Switching the Optical Divide: Fundamental Challenges for Hybrid Electrical/Optical Datacenter Networks

Hamid Hajabdolali Bazzaz[◇], Malveeka Tewari[◇], Guohui Wang[⋆], George Porter[◇], T. S. Eugene Ng[⋆], David G. Andersen[†], Michael Kaminsky[‡], Michael A. Kozuch[‡], Amin Vahdat[◇]
[◇]University of California, San Diego, [⋆]Rice University, [†]Carnegie Mellon University, [‡]Intel Labs

Abstract

Recent proposals to build hybrid electrical (packet-switched) and optical (circuit switched) data center interconnects promise to reduce the cost, complexity, and energy requirements of very large data center networks. Supporting realistic traffic patterns, however, exposes a number of unexpected and difficult challenges to actually deploying these systems "in the wild." In this paper, we explore several of these challenges, uncovered during a year of experience using hybrid interconnects. We discuss both the problems that must be addressed to make these interconnects truly useful, and the implications of these challenges on what solutions are likely to be ultimately feasible.

Categories and Subject Descriptors

C.2.1**[Network Architecture and Design]:** Network topology, Packet switching networks, Circuit switching networks

General Terms

Design, Experimentation, Performance

Keywords

Data center networking, optical circuit switching, hybrid network

1 Introduction

Cloud computing is imposing stringent performance requirements on the underlying data center networks: high-bandwidth connectivity across tens to hundreds of thousands of nodes, latency as low as a few microseconds, high reliability, and easy management. Recent work on Helios [8] and c-Through [21] independently proposed two network architectures to meet this challenge. Both create a *hybrid* network that combines the best properties of electrical packet switches and high-bandwidth optical circuit switches, the latter reconfigured at millisecond timescales using MEMS optical switches.

These efforts demonstrated the promise of deploying hybrid networks in commodity Ethernet environments. A number of important cloud applications, including virtual machine migration, large data transfers, and MapReduce, experienced significant performance improvements while running on such a hybrid network with the potential for much lower cost, deployment complexity, and energy consumption than purely packet-switched high bandwidth networks.

Unfortunately, both papers ignored an important property of modern cloud data centers: their fundamental heterogeneity and multi-tenancy. We have since encountered a number of challenges, some quite unexpected, in building these networks and using them for larger-scale, mixed workloads. In this paper, we discuss both the promise of hybrid networks and the importance of addressing the key challenges that can prevent their wider adoption.

Several hardware/firmware engineering challenges arise in incorporating switched optical circuits into the modern data center. Most notably, the link setup and switching time of optical components are far from their theoretical limits—perhaps because they were previously used primarily for slowly switching (tens of seconds or longer) telecom applications.

The software challenges are more complex, with the most thorny arising from the temporal and spatial heterogeneity in data center traffic: flow duration, correlations in demand and interference between applications, priority across flows within and across applications. Addressing these software challenges requires near real-time analysis of application requirements to support dynamic switch scheduling decisions. For scheduling at the application layer, such measurement and analysis should ideally take place on aggressive timescales of milliseconds or tens of milliseconds across networks of tens of thousands of servers.

The original Helios and c-Through designs focused on maximizing total bisection bandwidth, treating flows as interchangeable and undistinguished from each other—effectively ignoring the diversity described above. Although a reasonable starting point, these assumptions often produce circuit scheduling behavior and forwarding decisions that lead to unnecessary circuit flapping and low circuit utilization. This paper revisits several of these assumptions to articulate design goals for practical hybrid data center interconnects; specifically, the interconnect should 1) tolerate greedy and ill-behaved flows that try to occupy circuits but not use them, 2) tolerate inaccurate information about application demands and changes, 3) support flows that are interdependent and correlated, and 4) support flexible circuit sharing policies with flow and application differentiation.

Based on these challenges, our design goals, and our year-long experiences using hybrid networks, we propose a meta-solution to these challenges: that the control framework for a hybrid data center interconnect should allow flexible, fine-grained, and responsive control. The framework collects traffic statistics and network status information from various sources, performs data

SOCC'11, October 27–28, 2011, Cascais, Portugal.

analysis to understand traffic demand with application semantics, and configures the network with user-defined objectives about overall performance and sharing metrics. It should do so without imposing overhead or delays that would prevent reacting on millisecond timescales.

We therefore introduce an *observe-analyze-act* framework for optical circuit configuration. We have begun prototyping an OpenFlow [16] implementation of this framework; a modular topology manager controls the placement of individual flows onto optical circuits or electrical packet switches. While we do not have a final answer to the challenges we present above, our experience using this framework suggests that the fine granularity supported by per-flow placement supports the implementation and testing of advanced scheduling algorithms capable of capturing application and data center demands.

