Quartz: A New Design Element for Low-Latency DCNs

Yunpeng James Liu
University of Waterloo
y493liu@uwaterloo.ca

Peter Xiang Gao
UC Berkeley
petergao@berkeley.edu

Bernard Wong
University of Waterloo
bernard@uwaterloo.ca

Srinivasan Keshav
University of Waterloo
keshav@uwaterloo.ca

ABSTRACT

Most datacenter network (DCN) designs focus on maximizing bisection bandwidth rather than minimizing server-to-server latency. We explore architectural approaches to building low-latency DCNs and introduce Quartz, a design element consisting of a full mesh of switches. Quartz can be used to replace portions of either a hierarchical network or a random network. Our analysis shows that replacing high port-count core switches with Quartz can significantly reduce switching delays, and replacing groups of top-of-rack and aggregation switches with Quartz can significantly reduce congestion-related delays from cross-traffic. We overcome the complexity of wiring a complete mesh using low-cost optical multiplexers that enable us to efficiently implement a logical mesh as a physical ring. We evaluate our performance using both simulations and a small working prototype. Our evaluation results confirm our analysis, and demonstrate that it is possible to build low-latency DCNs using inexpensive commodity elements without significant concessions to cost, scalability, or wiring complexity.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network topology

Keywords

Optical Technologies; WDM; Datacenter; Latency

1. Introduction

Network latency is a critical factor in determining the performance of many datacenter-scale, distributed applications. For example, in most distributed realtime computation frameworks such as MPI [24] and Storm [13], network latency manifests as a substantial component of coordination delay and is therefore on the critical path. In realtime or interactive applications such as search engines, social networks, and high frequency financial trading, a wide-area request may trigger hundreds of message exchanges inside a datacenter. For example, in a measurement study from Facebook, servicing a remote HTTP request can require as many as 88 cache lookups, 35 database lookups, and 392 backend remote procedure calls [23]. Therefore, there is an urgent, market-driven need for reducing network latency.

SIGCOMM'14, August 17–22, 2014, Chicago, IL, USA.

http://dx.doi.org/10.1145/2619239.2626332 .

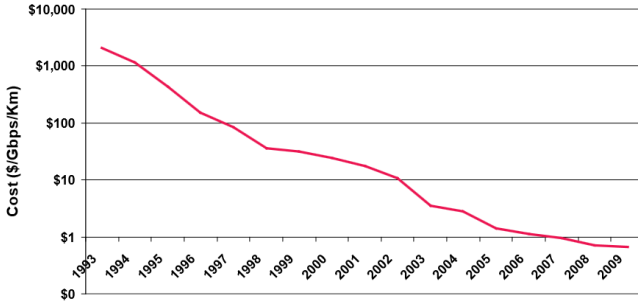


Figure 1: Backbone DWDM per-bit, per-km cost improvements over time [20].

Past work on low-latency datacenter networks (DCNs) has primarily focussed on reducing packet processing delays in the OS network stack [31], network interface card (NIC) [32] and network switches [3], and avoiding network congestion through bandwidth reservations [29] or congestion-aware flow scheduling [19, 27, 40, 41, 43]. Combining these techniques can, in theory, result in an order of magnitude reduction in end-to-end network latency.

However, there are a number of limitations with current techniques. For example, state-of-the-art low-latency cut-through switches are limited in scale ($\leq$ 64 ports) compared to standard store-and-forward switches ($\leq$ 1024 ports). Large DCNs must choose to deploy either standard switches in their core-switching tier or introduce an additional switching tier, which not only increases the number of switching hops but also creates additional congestion points.

In this paper, we explore an *architectural approach* to reducing network latency that is complementary to previous efforts. Our key insight is that switching latency is minimized by interconnecting top-of-rack (ToR) switches with a full mesh instead of a high-port count core switch or an aggregation layer of switches. Unfortunately, a full mesh topology is a wiring nightmare and does not scale well. We overcome wiring complexity by exploiting commodity photonics in the form of optical wavelength division multiplexers (WDMs) to implement a complex O(n^2) mesh as a simple O(n) optical ring. Similar techniques have been used to construct wide-area optical mesh networks [11, 34]. We show that implementing a full mesh in an optical ring is equally applicable in a datacenter. Using commodity photonics allows our solution, *Quartz* to ride the cost reduction curve in WDM equipment, which is driven by the large-scale roll out of fiber-to-the-home (see Figure 1 [20]).

To deal with the limited scalability of a mesh topology, we envision that Quartz can be used to replace *portions* of traditional DCNS:

Component	Standard	State of Art
OS Network Stack	15μs [36]	1 - 4 μs [31]
NIC	2.5 - 32μs [36]	0.5μs [32]
Switch	6 μs [5]	0.5μs [3]
Congestion	50μs [31]	

Table 2: Network latencies of different network components.

- Replacing large top-of-rack (ToR) switches in single-tier networks with a single Quartz mesh to reduce latency and allow incremental upgrades.
- Replacing both ToR and aggregation switches in a standard 3-tier network with a single Quartz tier to significantly reduce the impact of cross-traffic.
- Replacing core switches to provide a large-scale, low-latency, and incrementally upgradable core switching tier.
- Replacing sets of switches in a randomized DCN [37, 38] to provide lower latency for traffic with strong locality.

We evaluate Quartz through both simulations using a packet-level simulator and experiments on a working prototype consisting of four switches and six WDM muxes/demuxes. Our results demonstrate that Quartz can significantly reduce both switching and queuing latencies on various application workloads. We evaluate the cost of Quartz and show that it is cost competitive for many deployment scenarios. Moreover, we expect the cost of our solution to diminish over time as WDM shipping volumes rise.

Overall, our work makes three contributions:

- We present a novel use of commodity photonics in DCNs to build Quartz, a WDM-based design element for reducing communication latency.
- We show how Quartz can be used in different topologies to reduce latency and congestion.
- We evaluate performance using both simulations and a small working prototype and find that Quartz can reduce latency by 50% in many scenarios.

2. Background and Related Work

This section describes the different sources of network latency and the previous work to address each source. It also introduces the optical network technologies used in Quartz.

2.1 Sources of Latency

There are many sources of latency in DCNs (see Table 2 for a summary). We discuss each in turn.

2.1.1 Network Stack

Packets often need to wait for one or more context switches before they are received by a user-space application. Techniques such as kernel bypass and zero-copy processing can reduce this latency. In recent work, the Chronos [31] system uses these techniques to significantly reduce the operating system kernel latency for datacenter applications.

2.1.2 Network Interface Cards

Commodity Network Interface Cards (NICs) can introduce tens of microseconds of latency from performing basic buffering and packet processing. Recent work shows that by offloading packet processing to a FPGA and optimizing the communication between the FPGA and the host processor, NIC latency can be reduced to hundreds of nanoseconds [32]. Alternatively, RDMA over Infiniband or Ethernet [28] can reduce latency by eliminating communication between the NIC and the host processor for certain operations.

2.1.3 Switch Latency

Switching delay is a significant source of network latency. For example, the Cisco Catalyst 4948 10 Gigabit Ethernet Switch, which is commonly used in modern datacenters, has a switching latency of at least 6μs. In a typical three-tier network architecture, switching delay can therefore be as high as 30μs. Switching delay can be reduced by having fewer network tiers, and adopting low-latency cut-through switches. Unlike store-and-forward switches, cut-through switches start to send a frame before fully receiving it. Low-latency switches, such as the Arista 7100 [3], have a switching delay of approximately 500 ns. Cut-through switches command only a small price premium over the same port density store-and-forward switches [2, 6]. Unfortunately, they are currently limited to 64 ports, compared to more than 1000 ports for standard switches, and are therefore mainly used as ToR or aggregation switches.

2.1.4 Congestion

Although current datacenter networks support high bisection bandwidths, bursty traffic can still increase queuing delay over short time scales adding tens of microseconds to end-to-end latency. Recent work has identified several techniques for reducing network congestion. For example, DCTCP [19] utilizes Explicit Congestion Notifications (ECN) as congestion signals. However, DCTCP is only partially effective at reducing congestion delays over time scales shorter than a few round-trip-times. Recent proposals such as D^2TCP [40] and PDQ [27] use distributed congestion avoidance algorithms to manage flows at end-hosts and switches respectively.

DeTail [43] reduces network latency by detecting congestion and selecting alternative uncongested paths to reduce queuing delay. Deadline Driven Delivery (D^3) [41] is deadline-driven protocol where the sender requests the amount of bandwidth equal to the total amount of data divided by the time to the deadline. D^3 does not coexist with legacy TCP and requires that the user application knows the size and deadline of each of its flows, which in practice can lead to significant underutilization.

