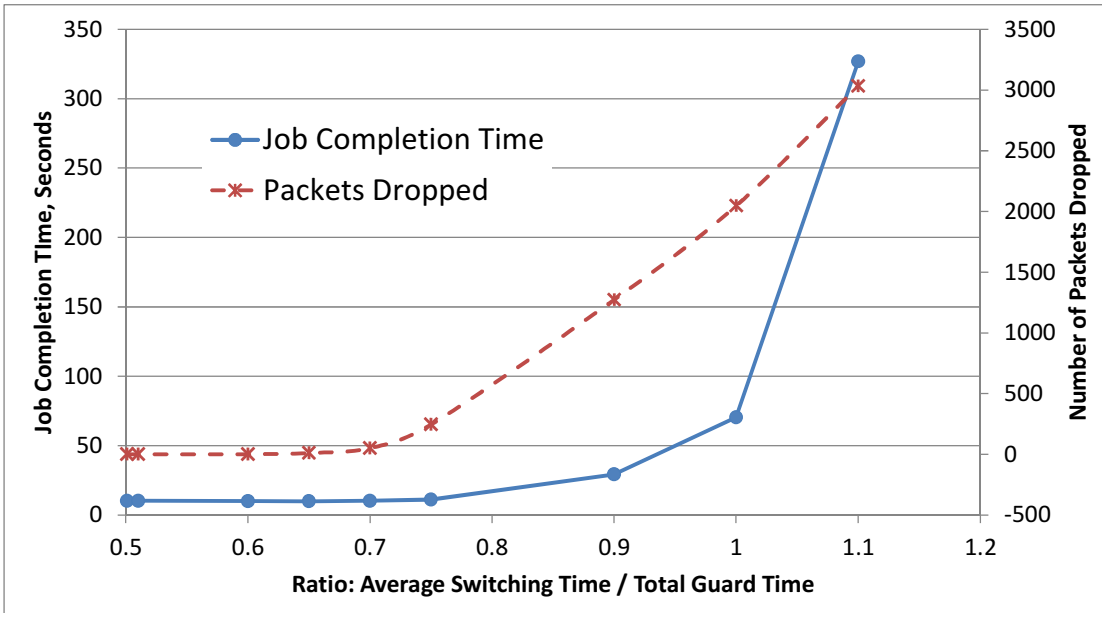


Figure 23: Effect of switching to guard tome ratio on system performance.

switch is modeled as having a switching time that is composed of a constant base and with an exponentially distributed additional time added to it. In this case, the average switching time turns out to be $T_{SW.B} + 1/\lambda$ where $T_{SW.B}$ is the base switching time, and λ is the rate parameter of the exponential distribution. The variance of the switching time is simply the variance of the exponential random variable used, $\frac{1}{\lambda^2}$. In the

experiment, the EPS link rate is set to 5 Mbps while the other link rates are kept as 100 Mbps. The schedule period is set to 10 ms, while the guard time is kept constant at 125 µs. Note that the guard time referred to here is the total time from switch reconfiguration command transmission to beginning of data transmission by the OXCI into the optical switch. The base switching time is set to one half of the guard time. The independent variable in the experiment is the ratio between the average switching time and the guard time. The effect of varying the average switching time, which in this case also changes the variance of the switching time, is shown in Figure 23. The experimental data makes it clear that a hybrid network should never start transmitting after only the average switching time of the switch has passed, since that would cause data loss after more than half of all switching events. The results also show that any packet drops caused by packets being transmitted into the void have a drastic effect workload performance, especially if the workload depends on TCP for reliable data delivery.