The contributions of this position paper are 1) a taxonomy of unexpected and non-obvious challenges that arise when actually deploying hybrid networks on real hardware and with real traffic, 2) an outline of the features that will define the solution space for these problems, 3) an *observe-analyze-act* framework for research and development in hybrid networks, and 4) a discussion of the inherent tradeoffs associated with deploying hybrid networks "in the wild."

2 Hybrid packet/circuit switched data center

Hybrid packet and circuit switched networks ("HyPaC"s) augment an Ethernet packet switching layer with optical circuits and optical circuit switches to flexibly provide high bandwidth to applications on demand. We have proposed two systems, c-Through [21] and Helios [8], and have demonstrated the feasibility of constructing such networks using currently available devices to support a wide variety of applications and traffic patterns. We briefly describe these networks here.

2.1 Design principles

Electrical packet interconnects switch rapidly, potentially choosing a different destination for each packet. They are well suited for bursty and unpredictable communication. Optical interconnects, on the other hand, can provide higher bandwidth over longer distances (any 10Gbps links longer than about 10 meters must be optical). Optical switches, however, cannot be reconfigured rapidly: the MEMS-based optical circuit switches we use take between 10 and 25ms to reconfigure. As a result of these properties, the two network types offer substantially different advantages, with optics being superior for long-lasting, very high bandwidth point-to-point flows, and electronics winning for bursty, many-to-many local communication.

HyPaC networks aim to achieve the best of both worlds by using these two interconnects together in the same network. Typically, these networks first connect electrically a group of nodes (a "pod") to a set of "pod switches". The pod switches connect to a core packet switch at a high degree of over-subscription. The pod switches also connect several uplinks to an optical MEMS switch, which can configure a matching of pod-to-pod links. Figure 1 shows the HyPaC prototype that we perform our experiments over. At any time, nodes in one pod can communicate with limited bandwidth to *any* node in the network using the core packet switch, and with nodes in a few other selected pods at high bandwidth using the optical switch. The optical circuit bandwidth between pods is both less expensive and less flexible than packet-switched bandwidth. A key problem for both c-Through and Helios thus becomes how to schedule these circuits: For example, given four pods, each with one uplink to the circuit switch, should the system connect 1-2 and 3-4, or 1-3 and 2-4?

Both c-Through and Helios estimate the network traffic demand, identify pods that have long-lived, stable aggregated demand between them, and establish circuits between them to amortize the high cost of switching circuits. The two systems use different techniques to estimate traffic demand, but use the same greedy approach to schedule optical circuits to maximize the amount of data sent over optical circuits. Their centralized controllers dynamically configure the circuit and packet switches based upon the greedy scheduler's output every reconfiguration interval. Neither system incorporates history or state into their scheduling, and both are reactive to observed traffic in the network.

2.2 Overly restrictive design assumptions

These current proposals for hybrid networks make five overly restrictive assumptions about the traffic on the hybrid network:

Flows are independent and uncorrelated: Assigning circuits between two pods will not affect the bandwidth desired by other flows in the system.
Reality: Many flows depend on the progress of other flows, such as those that relay traffic through a node.

All flows have same priority: Helios and c-Through schedule optical circuits without taking into account any application or flow level traffic prioritization.
Reality: Data center traffic has a rich mix of priorities and fair sharing desires within and between applications.

Flows will not under-utilize the circuits: Once flows are assigned to circuits, they will continue to use the same (or more) bandwidth as their predicted demand indicated until the control process reassigns that circuit elsewhere (which might not happen for hundreds of milliseconds or more).
Reality: Perfectly predicting future traffic demand is not as easy as one might hope.

Randomly hashing flows to optical circuits is effective: Given a set of N 10Gbps circuits between two pods, we can create a single *superlink* of $10N$ Gbps capacity using random hashing, in which flows are randomly hashed across the set of available circuits.
Reality: Random hashing cannot support flexible distribution of circuit bandwidth that maximizes application performance.

All flows that can use a circuit should use that circuit: There is no cost to using an optical circuit when it is available, or to switching flows between the electrical and optical circuits. The Helios design made this assumption because of switch software limitations—traffic could not be scheduled on a per-flow basis to the electrical or optical network. The c-Through design made the same assumption to keep its optical/electrical selection tables small using only per-destination entries.

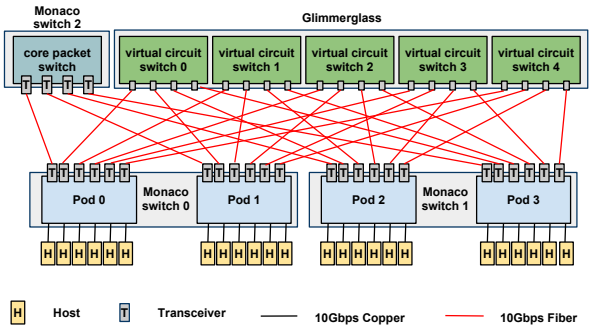

Figure 1: Hybrid testbed with 24 servers, 5 circuit switches and 1 core packet switch.