These protocol-based techniques require significant changes to the application, are unable to scale to a large number of short flows, and are limited by the amount of path diversity in the underlying network topology.

2.1.5 Topology

Topology is a critical factor in determining a network's equipment and deployment cost, wiring complexity, scalability, and performance characteristics (bisection and end-to-end latency). We outline the latency characteristics of different topologies next.

Tree Networks: Tree topologies, such as the standard multi-root tree structure [1] and Fat-Tree [17], organize the network into multiple switch tiers. Switches in each tier are only connected to switches in adjacent tiers, with the lowest tier of ToR switches connected to servers. This topology is scalable and, at least for the standard multi-root tree structure, has relatively low wiring complexity. However, each tier increases the maximum hop-count by two. The additional switching tiers also create focal points for congestion because the path diversity is very low. Even for Fat-Tree, where there is significant path diversity, congestion due to cross-traffic from different racks is still possible unless every flow is scheduled to use completely independent paths, which is difficult for short flows.

Server-Centric Networks: DCell [26], BCube [25] and CamCube [16] are networks that use servers as switches to assist in packet for-

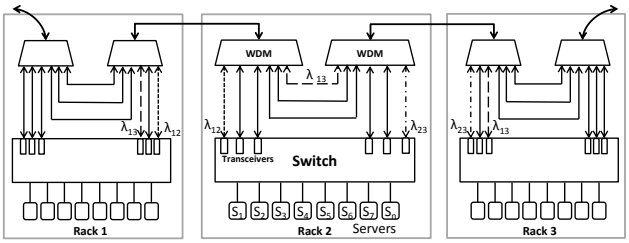


Figure 3: Quartz switches with $n = 8$ and $k = 6$. Each switch is only physically connected to two nearby switches by an optical cable. Switch 1 and switch 2 are connected using channel λ_{12}. Switch 1 and switch 3 are connected using wavelength channel λ_{13}.

warding. These are scalable networks with high bisection bandwidth and path diversity. However, using servers to perform packet forwarding can introduce substantial delays in the OS network stack. Furthermore, server-centric networks can reduce the performance of computationally-intensive jobs, because high-bandwidth packet forwarding requires a significant number of CPU cycles [16, 37].

Randomized Networks: SWDC [37] and Jellyfish [38] propose to randomly connect servers or switches in a datacenter network. Random topologies usually have high path diversity and high bisection bandwidth. However, the worst case network diameter is typically much larger than a similar size tree network, even if the average path length is smaller. Randomized networks are also difficult to deploy and manage as they have very high wiring complexity.

Mesh Networks: Mesh networks directly connect every node to every other node, where a node can either be a server or a switch. A full mesh network provides the lowest possible network diameter and eliminates congestion arising from cross-traffic from other nodes. These properties make mesh networks an attractive option for low-latency DCNs. However, mesh topologies are rarely used in DCNs because the O(n^2) connections requirement greatly limits scalability and increases wiring complexity.

2.2 Optical Network Technologies

Our work takes advantage of optical multiplexers and demultiplexers (mux/demux) to reduce the wiring complexity of mesh networks. An optical mux/demux uses Wavelength Division Multiplexing (WDM) to carry multiple network connections using different wavelengths in a single optical cable. Through judicious selection of wavelengths, it is possible to implement a full mesh network as a small set of optical rings that share a single physical ring. Importantly, unlike packet switching approaches, optical multiplexing and demultiplexing does not introduce additional switching or queuing latency.

It is also important to distinguish between a WDM and an optical switch. A WDM is a low-cost commodity photonic element that is deployed at mass-scale to build fiber-to-the-home networks. Figure 1 (taken from [20]) shows that the cost of DWDMs (Dense WDMs) have fallen at an exponential rate since 1993. Assuming this trend continues to hold, Quartz will only become more cost-competitive over time. In contrast, optical switches are complex, low-volume, and expensive to build due to the use of custom ASICs.

3. Quartz

Quartz is a *WDM-ring network* that implements a full mesh as a single physical ring. We now describe it in more detail.

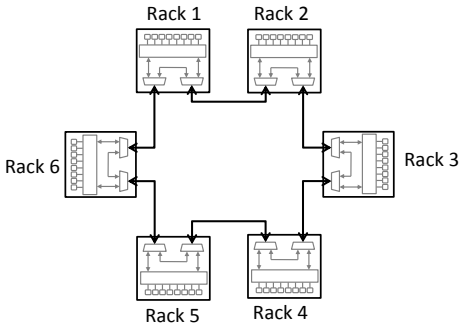


Figure 4: Quartz ring of size 6. The logical topology is equivalent to a mesh structure where any two switches are directly connected. By using WDM, only adjacent switches are connected with a optical fiber.

Each Quartz switch has, in addition to n standard ports, k *optical transceivers*[1]. These devices convert electrical signals to and from optical signals. Transceivers can be tuned to specific wavelengths of light. Two transceivers connected by an optical cable must be tuned to the same wavelength to communicate.

Each switch on a Quartz ring is also associated with an *add/drop multiplexer*. The multiplexer portion combines optical signals of different wavelengths (also called 'channels') onto a single *multiplexed optical signal* for transmission on a single optical cable. Conversely, the demultiplexer portion splits a multiplexed optical signal into its constituent channels, with each channel placed on a separate optical cable.

Direct connection between switches s and t on the Quartz ring requires allocating them a dedicated channel denoted λ_{st}. Moreover, one of the transceivers on each switch is tuned to this channel. Switch s uses the demultiplexer to remove ('drop') all channels of the form λ_{*s} from the ring and to add channels of the formλ_{s*} to the ring. Thus implementing a full mesh requires only *two* physical cables to connect to each Quartz switch.

Figure 3 shows a small Quartz ring. In this example, switch 1 and switch 2 are directly connected and they communicate using channel λ_{12}. Switch 1 and switch 3 communicate with each other using channel λ_{13}. At switch 2, the λ_{13} channel in the multiplexed optical signal essentially passes through to switch 3. Therefore, there is an optical connection between switch 1 and switch 3, even through they are not physically connected.

Quartz allows for a flexible tradeoff between cost and network oversubscription. As shown in Figure 3, the switches in Quartz have two parameters n (# ports to server) and k (# ports to other switches), where $n : k$ is the server-to-switch ratio, and $n+k$ is the port density of a switch. A DCN designer can reduce the number of required switches in the network by increasing the server-to-switch ratio at the cost of higher network oversubscription.

3.1 Channel Assignment

Note that communication between switch s and switch t in Quartz requires them to have exclusive ownership of channel λ_{st}. If an optical cable could support an infinite number of channels, we could build an optical ring of arbitrarily large size that supports pairwise communication between switches. However, current technology can only multiplex 160 channels in an optical fiber and commodity Wavelength Division Multiplexers can only support about 80 chan-

[1]Most 10GigE and all future 40/100GigE ToR switches already use optical transceivers due to their lower power consumption and higher signal quality.

nels. Therefore, we need to determine the optimal way to assign channels such that we can build a ring of size K using the minimum number of channels.

Quartz attempts to assign Λ available channels to each pair of switches in a ring of size M using two principles: (1) For any two switches s, t in the ring, there exists an optical path between them using wavelength λ_{st}. (2) For all optical links on the path between s and t, there is no other channel using the same wavelength λ_{st}. For example, in Figure 4 if switch 1 and switch 3 are using wavelength λ_{13}, then using λ_{13} between rack 2 and rack 4 should be avoided, because λ_{13} would be used twice on the link between switch 2 and switch 3.

Given these constraints, the wavelength assignment problem can be formulated as an Integer Linear Program (ILP) similar to [42]. The ILP problem is known to be NP-Complete. However, for a small ring, we can still find the optimal solution by ILP. We also introduce a greedy packing algorithm to calculate the minimum number of wavelengths needed for building such a ring for larger ring sizes. The ILP Formulation and our greedy algorithm are as follows:

Let $C_{s,t,i}$ denote the clockwise path from s to t using channel $i \in \{1, ..., \Lambda\}$ and let $C_{t,s,i}$ denote the anti-clockwise path. Variable $C_{s,t,i}$ is set to 1 if channel i is used for communication between s and t. Each s, t switch pair should use one channel for their communication on either clockwise or counter-clockwise direction (Eq. 2).

Variable $L_{s,t,i,m}$ is the indicator variable of whether link m, the link between switch m and $(m+1)\ mod\ M$, is using channel i for the path between switch s and t. We define static value $P_{s,t,m} = 1$ if the clockwise path between s and t passes through link m. If link m is on the path between switch s and switch t, and wavelength i is used for their communication $L_{s,t,i,m}$. This is guaranteed by Eq. 3.