Another important parameter in the proposed system as well as other hybrid optical/electronic networks is the scheduling period. The proposed system does not use a hotspot scheduler, but cycles through a predetermined set of switch configurations, varying the time alloted to each permutation according to demand. This is significantly more computationlly efficient than a hotspot scheduler, and provides better data granularity. Of primary interest is the effect of schedule period on the completion time of the simulated workload. An experiment was performed where the optical link rate was kept constant at 5 Mbps, with the other link rates constant at 100 Mbps. The scheduling period of the system was varied and effects of these changes recorded. The

effect of changing the scheduling period on job completion time is shown in Figure 24. The system's duty cycle was kept constant at 95% by varying the switching/guard time such that

$$95\% = \frac{T_{SCHED} - N * T_{GUARD}}{T_{SCHED}} * 100\%, \quad (21)$$

where N is the number of scheduling slots, T_{SCHED} is the schedule period, and T_{GUARD} is the time from transmission of the switch reconfiguration command by the switch controller to the beginning of data transmission into the optical network by the OXCI.

Although the workload is not latency sensitive, it can suffer from a scheduling period that is too long because job progress depends on many parties exchanging

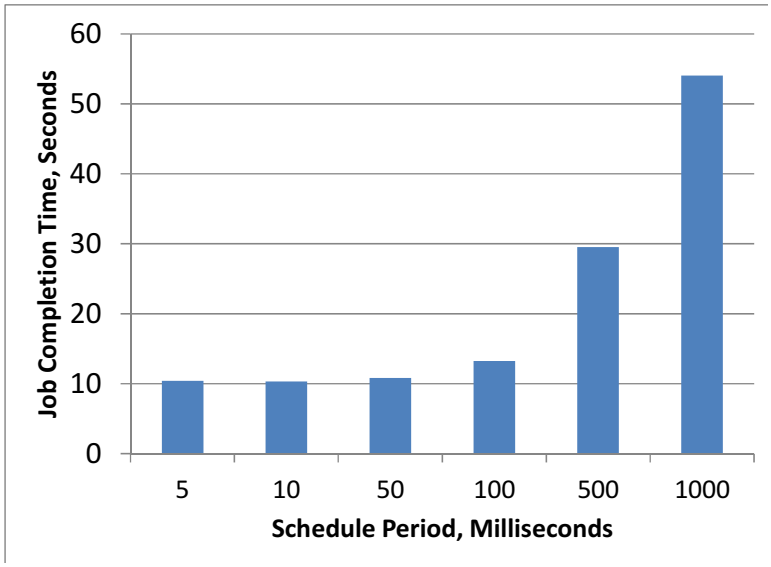


Figure 24: Effect of scheduling period on job completion time.

information relatively quickly. A more tightly synchronized workload, i.e. one where the servers involved exchange data more often, would begin to suffer performance loss with a smaller scheduling period. Given sufficiently fast switches to keep the duty cycle at 95%, no detrimental effect was observed from continuing to decrease the scheduling

period. In practice, it becomes difficult to maintain a high duty cycle while decreasing schedule period once sufficiently fast switches can no longer be found or become prohibitively expensive. However, significant progress is being made in optical switching speed and a 5 ms schedule period for an implementation of the proposed system is not overly optimistic.

Another quantity that should be affected by the scheduling period is the average latency of the traffic. Theoretically, it should be directly proportional to the scheduling