Reality: We have found several cases where keeping some lower-bandwidth, latency-sensitive flows on the packet switch reduces latency and variance for some applications.

3 Challenges to integrating optics into real data centers

This section demonstrates experimentally specific application traffic patterns that expose weaknesses in both Helios and c-Through. Figure 1 shows our prototype hybrid testbed with both electrical and optical circuit switches. We have 24 servers organized into 4 racks. Each rack is configured with five 10Gbps uplinks connected to Glimmerglass optical circuit switches and one 10Gbps uplink connected to a 10G Ethernet switch. To easily observe the effect of switching between optical path and electrical path, we rate limit the packet switched uplink to 5Gbps on all the racks. Therefore, each rack has 5Gbps bandwidth to all the other racks over packet switched network and up to 50Gbps bandwidth over optical circuits that can be reconfigured to different destination racks.

3.1 Effect of bursty flows

Both designs use the greedy Edmonds matching algorithm [7] to assign circuits between pods, based on an instantaneous snapshot of the traffic demand. Unfortunately, scheduling using an instantaneous demand estimate can find a locally maximal circuit configuration in which some circuits are under-utilized, while there exists another configuration with better overall circuit utilization.

We demonstrate this problem using a topology of three pods (Pod 0, 1 and 2), with two hosts each. Each pod switch has one 5Gbps uplink to the core packet switch and one 10Gbps uplink to the optical circuit switch. In this topology, only two pods can be connected optically at any given time. For the duration of the experiment, one host in Pod 1 sends data to a host in Pod 2 over a long-lived TCP connection (the "foreground flow"). The second host in Pod 1 sends data to a host in Pod 0 following a bursty ON-OFF pattern; we vary the burst ON-duration with the OFF-duration set to 2 second to observe the circuit scheduling decisions made by the Helios and c-Through circuit schedulers.

Figure 2 shows the average utilization of the optical link. In both designs, as the duration of the traffic bursts increases, the utilization of the link initially decreases and then increases as the bursts grow longer than 500ms. With short bursts, the circuit is

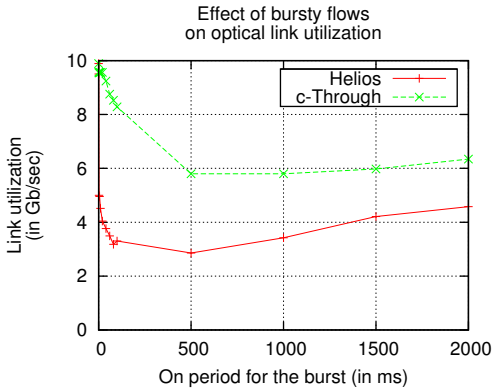

Figure 2: The utilization of a circuit in the presence of a bursty flow. By attempting to schedule the bursty flow on the circuit, bandwidth is reduced for the long-lived foreground flow.

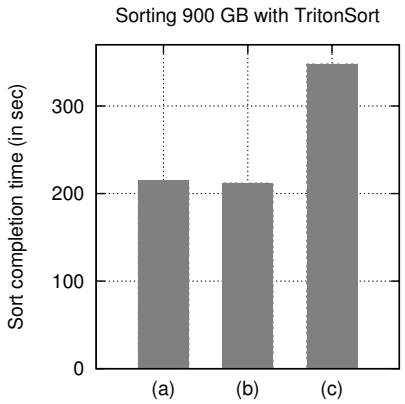

Figure 3: TritonSort completion time in three cases: (a) Fully non-blocking electrical switch (baseline case) (b) Helios/c-Through network with no background flows (c) Helios/c-Through network with one competing background flow

assigned to the bursty flow between Pod 1 and Pod 0, but after assignment, this flow goes quiescent, under-utilizing the optical capacity. In the next control cycle, the circuit is assigned back to the long-lived foreground flow. The control cycle is hundreds of milliseconds; bursts shorter than the control loop will reduce utilization for part of the control loop cycle. Longer bursts use the optical circuit for a longer fraction of the time it is assigned, improving overall optical utilization. Notably, the optical link capacity is never saturated by the long flow because of the constant flapping of the circuit between the pods.

3.2 Effect of correlated flows

An important component of data center traffic has a "shuffle", or all-to-all workload characteristic. For example, MapReduce [6] and Hadoop [1] require large scale sorting and shuffling of data between nodes. In this section, we evaluate the performance of a representative large scale sorting application, TritonSort [17]. We use TritonSort on Helios to show how suboptimal circuit scheduling impairs performance. TritonSort represents a balanced system where progress is largely a function of the speed of the slowest

flow; therefore, allocating high bandwidth to only few TritonSort flows will not improve overall performance. The key deficiency of the Helios and c-Through schedulers is that they do not take into account these dependencies across application flows.

