Leveraging Commodity Photonics to Reduce Datacenter Network Latency

by

Yunpeng Liu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2014

Abstract

Most datacenter network (DCN) designs focus on maximizing bisection bandwidth rather than minimizing server-to-server latency. They are, therefore, ill-suited for important latency-sensitive applications, such as high performance computing, realtime analytic systems and high-frequency financial trading. Although there are a number of existing approaches to reduce network latency, they are only partially effective, workload dependent, and often require network protocol changes.

In this thesis, we explore architectural approaches to building a low-latency DCN and introduce Quartz, a new optical design element consisting of a full mesh of switches connected by an optical ring. We can reduce the network latency of a hierarchical or random network by replacing portions of it with a Quartz ring. Our analysis shows that, in a standard 3-tier DCN, 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 by 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.

Acknowledgements

I would like to thanks all the people who helped me on my thesis. And I would like to express my gratitude to my supervisor Prof. Bernard Wong for the invaluable comments, kind advices and constant encouragement. I would also like to thank Prof. S. Keshav and Prof. Martin Karsten for being my thesis reader. Finally, I want to thank all my friends from Shoshin and the other labs.

Dedication

This is dedicated to Sophia and my parents for their constant support and understanding. This is also dedicated to Prof. Bernard Wong and Prof. S. Keshav for their trust and patience during my graduate studies.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Datacenter network (DCN) bandwidth has increased dramatically over the past decade. In contrast, DCN latency has reduced at a much slower rate. For example, network latency has only had a 30x reduction over last 30 years, while network bandwidth has improved by more than $3,000$x over the same period [38]. The focus on increasing network bandwidth is driven by the needs of distributed batch computation frameworks with any-to-any communication patterns, such as MapReduce [19] and Hadoop [6].

Nevertheless, network latency has become a critical factor in determining the performance of many datacenter-scale, distributed applications. In most distributed real-time or parallel computation frameworks such as Storm [9] and MPI [23], network latency manifests as a substantial component of coordination delay and is therefore on the critical path. Furthermore, interactive real-time analytics on Big Data sources must respond quickly to complex user queries. Even a small increase in latency can impact the user experience for these services, which can result in a significant reduction in revenue [44].

Past work on low-latency DCNs has primarily focussed on reducing packet processing delays in the OS network stack [32], network interface card (NIC) [33] and network switches [3], and avoiding network congestion through bandwidth reservations [30] or congestion-aware flow scheduling [15, 28, 43, 46, 49]. 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, although state-of-the-art low-latency cut-through switches are already available commercially from companies such as Arista and Mellanox [3], they are limited in scalability compared to standard store-and-forward switches. Current cut-through switches are limited to 64 ports compared to 1024 ports for store-and-forward switches. Therefore, large DCNs must choose to either deploy standard high latency and high port density 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. Moreover, a multi-tier network architecture, even one using cut-through switches, is still highly suspectable to latency spikes due to the congestion focal points at the aggregation and core switching tiers. Even small, infrequent bursts of cross-traffic can lead to highly unpredictable network latencies. As a result of these limitations, current latency-sensitive applications are forced to use high-cost dedicated networks in order to consistently meet their latency requirements.

Similarly, congestion avoidance techniques are often only effective for specific workloads and deployment scenarios. For example, congestion signal-based techniques mainly reduce the number of dropped packets [15, 49] and only minimally reduce end-to-end delays. Network-wide flow scheduling [14] techniques are unable to handle a large volume of short flows. Finally, resource reservation techniques [30] require appropriate knowledge of flow size and timing, which in practice can lead to significant resource underutilization.

1.1 Quartz

In this thesis, we explore an *architectural approach* to reduce 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 density core switch or an aggregation layer of switches. Unfortunately, a full mesh topology has significant wiring complexity and does not scale well. We address the wiring complexity problem by exploiting commodity photonics in the form of optical wavelength division multiplexers (WDMs) to implement a complex $\mathrm{O}(n^2)$ mesh as a simple $\mathrm{O}(n)$ optical ring. This 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 1.1 [17]).