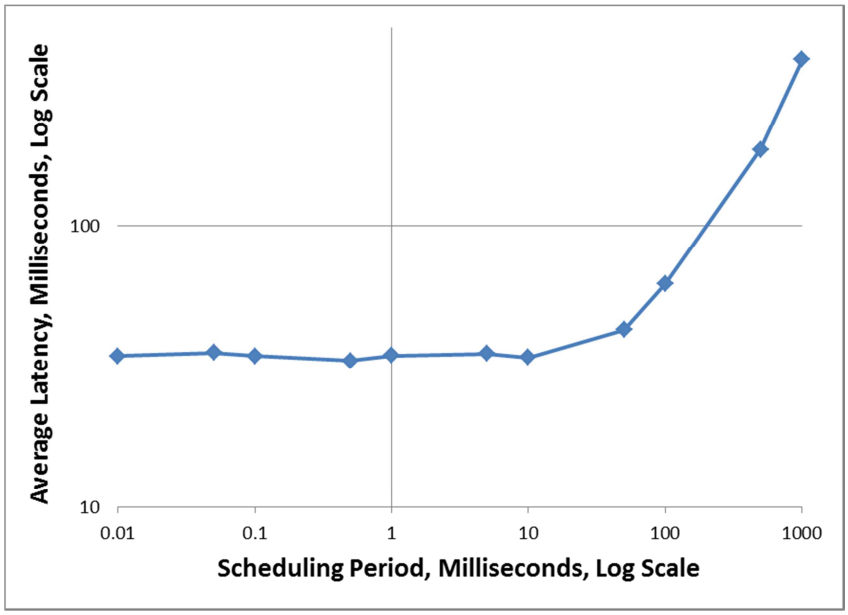


Figure 25: Effect of scheduling period on latency.

period itself. However, Figure 25 shows that for very small scheduling periods the average latency stays around 30 ms. This could be caused by the nature of the workload, time the packet spends waiting in queues before it gets to the optical core, or some other bottleneck in the network. A closer look at packet latency is given in Figure 26, which shows the latencies for about 30000 packets from the simulation with the 50 ms scheduling period, plotted over 1.4 seconds. Each point represents a packet's

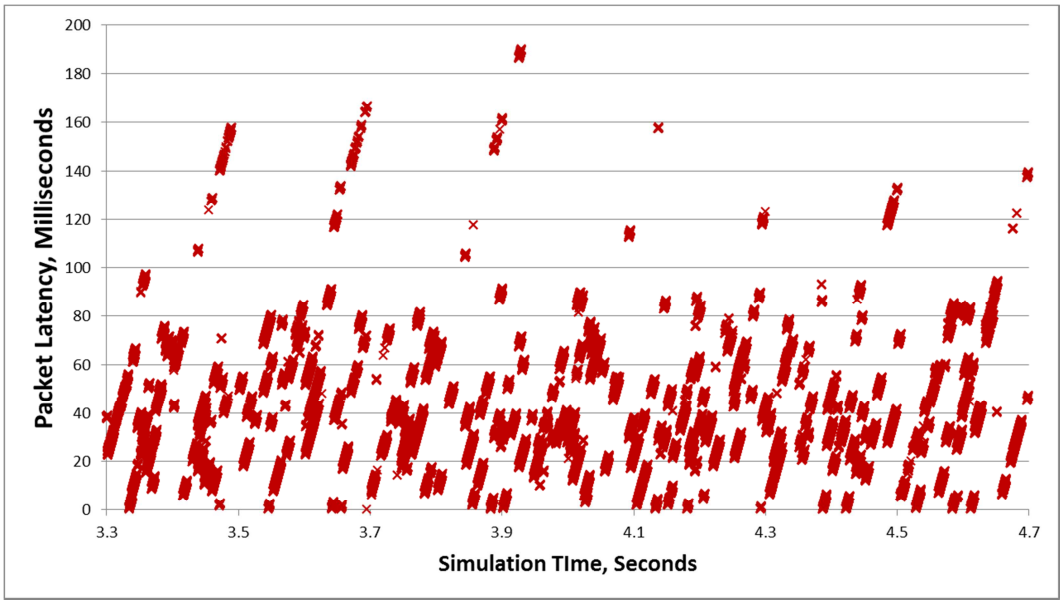


Figure 26: Individual packet latencies with 50ms scheduling period.

latency, and the time indicated on the x-axis is the arrival time of the packet. The straight lines are characteristic of large groups of packets queueing up after each other, then being drained from the queue. Packets generally experience latencies close to 50 ms, as expected, since the the optical network operates in quick bursts spaced 50 ms apart. There are also some outliers with approximately 150 ms latency, which probably waited in OXCI queues for multuiple schedule cycles because bandwidth was not available to send them or because other packets were deemed by the scheduler to have higher priority.