This experiment measures the completion time of TritonSort running together with a long-lived background TCP flow that competes for circuit capacity. We use four pods, each connected with a 5Gbps uplink to the core packet switch and a 10Gbps uplink to the circuit switch. TritonSort sorts 900GB of data using nine nodes (three hosts in each of pods 0, 2, and 3). Meanwhile, another host in Pod 2 sends traffic to a host in Pod 1, competing with TritonSort for the Pod 2 circuit uplink. With only one circuit uplink, Pod 2 can connect optically to only one other pod at a time; if the scheduler creates a circuit from Pod 2 to Pod 1, the TritonSort flows from Pod 2 are sent over the slower packet switch. This imbalances the system, slowing it down. We measure the effect of this background flow on the completion time for TritonSort.

Figure 3 shows the completion time for sorting 900GB of data when the pods are inter-connected by 1) a fully non-blocking electrical switch (the baseline case), 2) Helios/c-Through network with no background flow, and 3) Helios/c-Through network with the background flow. Absent other traffic, Helios matches the performance of a non-blocking switch, but in the presence of competing flows, TritonSort's completion time increases substantially, taking as much as 69% longer.

3.3 The Hashing effect on multiple circuits

Data center networks increasingly use multi-path topologies to increase reliability and performance. Their switching hardware computes a hash of the packet header's source and destination information to determine which of several equal-cost paths to send each flow on; this mechanism ensures that packets within a flow follow the same path to avoid re-ordering, which can cause problems with TCP. While hash-based load balancing works well when the number of flows is large and individual flows are relatively small, it can perform worse under several real application workloads. For example, in a workload where most of the flows are mice (small) with only a few elephant flows, hashing uniformly across the entire set of flows can result in the elephant flows receiving too little bandwidth. We perform the following experiment to illustrates this problem with hashing.

4 hosts in Pod 0 send data to 4 hosts in Pod 1. These 4 flows between two pods are hashed over a circuit superlink with four individual circuit uplinks. If the hashing is truly uniform, each of these flows gets hashed to one unique circuit separately, and as a result each flow gets 10Gbps throughput. However, we observe that two of the flows get hashed to the same circuit due to hash collision. As a result two flows receive 10Gbps but the other two flows receive only 5Gbps and one of the circuits is not used at all.

3.4 Hardware issues

Other than the application and network level challenges, there are a few hardware hurdles to achieving fast and efficient network reconfiguration in HyPaC networks. The major problem is that current optical components are not designed to support rapid reconfiguration due to their original requirements in the telecom industry, and thus there has been a lack of demand to develop faster optical control planes.

The switching time of current optical switches is still far from their ideal limits. The minimum time for switching lots of lambdas (over 80) at once is still around 10 to 25 ms for commercially available optical switches. However, free-space MEMS switches should be able to achieve switching times as low as 1ms. Another hardware issue comes from the design of optical transceivers. Currently available transceivers do not re-sync quickly after a network reconfiguration. The time between light hitting the photoreceptor in the transceiver and the path being established is needlessly too long–as long as several seconds in practice [9]. The underlying transceiver hardware can theoretically support latencies as low as a few nanoseconds. Beyond the physical limitations, many of the control plane performance limitations are due to the electrical control aspects of the design, including the implementation of the switch software controller.

4 Solution space

This section explores the requirements for data center optical circuit controllers and sketches the design of an observe-analyze-act framework that can (help) meet them.

4.1 Design requirements

An optical circuit controller should be able to meet four key requirements:

Tolerate inaccurate demand information: It is difficult for a controller to have precise knowledge about all application's traffic demands and performance requirements. Existing systems infer these demands from counters in the network, but as we have shown, these heuristics can result in flapping circuits and suboptimal network performance. A good circuit allocation mechanism must be robust to inaccurate measurement of application traffic demands.

Tolerate ill-behaved applications: As we showed in previous sections, bursty data center applications can cause unnecessary network reconfiguration. Non-malicious but selfish applications could claim to need more capacity than they really do. All such ill-behaved applications can reduce the performance of a HyPaC network. The circuit controller must therefore be robust to their behavior, ensuring that the network's performance is not affected by them.

Support correlated flows: To achieve good application layer performance, the circuit scheduling module must be able to accommodate flows whose demand and performance depends on the performance of another flow. The underlying framework supporting the scheduler must provide fine-grained control to handle differently traffic that is on the critical path of an application vs. less important flows. It should also provide an avenue for the controller to gather sufficient information to understand the application's dependence upon a flow's performance.