On each link m, a single channel i should be only used at most once. We ensure this set of constraints by Eq. 4. To count the total number of channels used, variable λ_i is created to show whether channel i is used in the ring. Eq. 5 makes sure that λ_i equals 1 if channel i is used in the ring.

$$\text{minimize: } \sum_i \lambda_i \tag{1}$$

subject to:

$$\forall s < t,\ \sum_i C_{s,t,i} + \sum_i C_{t,s,i} = 1 \tag{2}$$

$$\forall s, t, i, m,\ L_{s,t,i,m} = P_{s,t,m} C_{s,t,i} \tag{3}$$

$$\forall m,\ i, \sum_{s,t} L_{s,t,i,m} \leq 1 \tag{4}$$

$$\forall i,\ \sum L_{s,t,i,m} \leq M\lambda_i \tag{5}$$

$$\forall\ variable \in \{0, 1\} \tag{6}$$

The goal is to minimize the total number of used channels. If the ILP is solvable, it means all switch pairs can communicate with each other. The optimization result is the minimum number of channels required for the given ring size.

3.1.1 Greedy Channel Assignment

We outline a simple, greedy algorithm to solve the channel assignment problem. For all the paths between switch pairs (s, t), they are first sorted by their length. For a ring with M nodes, the maximum path length between two switches is $\lfloor M/2 \rfloor$, so there are $\lfloor M/2 \rfloor$ sets of path lengths. Consider an algorithm with $\lfloor M/2 \rfloor$ iterations, where paths in each set is assigned in one iteration. Our heuristic is to give priority to long paths to avoid fragmenting the available channels on the ring. Shorter paths are assigned later because short path are less constrained on channels that are available on consecutive links. In each iteration, starting from a random location, the channels are greedily assigned to the paths until all paths are assigned or the channels are used up.

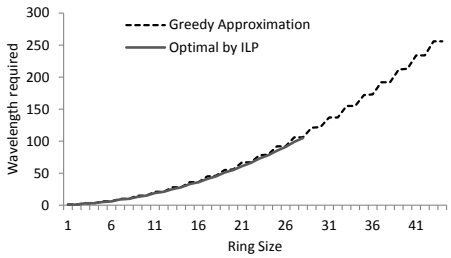


Figure 5: Optimal wavelength assignment

To evaluate the greedy algorithm, we compare the results with the ILP solution. Figure 5 illustrates the results of this evaluation, and shows that our greedy heuristic performs nearly as well as the optimal solution. Furthermore, it shows that the maximum ring size is 35 since current fiber cables can only support 160 channels at 10 Gbps. Note that wavelength planning is a one-time event that is done at design time. Quartz does not need to dynamically reassign wavelengths at runtime. Since we can use a fixed wavelength plan for all Quartz rings of the same size, wavelength planning and switch to DWDM cabling can be performed by the device manufacturer at the factory. Moreover, our greedy heuristic only requires seconds to compute on a standard workstation even for a ring size of 35. Therefore, we believe our greedy algorithm is fast enough for practical use.

3.2 Scalability

Because of its full mesh structure, the maximum size of a Quartz network is, in part, limited by the port count of the switches. Using low-latency 64-port switches, where each switch connects each one of 32 of its ports to a different switch, this configuration mimics a 1056 (32×33) port switch. This relatively small maximum network size suggests that Quartz should be used as a component in new DCN designs, rather than as a replacement for existing DCNs. We explore using Quartz as a network design element in Section 4.

For deployment scenarios where a larger Quartz network is necessary, one can increase the size of the network by connecting each server to more than one ToR switch. For example, for a configuration where (1) each server has two NICs, (2) there are two top-of-rack switches in each rack, (3) each server is connected to both switches in its rack, and (4) each rack has a direct connection to every other rack, the longest path between any two servers is still two switches. This configuration can support up to 2080 (32×65) ports at the cost of an additional switch per rack, and a second optical ring since wavelength restrictions limit the number of switches per ring to 35. In theory, even larger Quartz networks can be constructed by adding more switches per rack and more optical rings. However, we believe this approach to scalability is cost prohibitive in practice. For most deployment scenarios that require more than 1056 ports, Quartz should instead be used as a component in a large DCN.

3.3 Insertion Loss

Although an optical hop between switches does not introduce any discernible amount of latency, it does degrade the optical signal

due to insertion loss from the WDMs. Quartz compensates for this signal degradation by adding pump laser-based signal amplifiers as needed between optical hops.

To determine the feasibility of this approach, we evaluate the cost of adding signal amplifiers in a 24-node Quartz ring. In this example, we use 10Gbps DWDM transceivers [7], and 80 channel DWDMs [8]. The transceiver has a maximum output power of 4 dBm and a receiver sensitivity of -15 dBm. A typical 80 channel DWDM has an insertion loss of 6 dB. Therefore, the number of DWDMs that a channel can pass through without amplification is:

$$(4dBm - (-15dBm))/6dB = 3.17$$

In this configuration, we need to add a signal amplifier [12] after every sequence of three DWDMs. Since each optical hop in the ring requires traversing two DWDMs, we need one amplifier for every two switches. This only increases the cost of a 24-node Quartz ring by three percent. To avoid overloading the amplifiers and transceivers, we also need to add optical attenuators to the ring. However, attenuators [10] are simple passive devices that do not meaningfully affect the cost of the network.

3.4 Routing in Quartz

We now discuss the integration of Quartz into link layer addressing and routing. A naïve approach to routing in Quartz would be to treat all servers as being in the same L2 Ethernet network. However, because Ethernet creates a single spanning tree to determine the forwarding path of packets, it can only utilize a small fraction of the links in the network. To utilize all direct paths between switches, we advocate using ECMP routing in the Quartz's mesh. Since there is a single shortest path between any pair of switches in a full mesh, ECMP always selects the direct one-hop path, which minimizes hop count and interference from cross-traffic.

A possible problem with only using the direct paths in Quartz is the amount of bandwidth oversubscription between each pair of switches. In a Quartz configuration using 64-port switches where 32 ports from each switch are connected to servers, there is a 32:1 oversubscription between racks, which can be a problem for certain workloads. However, workloads that spread traffic across different racks, such as standard scatter/gather workloads in large-scale computational platforms such as MPI [24] and MapReduce [21], are unaffected by this kind of independent, rack-to-rack bandwidth oversubscription.

For workloads that concentrate traffic between two racks, one can significantly reduce rack-to-rack oversubscription by using Valiant Load Balancing (VLB) [22,33] to make use of two-hop routes. We configure each switch to send k fraction of the traffic through the $n-2$ two-hop paths and the remaining fraction through the direct path. For instance, if there is a large amount of traffic from rack 6 to rack 3 in Figure 7(b), VLB will send k fraction of this traffic through Rack 1, 2, 4, and 5 over two-hop paths. The parameter k can be adaptive depending on the traffic characteristics. We provide a detailed bandwidth analysis of Quartz and different tree topologies in Section 5.1. Quartz not only reduces transmission delay, it also reduces queuing delay by reducing congestion due to cross-traffic. We evaluate the impact of cross-traffic on latency in Section 6.1.

3.5 Fault Tolerance

Rings are well known to be less fault tolerant than a multi-rooted tree: two link failures in a ring partition the network. However, by using multiple physical optical fibers to interconnect switches and multi-hop paths, we can significantly reduce the likelihood of partitioning the network. For example, if a Quartz network with 33 switches requires 137 channels, we can use two 80-channel WDM muxes/demuxes instead of a single mux/demux at each switch. In this configuration, there will be two optical links between any two nearby racks, forming two optical rings, and link failures are less likely to partition the network. Of course, this resilience comes at an additional cost.

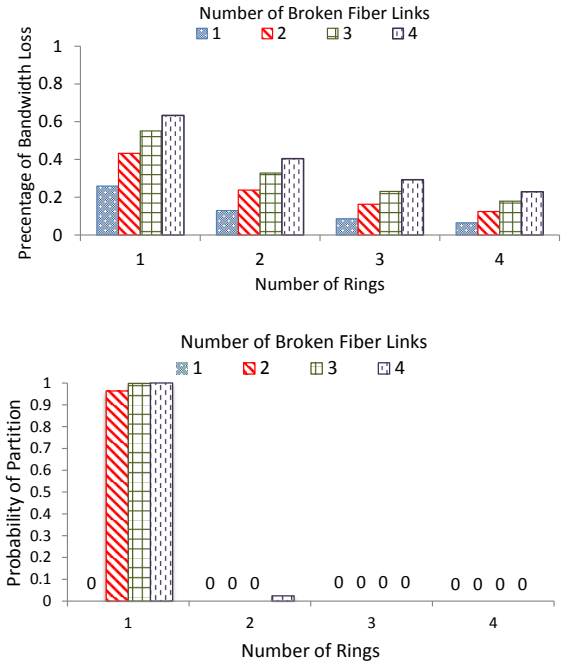


Figure 6: The top graph shows the percentage of bandwidth loss from broken fibre links with different ring sizes. The bottom figure shows the probability of network partitioning. With two rings, four link failures can partition the network at a very low probability of 0.0024.