5. Conclusions

The main goal of this thesis is to design a platform that overcomes limitations of using only modified commodity hardware in hybrid optical/electronic data center networks, and is easy for industry to adopt quickly once the subject has been sufficiently researched and developed to warrant enough confidence for commercial adoption. The system exhibits simulation performance improvements similar to those observed in previous hybrid network research. The simulation platform developed can be used to experiment with and further improve algorithms.

In the presence of workloads with relatively bursty network demand and sufficiently loose synchronization relative to switching speed, the optical fabric significantly improves workload completion times and allows for a large increase in oversubscription of the electronic core fabric. Most importantly, this work demonstrates the feasibility of a platform more suitable for future research than the previous methods of using commercial hardware, but modifying it in ways that would be unacceptable in a production environment, such as pretending that high-end servers with modified Linux kernels are ToR switches, modifying NIC drivers, or disabling optimization algorithms in network devices. As expected, system performance in distributed workloads still degrades as the scheduling cycle period increases.

The idea of leaving an electronic core in the network even with very fast optical switches has proven successful, because it allows the network to deliver timing-sensitive packets without introducing an unacceptable amount of latency to packets delivered through the optical network. This is especially important in the case of TCP

acknowledgement packets, which are sent directly to the electronic network in the proposed system to ensure that TCP is not bottlenecked by the lack of constant bidirectional communication in the optical network. The electronic core also provides some redundancy in case of optical switch failure.

The network architecture described in this thesis can be used in a commercial setting. Although the cost benefits of using OXCI devices in a small to medium-sized commercial network would be only marginal due to the cost of high-performance FPGAs currently required to implement them, even a small network would benefit from the ease of expansion offered by this new network architecture, since in traditional networks adding core capacity requires either extremely complex and costly rewiring for multi-rooted trees or similar topologies, or large electronic switches whose cost increases exponentially with port count. In contrast, the cost of an OXCI-based hybrid network increases linearly with rack count, given the availability of suitable switches.

In the future, if an FPGA-based version of the proposed architecture is successful, manufacturers will be able to build ASIC implementations of OXCI devices, drastically reducing costs and making the proposed architecture commercially advantageous in nearly all data center networks. The drive to optimize applications to work with a hybrid core network would further improve performance.

As optical switches become faster, the proposed network architecture can provide for a smooth transition from electronic switching to optical switching in the core, allowing the electronic core to become more oversubscribed until it disappears completely.

References

[1] 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 ACM SIGCOMM HotNets-VIII*, Barcelona, Spain, 2009.

[2] L. A. Barroso, and U. Hölzle. "The Data Center as a Computer: An Introduction to the Design of Warehouse-Scale Machines," *Synthesis Lectures on Computer Architecture,* vol. 4, no. 1, pp. 1-108, 2009.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, Commodity Datacenter Network Architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, Seattle, WA, 2008.

[4] M. Isard, M. Budiu, Y. Yu, A. Birrell, and Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," in *Proceedings of the 2nd ACM Sigops/Eurosys European Conference on Computer Systems*, Lisbon, Portugal, 2007.

[5] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[6] S. Ghemawat, H. Gobioff, and S. T. Leung, "The Google File System", in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, Lake George, NY, 2003.

[7] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The Nature of Data Center Traffic: Measurements & Analysis," in *Proceedings of the 9th ACM*

SIGCOMM Conference on Internet Measurement, Chicago, IL, 2009.

[8] T.Benson, A. Anand, A. Akella, and M. Zhang, “Understanding Data Center Traffic Characteristics,” in *Proceedings of the ACM Workshop on Research on Enterprise Networking*, Barcelona, Spain, 2009.

[9] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating Microsecond Circuit Switching Into the Data Center," in *Proceedings of the ACM SIGCOMM 2013 conference on Data Center Networks*, Hong Kong, China, 2013.

[10] 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 ACM SIGCOMM*, New Delhi, India, 2010.

[11] J. Edmonds, "Paths, Trees and Flowers." *Canadian Journal on Mathematics*, pp. 449–467, 1965.