Support flexible sharing policies among applications: Allocating circuits among mixed applications is challenging given the diversity of data center applications. Particularly in a multi-tenant cloud environment, applications may have very different traffic patterns and performance requirements. In addition, the management policies in data centers could assign applications different priorities. To share the limited number of optical circuits among these applications, the circuit allocator must be able to handle the performance interference among applications and support user-defined sharing policies among applications.

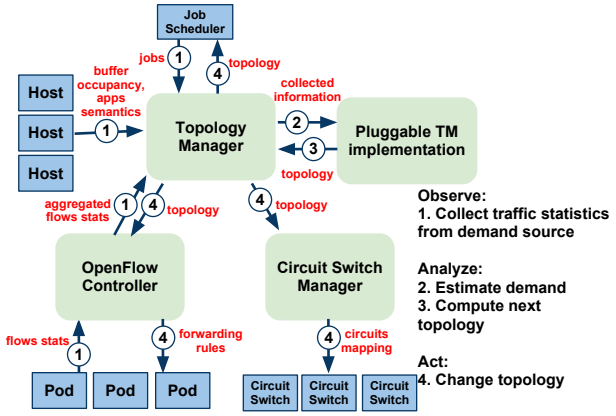

Figure 4: The control loop

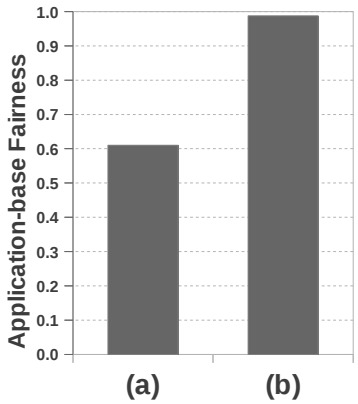

Figure 5: Application fairness based on Jain's fairness metric in (a) Helios/c-Through vs. (b) the proposed framework

4.2 An Observe-Analyze-Act framework

To achieve the above design requirements, the circuit controller must be able to obtain a detailed understanding of application semantics and fine-grain control of flows forwarding policies. We propose a three phase approach for managing HyPaC networks based on the *Observe-Analyze-Act* framework. To get a better understanding of the network dynamics and application heterogeneity in the cloud ecosystem, the HyPaC scheduler should be able to interact with different components and collect information from them. This information collected in this *Observe* phase would include the link utilization from the switches and application status from the cluster job schedulers (e.g., the Hadoop job Tracker), as well as application priorities and QoS requirements. The HyPaC manager then analyzes the aggregation of this information to infer the most suitable configuration for the network. The *Analyze* phase is a key step that helps the network controller understand the application semantics of traffic demand, detect ill-behaved applications, discover the correlated flows and therefore make the optimal configurations to support these applications. Finally, in the *Act* phase, it communicates this configuration to the other components in the system in order for the decision to be acted upon. The *Act* phase requires fine-grain control on the flow forwarding to support flexible configuration decisions and sharing policies.

We are prototyping a OpenFlow based system to support the *Observe-Analyze-Act* framework. Figure 4 shows the control loop and the main software components involved. Our prototype includes three main components: Topology Manager, OpenFlow Controller application and Circuit Switch Manager. The Topology Manager is the heart of the system, coordinating with other components to collect information for the Observe phase. To collect input from the network, we leverage the existing OpenFlow API using an OpenFlow Controller system (e.g., NOX, Beacon, or Maestro). We have implemented the Analyzer as a pluggable piece of software in the Topology Manager; this allows us to separate policy from mechanism and evaluate different optimization goals and algorithms. Based on the output of analysis, during the Act phase, the Topology Manager provides the new topology to the Circuit Switch Manager to configure the optical links, to the OpenFlow Controller application, and to the application job scheduler. The OpenFlow Controller configures the switches to forward traffic over appropriate links. The job scheduler can potentially use this information to schedule jobs that can take advantage of the resulting underlying network.

Our experience in using an OpenFlow controller for this prototype has been positive: it provides sufficiently fine-grained control over flow placement on the optical links, and it has allowed us to implement different hashing policies and flow management decisions that achieve better sharing among applications in the cloud. The overhead of installing new OpenFlow rules into Current OpenFlow-enabled devices can be quite high (e.g., one hardware switch we have access to can only accept six rule updates per second). However, newer devices already support much lower rule insertion overhead, and we expect that trend to continue moving forward. Our experience meshes well with prior work that used OpenFlow to create a unified control plane for IP/Ethernet and optical circuit-switched networks for long-haul backbone networks [5].

4.3 Usage examples

Although many design details remain to be fleshed out, we can already realize some flexible control over optical circuit using our fine-grain control framework. In this section, we demonstrate two simple examples: application level circuit sharing and ill-behaved flow detection.