The wiring complexity of a full mesh with n nodes is $O(n^2)$, since we need a pairwise direct connection between any pair of ToR switches. However, a single optical cable using WDM in Quartz can carry multiple wavelengths of light ('colours' or 'channels'). By 'splitting off' the right colours from the Quartz ring at each ToR switch, we can set up dedicated paths between any ToR switches, but with only *two* cables leaving each ToR switch. Therefore, an $\mathrm{O}(n^2)$ mesh is implemented by a simple $\mathrm{O}(n)$ optical ring in Quartz.

Compared to other topologies, Quartz has the lowest network diameter and, when paired with Valiant Load Balancing (VLB), the highest path diversity. However, the scalability of Quartz is limited by the size of the switches and the channel capacity of the

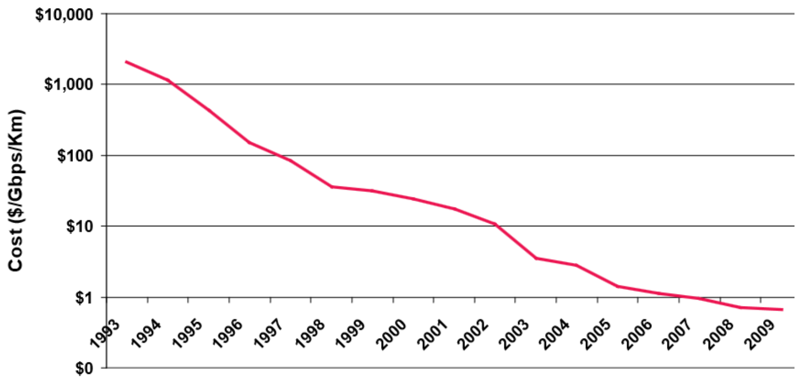


Figure 1.1: Backbone Dense WDM per-bit, per-km cost improvements over time [17].

optical fiber. To deal with its limited scalability, we envision that Quartz can be used to replace *portions* of traditional DCNs.

Traditional hierarchical networks, such as a 3-tier tree or fat-tree [13], are scalable and easy to deploy. However, each switching tier introduces two extra switching hops and additional congestion points. Consequently, we first explore replacing the ToR and aggregation tiers with a single Quartz tier to significantly reduce the impact of cross-traffic. We also explore replacing the core-tier switches with a Quartz ring. In a standard tree network structure, the core tier must utilize high-port density store-and-forward switches due to the limited port count of cut-through switches. By replacing the core switches with Quartz using cut-through switches, we can significantly reduce the core-tier switching latency.

Random topology networks [39, 40] are a new design point in DCNs that provide low network diameter and high path diversity. Nevertheless, their randomness increase wiring complexity, which makes them difficult to deploy in a datacenter. To take advantage of this new design point, we explore grouping nearby switches together into a Quartz ring, and then connecting the Quartz rings together to form a random graph.

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. Results from both our simulation and prototype show that replacing portions of a DCN with Quartz can significantly lower the DCN's network latency. We also evaluate the cost of Quartz and show that it is cost competitive with traditional networks in many deployment scenarios. Moreover, we expect the cost of our solution to reduce over time as WDM shipping volumes rise.

1.2 Contributions

We make the following contributions in this thesis:

- We present a novel design element called Quartz consisting of a WDM-based full mesh to reduce communication latency in DCNs.
- We show how Quartz can be used in different topologies to reduce latency and congestion.
- We validate Quartz by building a small working prototype.
- We evaluate Quartz's performance using both our prototype and a discrete-event simulator. We find that Quartz can reduce network latency by more than 50% in many scenarios.

1.3 Thesis Organization

The remainder of this thesis is structured as follows: Chapter 2 provides the background and related work for this thesis. Chapter 3 describes the detailed architecture of Quartz and explores using Quartz as a design element for DCNs. Chapter 4 shows our experimental results comparing Quartz with other DCNs. Chapter 5 concludes this thesis.

Chapter 2

Background and Related Work

In this chapter, we first discuss several types of applications that require low network latency, then describe different sources of network latency and previous work that addresses each source. Finally, we introduce the optical network devices that are used in Quartz and outline related optical DCN proposals.

2.1 Latency-Sensitive Applications

In this section, we describe three common latency-sensitive application types.