Since combinatorial analysis of multi-hop path reachability is complex, we use simulations to evaluate the performance of Quartz with one to four physical rings under random link failures. Figure 6 shows the average bandwidth loss and probability of the network partitioning in a 33-switch Quartz network. The top figure shows the percentage of aggregate bandwidth loss. With only one ring, a physical optical link failure results in a 20% reduction of the network bandwidth. Using 4 rings, the average bandwidth reduction is only 6%. The network partition probability for two or more link failures in a single-link network is more than 90%. Surprisingly, by adding a single additional physical ring, the probability of the network partitioning is less than 0.24% even when four physical links fail at the same time.

4. Quartz as a Design Element

A Quartz mesh has low latency, high path diversity with VLB, and the same wiring complexity as ring networks. Yet its scalability is limited by the size of low-latency cut-through switches. A Quartz network built using 64-port switches and a single switch per rack provides 1056 server ports, which, as a DCN, is only sufficient for small datacenters.

Larger datacenters can instead use Quartz as a design element to, for instance, replace portions of their DCNs in order to reduce switching or congestion delays for traffic between servers in nearby racks. In the following sections, we explore using Quartz in the edge as a replacement for both the ToR and aggregation tiers in

Datacenter Size	Network Utilization	Sample Topologies	Latency Reduction using Quartz	Cost/Server
Small (500 Servers)	Low	Two-tier tree	33%	$589
		Single Quartz ring		$633
	High	Two-tier tree	50%	$589
		Single Quartz ring		$633
Medium (10K Servers)	Low	Three-tier tree	20%	$544
		Quartz in edge		$612
	High	Three-tier tree	40%	$544
		Quartz in edge		$612
Large (100K Servers)	Low	Three-tier tree	70%	$525
		Quartz in core		$525
	High	Three-tier tree	74%	$525
		Quartz in edge and core		$614

Table 8: Approximate cost and latency comparison. Costs include all the hardware expenses except servers. Costs of Quartz include a single ring of switches.

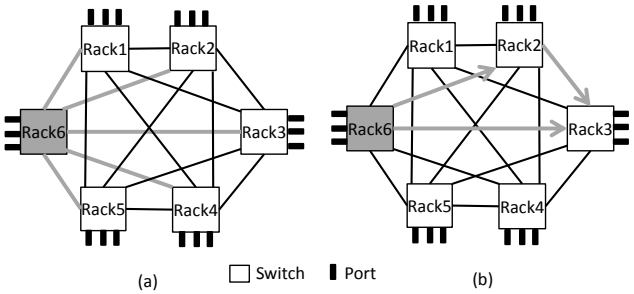


Figure 7: A flat mesh network where ToR switches are directly connected. (a) direct routing (b) two-hop routing.

a 3-tier tree network, in the core tier as a replacement for large, high-latency core switches, and in randomized networks to reduce switching and congestion delays for traffic with strong locality.

4.1 Quartz in the Edge

Current DCNs typically employ either a 2 or 3-tier tree topology [17]. Figure 15(a) illustrates a 3-tier tree network in which each ToR switch in the edge tier is connected to one or more switches in the aggregation tier, each of which is in turn connected to the switches in the core tier. Although a 3-tier tree network is highly scalable and easy to deploy, it has significant switching and queuing delays due to its high hop count and the congestion points in its top two tiers. Furthermore, because of the low path diversity in tree topologies, applications running in a 3-tier tree network cannot avoid congestion points even when they are generating traffic with strong locality (i.e., between nearby racks).

We can significantly reduce the latency of a 3-tier tree network by replacing a portion of the edge and aggregation tiers with a Quartz tier, as illustrated in Figure 15(c). This configuration reduces the maximum hop count from five to three and the number of congestion points by effectively eliminating a tier from the network. Moreover, unlike in a 2-tier tree network, localized traffic that span multiple racks can be grouped into a single Quartz ring and can therefore avoid the large switching delay of the core switch. The full connectivity of the Quartz network also provides more freedom for application specific locality optimizations, which is important given that most datacenter traffic patterns show strong locality [30].

4.2 Quartz in the Core

Large DCNs that connect hundreds of thousands of servers require core switches with port count in excess of a thousand ports because of the need to interconnect several hundred aggregation switches. These switches are based on slower but more scalable store-and-forward designs, and have switching latencies that are an order of magnitude more than low-latency cut-through switches. Furthermore, they are generally very expensive, with a significant portion of the cost being the large chassis that connects the switch line cards. Therefore, although these switches provide modular scalability, the large upfront cost of the chassis means incorrect growth prediction is very costly.

To avoid the high latency and poor price scalability of current core switches, we explore a configuration that replaces core switches with Quartz, as illustrated in Figure 15(b). A Quartz network using low-latency switches has significantly lower switching delays than core switches. It also does not require an expensive upfront investment; switches and WDMs can be added as needed. A potential problem with replacing core switches with Quartz is that, unlike core switches, Quartz does not offer full bisection bandwidth. We evaluate the impact of this limitation using a pathological traffic pattern in Section 7.2.

4.3 Quartz in Random Topology Networks

Random topology networks, such as Jellyfish [38] and SWDC [37], offer an exciting new design point in DCNs. However, without any network structure, it is difficult for applications running in random topology networks to take advantage of traffic locality. Furthermore, much like mesh networks, random topology networks have high wiring complexity, which limits their scale.

To address these issues, we propose an alternative design to Jellyfish that, instead of creating a random graph of switches, creates a random graph of Quartz networks. This configuration enables applications to take advantage of traffic locality in the same way as we discussed in Section 4.1. Furthermore, by grouping nearby switches together into a Quartz network before connecting the Quartz networks together into a random graph, this configuration reduces the number of random connections and therefore greatly simplifies the DCN's wiring complexity.

4.4 Configurator

Datacenter providers must balance the gain from reducing end-to-end latency with the cost of using low-latency hardware. Unfortunately, as we have discussed earlier, latency arises from numerous factors, including datacenter size, network topology, traffic load, and switch type. Therefore, it is possible to only give approximate guidelines regarding the gain from introducing low-latency components into a DCN.

We make a 'best-effort' attempt to quantify the cost-benefit tradeoff of using Quartz in Table 8, which summarizes the cost of using

Quartz in various network configurations. We consider different datacenter sizes, ranging from 500 servers to 100,000 servers. For all of the tree-based configurations, we use cut-through switches [4] in the edge and aggregation tiers, and high-port density store-and-forward switches [9] in the core tier. We also use commercially available amplifiers [12], DWDMs [8], and transceivers [7]. To simplify the evaluation, the prices include all the hardware expenses except for the cost of the servers.

We take into account (at a very high level) the network utilization of a datacenter; we consider when the network's utilization is 'high,' which corresponds to a mean link utilization of 70%, and 'low,' which corresponds to a mean link utilization of 50%. We also investigate various network topologies including a two-tier tree, three-tier tree, a single Quartz ring, and the use of Quartz in the edge or core layer (or both). We present the approximate packet latency reduction from using Quartz based on our simulation results in Section 7 and the cost per server for these various network configurations.

We first analyse the use of Quartz for small datacenters, which have approximately 500 servers. We observe that the use of Quartz increases the cost per server by 7% compared to a two-tier tree structure. However, we can achieve a latency reduction of at least 33% in an environment with low network utilization and more than 50% with high network utilization.

In the case for medium-sized datacenters, which consists of 10,000 servers, we find that the use of Quartz increases the cost of the datacenter by 13% and reduces the datacenter's latency by 20% with low traffic, and more than 40% with high traffic. The cost of using Quartz is higher in a medium-sized datacenter than a small-sized datacenter because of the larger size of the Quartz ring needed to serve more servers; thus, more optical hardware is required.

Finally, we consider a large datacenter that contains 100,000 servers. We find that using Quartz at the core layer does not increase cost per server since the three-tier tree requires a high port density switch. As high port density switches are also expensive, their cost is similar to the optical hardware that is found in a Quartz ring. By replacing high port density switches with Quartz rings, we see a 70% improvement in latency with low traffic. We also consider the use of Quartz at the both edge and core layer in an environment with high network utilization. We see that the cost increases by 17% and latency is reduced by more than 74%.