Application level fairness: Consider the following scenario in a HyPaC system: Application A has 18 flows sharing a superlink of 20 Gbps capacity (consisting of two 10 Gbps circuit links) with application B, which sends 2 flows over the shared superlink. Since Helios/c-Through implement naive hashing of flows across the circuits in the superlink, each application gets a share of the optical link bandwidth proportional to the number of flows they have. This means that even though Helios/c-Through provide flow level fairness in the network, their flow management leads to unfair utilization of the superlink at the application level. We overcome this deficiency with our proposed OpenFlow integrated prototype. With more flexible ways to assign flows to the circuit, our prototype can guarantee different fairness objectives including application based fairness by first classifying flows into applications and then assigning flows such that application level fairness is achieved. Figure 5 shows how this modification to the scheduling algorithm increases the Jain's fairness index [12] for applications A and B compared to the naive Helios/c-Through policy. Importantly, no external input is required from the appli-

cations or network administrators to provide this application-level fairness. The controller can use port numbers as a coarse-grained way to classify flows into applications for use in subsequent circuit selection.

Ill-behaved flow detection: We have also implemented a simple heuristic in the *Analyze* phase of our prototype to eliminate the undesirable effect of the bursty flows. The idea is to identify the likely bursty flows and filter them out from the inter-pod traffic matrix before using that as the input for scheduling of circuits. We do this by maintaining idle and active counters for each flow that denote the number of control loop cycles for which the flow has been idle or active, respectively. According to our heuristic, a flow is considered active while it is sending data across the network and we classify a flow as non-bursty if its active counter is greater than a configurable threshold value. If a flow has been idle for a specific number of cycles, we again reset its idle and active counters in order to account for the changing nature of flow, e.g. when a previously stable flow becomes bursty. We evaluated the effectiveness of this approach by repeating the experiment from Section 3.1 for demonstrating the effect of bursty flows. Our results confirmed that our prototype avoided the unnecessary circuit flapping due to incorrect demand estimation for the bursty flows and the circuit link was utilized to its full capacity by the long foreground flow.

We emphasize that the point of these preliminary designs is not to show that they are the final answer to the challenges we described—indeed, we are fairly certain they are *not*. Instead, we believe they demonstrate that the framework we propose helps pave the way to future solutions in this area.

5 Discussion

The flexible control framework allows us to explore design choices in managing hybrid network in data centers. We hope to investigate these components in more detail in the future, and hope that this paper spurs others to do likewise.

Traffic analysis with application semantics: Efficient traffic engineering in HyPaC networks relies on precise and detailed traffic analysis. In addition to bursty flow detection, the traffic analyzer should detect any ill-behaved flows that may disrupt the network configuration or consume more bandwidth than their fair-share. This feature is important for system reliability. To support controlled sharing among mixed applications, a second challenge is learning the salient application characteristics, either by inferring them using traffic analysis or by devising a cluster-wide abstraction for applications to provide such information.

Traffic analysis requires thorough, but efficient, monitoring and aggregation of network traffic and status. We can classify application flows in controlled data center environments based on flexible header signatures, especially in OpenFlow deployments. Bursty flows can be detected by analyzing the flow counters on each switch. We can reveal correlated flows by analyzing the dependency of flow rate changes when the network is reconfigured. Given the frequent reconfiguration of circuits, the system level challenge is to perform traffic analysis accurately and efficiently with a limited number of flow samples, a problem similar to many in streaming databases.

Circuit scheduling: The objective of circuit scheduling is a combination of performance and fairness. For same-priority applications, the objective is to maximize the overall throughput of the optical circuits. For applications with different priorities, appropriate sharing policies must be defined (for example, strict priority, max-min fairness). Based on application semantics from the traffic analysis, different sharing policies can be implemented in the controller. The remaining questions are what scheduling algorithms will achieve optimal circuit utilization, and how the scheduling framework coordinates and balances the trade-offs between performance and sharing objectives.

HyPaC networks can achieve maximum optical throughput by using maximum weighted matching algorithms provided they accurately predict the application traffic demands for the next configuration round [21, 8]. Unfortunately, future application demand cannot be perfectly known for most applications. We are exploring different opportunities for improving the traffic demand prediction accuracy. For instance, instead of relying on heuristics alone for demand estimation (as in c-Through and Helios), one possible approach is to compare the measured and predicted throughput for each round and use that to predict the "natural" traffic demand for the application. When optical links are provisioned, they provide a fast interconnect to relieve the bandwidth bottleneck. Therefore, throughput on the optical links represents the natural traffic demand of applications. Assuming that the application traffic is stable at the pod level, we can combine the measured and estimated traffic demand to predict future application demands. There are many potential ways to combine these traffic matrices, such as weighted average, time sequence model fitting.