[12] 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 ACM SIGCOMM*, New Delhi, India, 2010.

[13] N. Farringon, A. Forencich, G. Porter, P. C, Sun, J.E. Ford, Y. Fainman, G.C. Papen, and A. Vahdat, "A Multiport Microsecond Optical Circuit Switch for Data Center Networking," *IEEE Photonics Technology Letters*, vol.25, no.16, pp. 1589-1592, Aug. 2013.

[14] X. Ma and G.S. Kuo, "Optical Switching Technology Comparison: Optical MEMS vs.

Other Technologies," *IEEE Communications Magazine*, vol.41, no.11, pp.S16,S23, Nov. 2003.

[15] U. G. Rothblum, H. Schneider, and M. H. Schneider, "Scaling Matrices to Prescribed Row and Column Maxima," *SIAM Journal of Matrix Analysis Applications*, vol. 15, no. 1, pp. 1-14, Jan. 1994.

[16] 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.

[17] K. Noguchi, "Transparent Optical Crossbar Switch Using Liquid-Crystal Optical Light Modulator Arrays, Integrated Optics and Optical Fibre Communications," *11th International Conference on, and 23rd European Conference on Optical Communications,* Conf. Publ. No.: 448, vol.4, 1997.

[18] A. Bazin, K. Lenglé, M. Gay, P. Monnier, L. Bramerie, R. Braive, G. Beaudoin, I. Sagnes, R. Raj, and F. Raineri, "Ultrafast All-Optical Switching and Error-Free 10 Gbit/s Wavelength Conversion in Hybrid InP-Silicon on Insulator Nanocavities Using Surface Quantum Wells," *AIP Applied Physics Letters*, No. 104, 2014.

[19] N.A. Shalmany and A.G.P. Rahbar, "On the Choice of All-Optical Switches for Optical Networking," in *Proceedings of the HOTNETS 2007 International Symposium on High Capacity Optical Networks and Enabling Technologies*, Atlanta, GA, 2007.

[20] A. Wonfor, H. Wang, R.V. Penty, and I.H. White, "Large Port Count High-Speed Optical Switch Fabric for Use Within Data centers [Invited]," *IEEE Journal of Optical Communications and Networking*, vol. 3, no. 8, pp. A32,A39, Aug. 2011.

[21] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," IEEE Std. No. 1588, 2002.

[22] "IEEE 1588 Precision Time Protocol Time Synchronization Performance," Texas Instruments, Application Note AN-1728, Apr. 2013.

[23] R. Zarick, M. Hagen, and R. Bartos, "Transparent Clocks vs. Enterprise Ethernet Switches," in *Proceedings of the International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication*, Munich, Germany, 2011.

[24] "Cisco Connected Grid Switch Software Configuration Guide, Cisco IOS Release 12.2(58)EY." Cisco, Manual, Jul. 2011.

[25] "IBM Networking OS 7.4 Features Summary." IBM Corp., Technical Document BMD00326, 2012.

[26] H. Wu, J. Taheri, and A. Zomaya, "B-Alarm: An Entropy Based Burst Traffic Prediction Approach for Ethernet Switches in Data Centers," In *Proceedings of the IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, Dalian, China, 2013.

[27] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks." in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, San Jose, CA, 2010.

[28] R. Jain, D.M. Chiu, W. Hawe, "A Quantitative Measure of Fairness and

Discrimination for Resource Allocation in Shared Computer Systems," DEC, Hudson, MA, Research Report TR-301, 1984.

[29] A. Papoulis, S. U. Pillai. “The Axioms of Probability.” in *Probability, Random Variables and Stochastic Processes*, 4th ed. Boston: McGraw Hill, 2003, ch. 2, pp. 15-44.

[30] “NS-3.20 Manual,” Internet: http://www.nsnam.org/ns-3-20/documentation/, Jun.17, 2014 [Jul. 10, 2014].