To summarize, we realize that it is impossible to give exact cost-benefit tradeoffs due to the numerous sources of network latency. We demonstrate however that (a) Quartz can be used as a design element in many different standard DCNs (b) the additional (one-time) cost due to introducing Quartz is fairly small and (c) in all cases, using Quartz significantly reduces end-to-end latency.

5. Analysis

We analyze the properties of five representative network topologies (2-tier tree, Fat-Tree [17], BCube [25], Jellyfish [38] and mesh) and determine their suitability as a low-latency network design element. In this analysis, we configure each topology to mimic a single switch with approximately 1000 ports. We compute the path diversity of each topology using the metric defined in [39]. Note that Jellyfish's path diversity depends on both the chosen routing algorithm (k-shortest-path or ECMP) and the number of switch-to-switch links. We define wiring complexity as the number of cross-rack links. Table 9 shows a summary of their key properties.

Out of the five network topologies, the 2-tier tree structure requires the fewest switches to provide 1k usable ports and therefore has the lowest relative equipment cost. It is also the simplest structure to wire; each ToR switch only has a small constant number of connections to each of the second tier switches. However, as established by previous work [17], providing high bisection bandwidth in a tree network requires high port count second tier switches that are both expensive and, more importantly, have high latency. This problem, combined with its low path diversity, which can result in significant congestive delays, make 2-tier tree networks a poor choice for a low-latency design element.

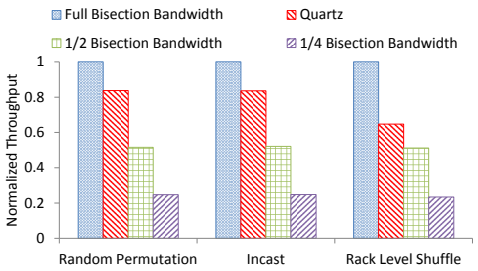


Figure 10: Normalized throughput for three different traffic patterns

By increasing path diversity, Fat-Tree, BCube, and Jellyfish have lower congestive delays and offer significantly more bisection bandwidth than 2-tier tree networks without requiring high port count switches. Fat-tree is the most expensive of the three structures, but provides full bisection bandwidth without requiring server-side packet switching. BCube's use of server-side packet switching results in the highest latency of the five topologies. All three systems have relatively high wiring complexity.

Finally, a mesh network offers the highest path diversity, lowest hop count and latency when using direct routing, and relatively high bisection bandwidth when using indirect routing with VLB. It also has relatively high wiring complexity, but by implementing it using a WDM ring, the wiring complexity can be simplified to be as low as a 2-tier tree network.

5.1 Bisection Bandwidth

Given Quartz's high path diversity, it is difficult to analytically calculate its bisection bandwidth. Instead, we use simulations to compare the aggregate throughput of a Quartz network using both one- and two-hop paths to that of an ideal (full bisection bandwidth) network for typical DCN workloads. We also compare Quartz's throughput with reduced capacity networks with 1/2 and 1/4 bisection bandwidth. We use the following common datacenter communication patterns in our comparison:

1. **Random Permutation Traffic**. Each server sends traffic to one randomly selected server, while at the same time, it receives traffic from a different randomly selected server.
2. **Incast Traffic**. Each server receives traffic from 10 servers at random locations of the network, which simulates the shuffle stage in a MapReduce workload.
3. **Rack Level Shuffle Traffic**. Servers in a rack send traffic to servers in several different racks. This represents traffic when the administrator is trying to balance the load between racks through VM migration. This load pattern is common in elastic datacenters, where servers are turned off at off peak hours.

Figure 10 shows the normalized throughput for three traffic patterns. The normalized throughput equals to 1 if every server can send traffic at its full rate. For random permutation traffic and incast traffic, Quartz throughput is about 90% of a full bisection bandwidth network. For rack level shuffle traffic, the normalized

	Network Latency without Congestion	# of 64-port Switches	Wiring Complexity	Path Diversity
2-Tier Tree	1.5μs (3 Switch Hops)	17	16	1
Fat-Tree	1.5μs (3 Switch Hops)	48	1024	32
BCube	16μs (2 Switch Hops & 1 Server Hop)	32	960	2
Jellyfish	1.5μs (3 Switch Hops)	24	240	$\leq$32
Mesh	1.0μs (2 Switch Hops)	33	528 32 (with WDMs)	32

Table 9: Summary of different network structures with 1k servers

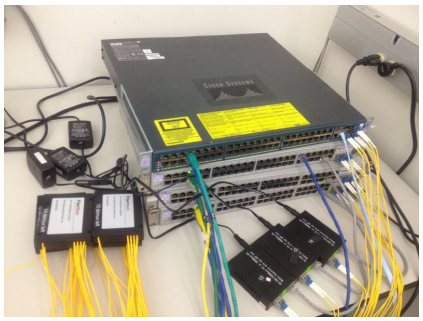


Figure 11: Quartz with 4 switches, connected by WDMs

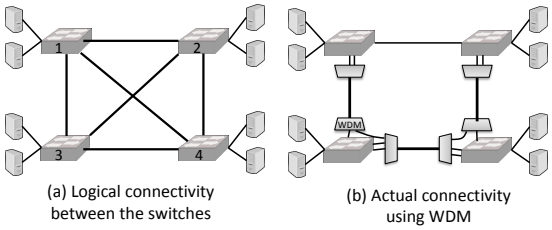


Figure 12: Topology of our testbed.

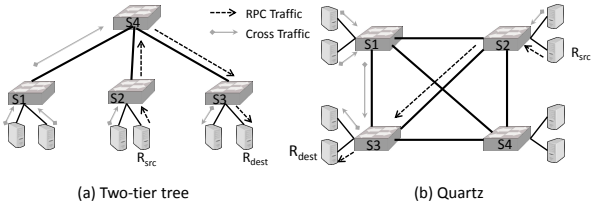


Figure 13: Traffic flows of cross-traffic experiment

throughput is about 0.75. We conclude that Quartz's bisection bandwidth is less than full bisection bandwidth but greater than 1/2 bisection bandwidth. Overall, Quartz provides significantly higher throughput than the other oversubscribed network topologies for all three traffic patterns.

6. Prototype

In this section, we validate the Quartz design by building a small prototype, and determine the relative performance difference between Quartz and a 2-tier tree in a simple cross-traffic experiment. Our prototype consists of 4 commodity switches and 8 servers, as illustrated in Figure 11. Three of the switches are Nortel 5510-48T, and the fourth is a Cisco Catalyst 4948. All four are 1 Gbps managed switches with 48 ports. Figure 12(a) shows the logical connectivity between the switches. By using WDM muxes/demuxes, we can simplify the network cabling to Figure 12(b), where there are only optical links between nearby switches in a ring. The network has 8 servers in total and 4 links between any bisection of the network, which means it can provide full bisection bandwidth. Our prototype has 12 CWDM SFP transceivers which support 1.25Gbps bidirectional communications. Among the transceivers, 8 of them use the 1470nm wavelength band, 2 of them use the 1490nm band, and the remaining 2 use the 1510nm band. We also use 4-channel CWDM muxes/demuxes in our prototype to multiplex different channels into one optical fibre cable. Signal amplifiers are unnecessary in this small testbed. We actually need to use attenuators to protect the receivers from overloading.

Each server is equipped with a 1 Gbps Intel Pro/1000 GT Desktop Adapter and is running Ubuntu 11.10 without any low-latency OS or network stack changes. In order to isolate the impact of the network architecture on latency, we present relative rather than absolute performance results while maintaining the same software and hardware configurations throughout our experiments.

To precisely control the traffic paths in our experiments, we use the technique introduced in SPAIN [35] to expose alternative network paths to the application. We create 4 virtual interfaces on each server, where each virtual interface sends traffic using a specific VLAN and the spanning trees for the VLANs are rooted at different switches. Therefore, an application can select a direct two-hop path or a specific indirect three-hop path by sending data on the corresponding virtual interface.

6.1 Impact of Cross-Traffic

We evaluate the performance of Quartz and a 2-tier tree topology using a cross-traffic workload. In order to ensure a fair comparison, we use the prototype described in Section 6 for evaluating Quartz, and rewired the switches from the Quartz prototype into a 2-tier tree (1 aggregation and 3 ToR switches) to perform our 2-tier tree experiments. Each ToR switch is connected to two servers and we use six total servers in our experiments.