Another approach can be to collect explicit traffic demand from a centralized job scheduler. For example, the Topology Manager can communicate with the Hadoop Job Tracker so that Hadoop jobs can explicitly request optical links for certain period of time. The advantage of this approach is that applications can provide precise circuit request but this requires modifying applications to interact with the Topology Manager.

Efficient flow table update and aggregation: The OpenFlow-based control framework enables flexible and fine-grain forwarding decisions on both electrical and optical links. In a rapidly changing and high throughput data center environment, optical circuits are reconfigured in sub-second time scales with thousands of flows being impacted. It imposes significant challenges on the efficiency of OpenFlow control framework.

The challenges are two-fold: first, the rapid reconfiguration of optical circuits requires fast flow table updates on OpenFlow switches. On currently available OpenFlow switches, inserting flow rules is relative slow because they are not designed to support rapid network changes. A recent study [4] has reported that current implementation of OpenFlow switches can only achieve 275 flow setups per second. While OpenFlow based control is still an active research topic, it remains to be seen how fast we can achieve on rapid flow table updates. Recent proposals such as DevoFlow [4] can be leveraged to explore this space.

Second, due to limited number of flow table entries on pod switches, it is not feasible to install a forwarding rule for each flow and flow aggregation becomes necessary. This leads to another optimization objective: minimizing the number of forwarding entries required to configure the HyPaC network while preserving the desired traffic control policies. A possible solution is to group all the correlated flows and flows with the same priority into a single flow group, and compute the most concise *wildcard* rule associated with that flow group. It might be possible to leverage similar algorithms proposed in related work [22].

6 Related work

There have been various proposals that explore the feasibility and applicability of optical interconnects in the data center context [20, 15]. However, these papers focus on understanding the performance of optical interconnects, their spectral efficiency and power consumption, whereas this paper highlights the challenges in designing and building a network fabric that can effectively take advantage of optics in the data center. While both Helios and c-Through propose a hybrid data center architecture that uses both packet and circuit switches for interconnecting the top of rack (ToR) switches, there are other network designs that use optical switches very differently in the data center. For instance, Proteus [19] proposes an all optical backbone for the data center and only uses packet switches as ToR switches. Proteus relies on wavelength selective switches that transmit multiplexed wavelengths from a ToR to the optical switches. One disadvantage of the Proteus architecture is that it requires multi-hop routes for ToRs that do not connect to the same optical switch. This leads to unnecessary conversion of signals from optical to electrical and back.

Recent studies have explored the problem of sharing data center network among multiple tenants. SecondNet [11] is a virtual data center architecture that controls the bandwidth allocation to applications with different service types in hypervisors. Seawall [18] allocates network bandwidth among non-cooperative applications using congestion controlled, edge to edge tunnels. These studies are mostly focused on network sharing in virtualized data center environment, while our work targets specifically on HyPaC network and we explicitly address the optical circuit allocation problem for mixed applications.

Optical circuit scheduling has been studied extensively in the context of backbone network. For example, previous work [24, 25] have studied the circuit scheduling with QoS guarantee in optical burst switching network. In more recent work [2], Andrei et al. have studied the provisioning of data intensive applications over optical backbone network. Another work [5] proposes a unified control plane for IP/Ethernet and optical circuit switched network using OpenFlow. Another related effort along a similar approach to dynamically configure lightpaths is [14] in which the authors augment wide area networks with high bandwidth optics. This work implicitly applies the idea of observe-analyze-act framework which automatically configures the optical part of the network without any user intervention or the need for modification in the applications. Our work is different from these existing studies given the different technology assumptions in the data center as compared to wide-area and grid computing environments. Supporting the highly dynamic requirements of data center applications requires that optical circuits must be reconfigured in sub-second timescales. Therefore, the circuit allocation mechanism in data centers must be able to adapt to traffic changes more quickly than in wide-area provisioning and grid link setup tasks.

Another line of related works is recent proposals to control networks with a software controller with global view of the topology (e.g. [10, 23, 3]). The idea is to provide a high level logically centralized (potentially physically replicated) software component to observe and control a network. The main advantages of this is flexibility as the network programs/protocols would be written as if the entire network were present on a single decision element as opposed to requiring a new distributed algorithm across all network switching elements. Also, programs may be written in terms of high-level abstractions that can ignore messy details of the lower-level configuration parameters (e.g., IP and MAC addresses of individual network elements). Onix [13] is an example of a centralized controller built over OpenFlow which is deployed in production data center network. This gives us confidence in our design that also leverages OpenFlow in the Observer and Act phases of our proposed framework.

7 Conclusion

Hybrid electrical/optical networks show significant promise for supporting the ever increasing demand placed on data center networks. Unfortunately, several unsolved challenges could prevent their adoption. This paper enumerated a set of challenges encountered during a year of experience with hybrid networks deployed on real hardware. Although the complete picture of how to build these networks is still unknown, we propose a flexible and framework based on OpenFlow for that we believe will enable a variety of future solutions to the remaining challenges. We are optimistic that solving these problems will lead to increased adoption of optical circuit switching into data center networks, leading to lower data center costs, complexity, and energy use.

Acknowledgments: This research was supported in part by NSF awards 0546551, 0619525, CNS-0448546, CNS-0546551, CNS-0721990, MRI-0923523, the Alfred P. Sloan Foundation, the NSF Center for Integrated Access Networks (CIAN), and by the Intel Science and Technology Center for Cloud Computing.

References

[1] Apache Hadoop, http://hadoop.apache.org.

[2] D. Andrei. Efficient provisioning of data-intensive applications over optical networks. Fall 2009. Ph.D. thesis, UC. Davis.

[3] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. In *Proceedings of ACM SIGCOMM '07 Computer Communication Review*. ACM, Aug. 2007.

[4] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. Devoflow: Scaling flow management for high-performance networks. In *ACM SIGCOMM*, Aug. 2011.

[5] S. Das, G. Parulkar, P. Singh, D. Getachew, L. Ong, and N. Mckeown. Packet and circuit network convergence with openflow. In *Optical Fiber Conference (OFC/NFOEC'10)*, Mar. 2010.

[6] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI 2004*, pages 137–150, Apr. 2004.

[7] J. Edmonds. Paths, trees and flowers. *Canadian Journal on Mathematics*, pages 449–467, 1965.

[8] N. Farrington, G. Porter, S. Radhakrishnan, H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat. Helios: A hybrid electrical/optical switch architecture for modular data centers. In *ACM SIGCOMM*, Aug. 2010.

[9] N. Farrington, Y. Fainman, H. Liu, G. Papen, and A. Vahdat. Hardware requirement for optical circuit switched

data center networks. In *Optical Fiber Conference (OFC/NFOEC'11)*, Mar. 2011.

[10] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. Clean slate 4d approach to network control and management. In *Proceeding of ACM SIGCOMM Computer Communication Review*. ACM, Oct. 2005.

[11] C. Guo, G. Lu, H. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. Secondnet: A data center network virtualization architecture with bandwidth guarantees. In *ACM CoNEXT'10*, Dec. 2010.

[12] R. Jain, D. Chiu, and W. Hawe. *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corp., 1984.

[13] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: a distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*. USENIX Association, Oct. 2010.

[14] J. R. Lange, A. I. Sundararaj, and P. A. Dinda. Automatic dynamic run-time optical network reservations. In *In Proceedings of the Fourteenth International Symposium on High Performance Distributed Computing (HPDC)*, pages 255–264, July 2005.

[15] H. Liu, C. F. Lam, and C. Johnson. Scaling optical interconnects in datacenter networks opportunities and challenges for wdm. In *Proceedings of the 2010 18th IEEE Symposium on High Performance Interconnects*, HOTI '10. IEEE Computer Society, Aug. 2010.

[16] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *ACM SIGCOMM CCR*, Mar. 2008.

[17] A. Rasmussen, G. Porter, M. Conley, H. V. Madhyastha, R. N. Mysore, A. Pucher, and A. Vahdat. Tritonsort: A balanced large-scale sorting system. In *USENIX NSDI'11*, Mar. 2011.

[18] A. Shieh, S. Kandula, A. Greenberg, and C. Kim. Sharing the data center network. In *USENIX NSDI'11*, Mar. 2011.

[19] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang. Proteus: a topology malleable data center network. In *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, Oct. 2010.

[20] A. Vahdat, H. Liu, X. Zhao, and C. Johnson. The emerging optical data center. In *Optical Fiber Communication Conference*. Optical Society of America, Mar. 2011.

[21] G. Wang, D. G. Andersen, M. Kaminsky, D. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. Ryan. c-Through: Part-time optics in data centers. In *ACM SIGCOMM*, Aug. 2010.

[22] R. Wang, D. Butnariu, and J. Rexford. Openflow-based server load balancing gone wild. In *Workshop of HotICE'11*, Mar. 2011.

[23] H. Yan, D. A. Maltz, T. S. E. Ng, H. Gogineni, H. Zhang, and Z. Cai. Tesseract: A 4D Network Control Plane. In *USENIX NSDI'07*, Apr. 2007.

[24] M. Yoo, C. Qiao, and S. Dixit. Qos performance of optical burst switching in ip-over-wdm networks. *IEEE Journal of Selected Areas in Communications, Vol.18, No.10*, 2000.

[25] M. Yoo, C. Qiao, and S. Dixit. Optical burst switching for service differentiation in the next-generation optical internet. *IEEE Communication Magzine*, Feb. 2001.