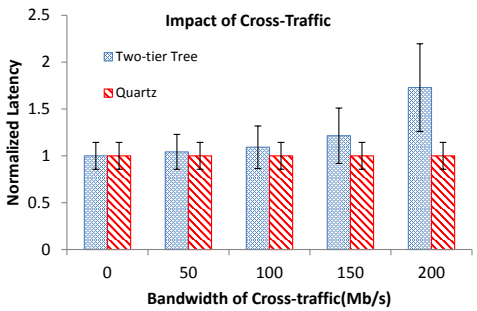
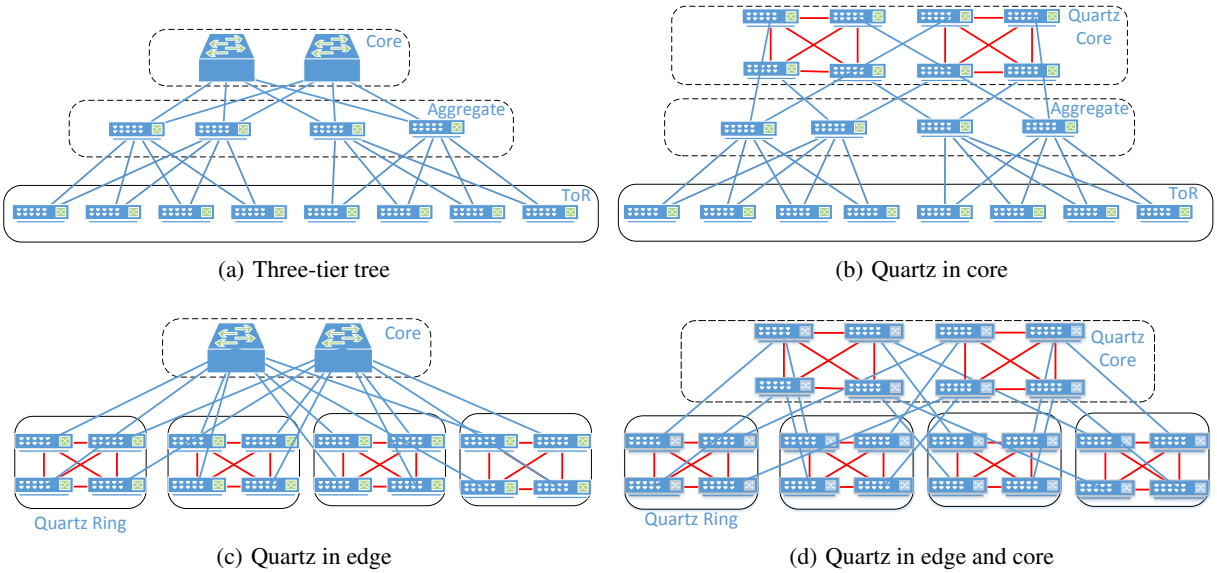


Figure 14: Impact of cross-traffic on different topologies



(a) Three-tier tree

(b) Quartz in core

(c) Quartz in edge

(d) Quartz in edge and core

Figure 15: Simulated network topologies. Jellyfish and Quartz in Jellyfish topologies are not shown..

Our cross-traffic workload consists of a "Hello World" RPC written in Apache Thrift [14] between two servers (R_{src} and R_{dst}) connected to different ToR switches (S_2 and S_3), which represents a latency sensitive flow, and additional bursty traffic generated using Nuttcp [15] from three servers connected to S_1 and S_2 to a server connected to S_3. Figure 13 illustrates the topologies and the different flows. Note that we only use direct paths and we do not use the servers connected to S_4 in this experiment. To minimize OS scheduling related uncertainties, each server is only involved with one flow. The RPC application executes $10,000$ RPC calls one at a time for each experiment, and the bursty cross-traffic consist of 20 packet bursts that are separated by idle intervals, the duration of which is selected to meet a target bandwidth. The bursty traffic from the three servers are not synchronized. We perform 100 runs of each experiment and show the 95% confidence interval as error bars.

Figure 14 shows that, as we increase the cross-traffic from 0 to 200 Mbps (0 to 20% of the link bandwidth), the RPC latency rapidly increases for the tree topology due to congestion. At 200 Mbps, the RPC latency for the tree topology increases by more than 70% compared to its latency without cross-traffic. In contrast, the RPC latency is unaffected by cross-traffic with Quartz. These results corroborate with our analysis in Section 5, and demonstrate the impact of the network topology on reducing network congestion from cross-traffic.

7. Simulation Study

In order to evaluate the performance of Quartz and other topologies at scale, we implemented our own packet-level discrete event network simulator that we tailored to our specific requirements. We have performed extensive validation testing of our simulator to ensure that it produces correct results that match queuing theory. We make the simplifying assumptions that servers send 400-byte packets according to a Poisson process. We model two state-of-the-art switches in our simulator:

- Cisco Nexus 7000 core switch (CCS)
- Arista 7150 ultra low latency switch (ULL)

The specifications of these switches are summarized in Table 16. In our simulated architectures, we use ULL for both ToR switches and aggregation switches, and CCS as core switches. We use ULL exclusively in Quartz. Each simulated Quartz ring consists of four switches; the size of the ring does not affect performance and only affects the size of the DCN. We implement the following network architectures in our simulator:

1. **Three-tier Tree (Figure 15(a)):** A basic three-tier tree structure with each ToR switch connected to two aggregation switches over 40Gbps links, and each aggregation switch connected to two core switches over 40Gbps links.
2. **Quartz in Core (Figure 15(b)):** Each core switch is replaced with a Quartz ring. The aggregation switches connect to the Quartz ring over 40 Gbps links.
3. **Quartz in Edge (Figure 15(c)):** The ToR and aggregation switches are replaced with Quartz rings. Servers connect to the Quartz rings using 10 Gbps links, and the Quartz rings connect to the core switches using 40 Gbps links.
4. **Quartz in Edge and Core (Figure 15(d)):** Both core and edge layers are replaced with Quartz rings.
5. **Jellyfish:** A random topology consisting of 16 ULL, with each switch dedicating four 10 Gbps links to connect to other switches.
6. **Quartz in Jellyfish:** A random topology consisting of four Quartz rings, with each Quartz ring dedicating a total of four 10 Gbps links to connect to switches in the other rings.

7.1 Results

We evaluate the performance of the different topologies using three common traffic patterns:

- **Scatter:** One host is the sender and the others are receivers. The sender concurrently sends a flow of packets to the receivers.

Switch	Latency	Port Count
Cisco Nexus 7000 (CCS)	6 us	768 10Gbps or 192 40Gbps
Arista 7150S-64 (ULL)	380 ns	64 10Gbps or 16 40Gbps

Table 16: Specifications of switches used in our simulations.

- **Gather:** One host is the receiver and the others are senders. The senders concurrently send a flow of packets to the receiver.
- **Scatter/Gather:** One host sends packets to all the other hosts, then all the receivers send back reply packets to the sender.

These traffic patterns are representative of latency sensitive traffic found in social networks and web search [19], and are also common in high-performance computing applications, with MPI [24] providing both scatter and gather functions as part of its API. With these workloads, we are primarily interested in determining the impact of cross-traffic on latency, where cross-traffic is generated by running multiple instances of these traffic patterns at the same time. Note that we do not show Quartz using ECMP and VLB separately. This is because there is negligible performance difference between the two protocols when using these traffic patterns.

Figure 17 shows the average latency of each scatter, gather, or scatter/gather operation in which the senders and receivers are randomly distributed across servers in the network. Both the scatter and the gather workloads show that the three-tier tree introduces significant latency even with only a single task where there is minimal congestion. Most of this latency is from the high-latency core switch. There is also an approximately linear increase in latency with additional tasks for the three-tier tree. Performing both scatter and gather exacerbates the problem with the three-tier tree exhibiting both a higher linear increase in latency with additional tasks, and a substantial jump in latency going from three to four tasks. This latency jump is due to link saturation from an oversubscribed link.

Using Quartz in the edge reduces the absolute latency compared to the three-tier tree even with only one task. This is due to the additional paths between servers in the same ring that avoid the core switch. More importantly, latency is mostly unaffected by adding additional scatter or gather tasks, which can be attributed to the high path diversity of the Quartz ring. Introducing additional scatter/gather task does increase the latency of Quartz in the edge, although at a lower rate than the three-tier tree.

As we had expected, the main performance benefit from using Quartz in the core is a reduction in the absolute latency of the core tier. There is more than a three microsecond reduction in latency by replacing the core switches in a three-tier tree with Quartz rings.

Using Quartz in both the edge and core reduces latency by nearly half compared to the three-tier tree. The latency reduction comes from a combination of a reduction in hop-count, and a significantly lower latency core tier.

Jellyfish and Quartz in Jellyfish perform almost identically for these traffic patterns. Therefore, we omit the Quartz in Jellyfish line to improve the clarity of the graphs. These random networks exhibit low latency due to their relatively low average path length and high path diversity. They have a similar response to an increase in cross-traffic as Quartz in the core, with a slightly lower absolute latency. However, these results are, in part, due to the small simulated network size. For networks that are one or two orders of magnitude larger, we would expect a small increase in path length that would increase the absolute latency by a few microseconds. In

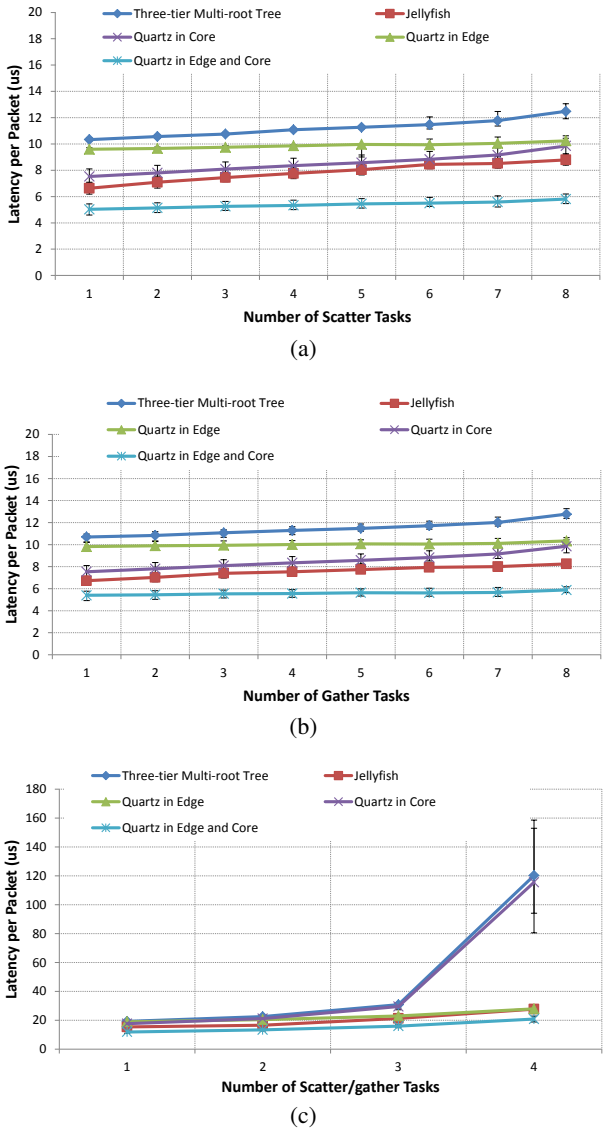
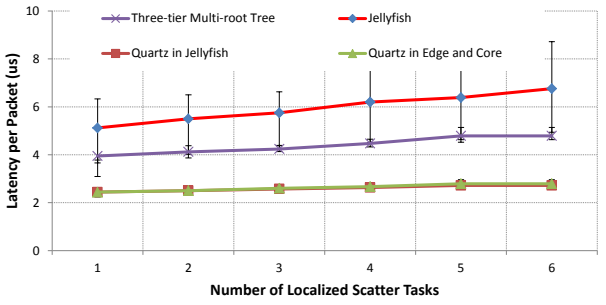


Figure 17: Average latency comparison for global traffic pattern. This graph is best viewed in colour.

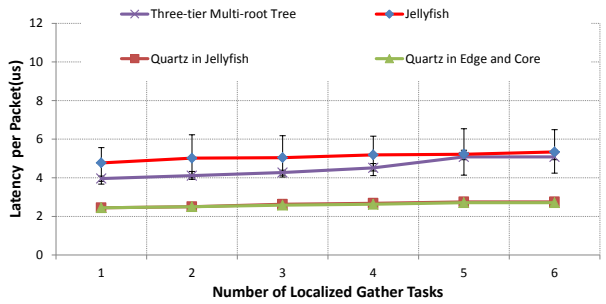
contrast, the other network topologies can support these larger network sizes without an increase to their path lengths. Furthermore, Jellyfish's random topology is especially well-suited for handling globally distributed traffic patterns. Therefore, we next look at traffic patterns that exhibit strong locality.

Figure 18 shows the average latency of a local task, that is, a task that only performs scatter, gather, or scatter/gather operations between servers in nearby racks. There is only one local task per experiment; the remaining tasks are have randomly distributed senders and receivers and are used to generate cross-traffic. The three-tier tree has significantly lower latency as the local task traffic does not have to traverse the core tier. However, it still exhibits a linear increase in latency with additional tasks.

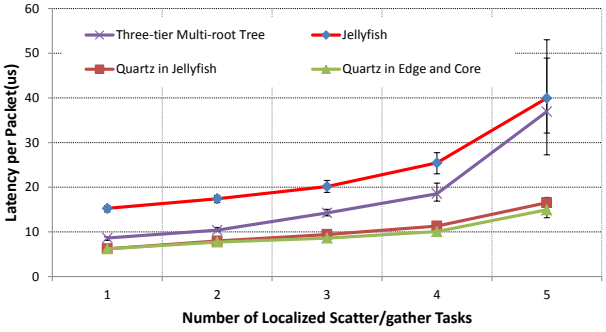
Jellyfish has the highest latency for these traffic patterns because it is unable to take advantage of the traffic locality. Note that in our experiments, the local task performs scatter, gather, operations to fewer targets than the non-local tasks. This accounts for the slight



(a)



(b)



(c)

Figure 18: Average latency comparison for localized traffic pattern. This graph is best viewed in colour.

reduction in latency for Jellyfish's local task compared to its non-local tasks.

By using Quartz in the edge or as part of Jellyfish, there is a significant reduction in latency for the local task. Traffic from the local task remain within the Quartz ring, and because of Quartz's high path diversity, these topologies are mostly unaffected by cross-traffic. We only see an increase in latency when increasing the number of scatter/gather tasks.

7.2 Pathological Traffic Pattern

Replacing a core switch with a Quartz ring significantly reduces latency and, for smaller networks with plans for growth, avoids the upfront cost of purchasing a large, expensive, and mostly empty core switch chassis. However, a Quartz ring does not provide full bisection bandwidth, which can impact performance for certain workloads. In this section, we compare the performance of a non-blocking core switch to that of using Quartz in the core. We use a simple pathological traffic pattern that, when used on Quartz, sends multiple flows of traffic from different ports on switch S_1 to

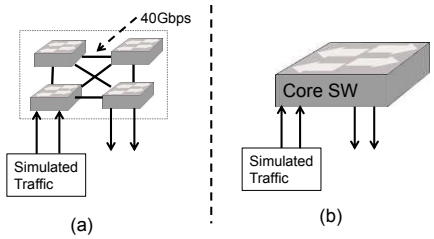


Figure 19: Pathological traffic pattern.

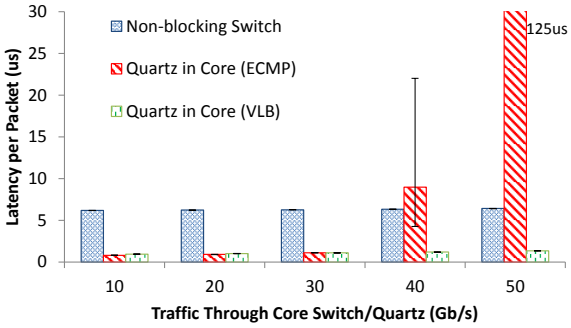


Figure 20: Average latency comparison for pathological pattern.

multiple receivers connected to switch S_2, which stresses switch-to-switch bandwidth. Our Quartz ring consists of four $40GbE$ switches logically connected as shown in Figure 19(a) and we use a standard non-blocking core switch as shown in Figure 19(b) for comparison.

Figure 20 shows the packet latency of these flows as we increase the aggregate flow bandwidth. As expected, the non-blocking core switch is unaffected by the competing flows, but introduces a significant amount of switch latency because of its store-and-forward design. Using Quartz in the core with ECMP routing, which only uses direct paths, offers significantly lower latency than the core switch until it saturates the link bandwidth between the source and destination switch. Beyond saturation, the packet latency of these flows becomes unbounded. Using Quartz in the core with VLB routing, which uses both direct and indirect paths, the latency is essentially equivalent to using ECMP routing for the low traffic experiments. Even with 50 Gbps aggregate traffic, there is no noticeable increase in packet latency when performing VLB routing.

8. Discussion

Our work is motivated by the requirements of latency-sensitive applications in DCNs. Existing approaches to reduce latency are primarily based on protocol changes, such as using explicit congestion signals to reduce the number of dropped packets [19, 43], perform network-wide flow scheduling [18], or offer strict resource reservations [29]. Instead, we make the observation that, similar to optical mesh networks in the wide area, commodity WDMs allow an O(n^2) mesh to be implemented as an O(n) physical ring in a datacenter.

This observation could not be put into practice earlier due to the high cost of optical WDMs. However, due to the recent large-scale rollout of fiber-to-the-home, optical part costs have been dropping exponentially, making their use feasible as replacements for electronic switches. Although Quartz still carries a price premium over traditional DCNs, we expect the price difference will diminish as WDM shipping volumes continue to rise. Some DCN operators

may prefer to invest the price premium into higher bisection bandwidth instead of lower network latency. However, for applications that require ultra low latency, our analysis shows that there is no cost-effective alternative that can realistically be deployed in a modern datacenter. More importantly, we believe our work paves the way for future innovative architectures that integrate low-cost commodity WDM hardware into DCNs.

The main limitation of a WDM mesh ring is its limited scalability. We have turned this limitation into an advantage by making Quartz a design element that can replace *portions* of current DCNs. Quartz, by design, is completely backwards compatible with existing DCNs. It can be incrementally deployed as needed to cut latency in portions of DCNs, or to allow incremental deployment of a core switch. Moreover, if port count of low-latency cut-through switches increase, Quartz becomes more scalable.

9. Conclusions

Drawing on the requirements of latency-sensitive applications such as high-performance computing, we present an *architectural approach* to latency reduction. Our key breakthrough is to present Quartz, a logical O(n^2) mesh implemented as a physical O(n) ring using inexpensive commodity Wavelength Division Multiplexers. Although current port counts of low-latency switches limit the scale of a single Quartz network, we demonstrate that Quartz can be used to incrementally replace latency-sensitive portions of standard DCN topologies to cut end-to-end latency between 33% to 50%.

Compared to existing topologies, Quartz has three advantages: (a) least possible hop count, so minimal latency (b) elimination of queuing latency due to cross traffic and (c) if necessary, the ability to use two-hop paths to achieve high bisection bandwidths, at the expense of slightly higher latency. Quartz is completely legacy-compatible: its benefits can be obtained *without* having to replace current DCNs.

We have evaluated our ideas using analysis, simulations, and a small prototype. We find that Quartz can significantly reduce latency. In particular, we found that using Quartz in both the core and edge can reduce latency by 50% in typical scenarios.

Acknowledgment

We thank our shepherd Jitendra Padhye and the anonymous reviewers for their insightful comments. This work is supported by the Natural Sciences and Engineering Research Council of Canada, and the Canada Research Chairs Program.

10. References

[1] Data Center Multi-Tier Model Design. *Cisco Data Center Infrastructure 2.5 Design Guide*, December 2007. http://bit.ly/11usMXi.

[2] Acmemicro - Arista 7148SX, December 2013. http://bit.ly/14xquvm.

[3] Arista 7100 Series. December 2013. http://bit.ly/VBo1Ll.

[4] Arista 7150 Series. December 2013. http://bit.ly/PUH7uD.

[5] Cisco Catalyst 4948 Switch. December 2013. http://bit.ly/VtlIbf.

[6] Quanta T3048-LB8, December 2013. http://bit.ly/173tPpz.

[7] 10Gbps 10GHz DWDM SFP Single Mode 40km Optical Transceivr, May 2014. http://bit.ly/1hAwCXP.

[8] 80 channels, 2RU Rack Mount, Duplex, Athermal AWG, DWDM Mux & Demux, May 2014. http://bit.ly/1iaUSzG.

[9] Cisco Nexus 7700 Switches. May 2014. http://bit.ly/SZGzs2.

[10] Fixed Flange E2000/APC Fiber Optic Attenuator 1 to 30dB optional, May 2014. http://bit.ly/1r0skSW.

[11] Optical Impairment-Aware WSON Control Plane for Cisco ONS 15454 MSTP Data Sheet, May 2014. http://bit.ly/RC8Ljo.

[12] OpticTrans DWDM 80 Channel Amplifier / EDFA / Fiber Optic Amplifier, June 2014. http://bit.ly/TxiMjg.

[13] Storm: Distributed and fault-tolerant realtime computation, January 2014. http://storm-project.net/.

[14] Website for Apache Thrift, January 2014. http://thrift.apache.org/.

[15] Website for Nuttcp, January 2014. http://bit.ly/1bDluG9.

[16] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, and A. Donnelly. Symbiotic routing in future data centers. In *Proceedings of SIGCOMM*, New Delhi, India, August 2010.

[17] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proceedings of SIGCOMM*, Seattle, Washington, August 2008.

[18] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *Proceedings of NSDI*, San Jose, California, April 2010.

[19] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center TCP (DCTCP). In *Proceedings of SIGCOMM*, New Delhi, India, August 2010.

[20] J. E. Berthold. Optical Networking for Data Center Interconnects Across Wide Area Networks. In *Proceedings of the Symposium on High-performance Interconnects*, New York, New York, August 2009.

[21] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM Magazine*, January 2008.

[22] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy. Routebricks: Exploiting parallelism to scale software routers. In *Proceedings of SOSP*, Big Sky, Montana, October 2009.

[23] N. Farrington and A. Andreyev. Facebook's data center network architecture. 2013. http://bit.ly/1mABW01.

[24] I. Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Boston, MA, Feburary 1995.

[25] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: a high performance, server-centric network architecture for modular data centers. In *Proceedings of SIGCOMM*, Barcelona, Spain, August 2009.

[26] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. DCell: a scalable and fault-tolerant network structure for data centers. In *Proceedings of SIGCOMM*, Seattle, Washington, August 2008.

[27] C. Hong, M. Caesar, and P. Godfrey. Finishing flows quickly with preemptive scheduling. In *Proceedings of SIGCOMM*, Helsinki, Finland, August 2012.

[28] N. S. Islam, M. W. Rahman, J. Jose, R. Rajachandrasekar, H. Wang, H. Subramoni, C. Murthy, and D. K. Panda. High performance rdma-based design of hdfs over infiniband. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, Salt Lake City, Utah, November 2012.

[29] N. Jiang, D. Becker, G. Michelogiannakis, and W. Dally. Network congestion avoidance through speculative reservation. In *Proceedings of the HPCA*, New Orleans, Louisiana, Feburary 2012.

[30] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. the nature of data center traffic: Measurements & analysis. In *Proceedings of the IMC*, Chicago, Illinois, November 2009.

[31] R. Kapoor, G. Porter, M. Tewari, G. Voelker, and A. Vahdat. Chronos: predictable low latency for data center applications. In *Proceedings of the Symposium on Cloud Computing(SoCC)*, San Jose, CA, October 2012.

[32] H. Litz, H. Froning, M. Nuessle, and U. Bruning. A HyperTransport Network Interface Controller for Ultra-low Latency Message Transfers. *Hyper Transport Consortium*, Feburary 2008.

[33] H. Liu and R. Zhang-Shen. On direct routing in the valiant load-balancing architecture. In *Proceedings of the GLOBECOM*, St. Louis, Missouri, November 2005.

[34] S. Miller. *Optical fiber telecommunications*. Elsevier, 1979.

[35] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul. Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In *Proceedings of NSDI*, San Jose, CA, April 2010.

[36] S. M. Rumble, D. Ongaro, R. Stutsman, M. Rosenblum, and J. K. Ousterhout. It's time for low latency. In *Proceedings of HotOS*, Napa, California, May 2011.

[37] J.-Y. Shin, B. Wong, and E. G. Sirer. Small-world datacenters. In *Proceedings of the ACM Symposium on Cloud Computing*, Cascais, Portugal, October 2011.

[38] A. Singla, C. Hong, L. Popa, and P. Godfrey. Jellyfish: Networking data centers randomly. In *Proceedings of NSDI*, San Jose, California, April 2012.

[39] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker. Characterizing and measuring path diversity of internet topologies. June 2003.

[40] B. Vamanan, J. Hasan, and T. Vijaykumar. Deadline-aware datacenter tcp (D2TCP). In *Proceedings of SIGCOMM*, Helsinki, Finland, August 2012.

[41] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron. Better never than late: meeting deadlines in datacenter networks. In *Proceedings of SIGCOMM*, Toronto, Canada, August 2011.

[42] H. Zang and J. P. Jue. A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *Springer's Optical Networks Magazine*, January 2000.

[43] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz. DeTail: Reducing the flow completion time tail in datacenter networks. In *Proceedings of SIGCOMM*, Helsinki, Finland, August 2012.