High-performance computing applications typically require extremely low inter-server communication delays. Most non-trivial high-performance computing applications require significant coordination between different cluster nodes. Communication techniques such as MPI [23] and Remote Direct Memory Access (RDMA) were developed specifically to reduce communication latency for these applications. Communication delays reduce speedup and therefore limit overall application performance.

Big-memory distributed data-storage systems [21,47] provide a platform for real-time data analytics applications [9] that require extremely high transaction rates with strict latency limits per transaction. Reducing network latency is especially critical in big-memory data-storage systems that offer distributed transactions, because lower network latency can lower the transaction abort rate by reducing the duration of each transaction.

Financial applications, such as those that perform high-frequency algorithmic trading, are a critical element in today's global financial markets. These applications typically

Component	Standard	State of Art
OS Network Stack	15μs [38]	1 - 4 μs [32]
NIC	2.5 - 32μs [38]	0.5μs [33]
Switch	6 μs [4]	0.5μs [3]
Congestion	50μs [32]	

Table 2.1: Network latencies of different network components.

receive data directly from stock exchanges and use proprietary algorithms to perform trades every few microseconds, with projections that nanosecond-scale trading will soon be possible [27, 42]. Different trading strategies have varying latency requirements, ranging from hundreds of milliseconds for low frequency alpha trading to several microseconds for latency arbitrage.

Financial applications already use several techniques to reduce network latency between the datacenter and the stock exchange, such as collocating trading servers within exchange buildings and using direct line-of-sight fiber optic links from the exchange to the datacenter. In order to fully capitalize on high speed trading, inter-server communication delays within the same datacenter also need to be minimized. The need for low server-to-server latency is driven by the increasing sophistication of the trading algorithms, which requires computation to be distributed across multiple servers.

2.2 Sources of Latency

There are many sources of latency in DCNs (see Table 2.1 for a summary). In this section, we briefly discuss several sources including the network stack, network interface cards, switch latency, congestion and topologies. We also describe past work that addresses these latency sources.

2.2.1 Network Stack

A packet received by a Network Interface Card (NIC) from the network is first processed by the operating system kernel before being passed to user space. Therefore, 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. Kernel bypass allows user space applications to communicate directly with NICs, eliminating additional context switching. In zero-copy processing, data can be moved from the read buffer to the send buffer without requiring additional processing or memory transfers, which reduces processing latency. In recent work, the Chronos [32] system uses these techniques to significantly reduce the operating system kernel latency for datacenter applications. Another novel framework netmap [37] also enables fast packets I/O by preallocating resources, sharing buffers and metadata between kernel and userspace.

2.2.2 Network Interface Cards

Commodity NICs can introduce tens of microseconds of latency from performing basic buffering and packet processing. Recent work shows that by offloading packet processing to an FPGA and optimizing the communication between the FPGA and the host processor, NIC latency can be reduced to hundreds of nanoseconds [33]. Alternatively, Infiniband can be used in place of Ethernet to achieve lower NIC latency. Infiniband supports Remote Direct Memory Access (RDMA), which allows the NIC to bypass the operating system kernel to directly access memory. Support for RDMA over Ethernet [29] has recently been introduced by vendors such as Myricom, Solarflare, and Intel. With increasing demand for low-latency NICs from latency-sensitive applications, commodity NICs with RDMA support may be available in the near future.

2.2.3 Switch Latency

Switching delay is a significant source of network latency. For example, the Cisco Catalyst 4948 10 Gigabit Ethernet Switch, which is representative of current datacenter network switches, 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 forward a frame before the whole frame has been received. Low-latency switches, such as the Arista 7100 [3], have a switching delay of approximately 500 ns. Cut-through switches command only a relatively small price premium over the same port density store-and-forward switches [2,8]. Unfortunately, they are limited in scale (currently support up to 64 ports) compared to standard store-and-forward switches (currently support more than 1000 ports), and are therefore mainly used as ToR or aggregation switches rather than core switches.

2.2.4 Congestion

Although current datacenter networks support high bisection bandwidth, 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.

Some of these techniques aim to reduce congestion by introducing some changes to TCP. For example, DCTCP [15] utilizes Explicit Congestion Notifications (ECN) to ensure that queues do not overflow. 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 [43] and PDQ [28] use distributed congestion avoidance algorithms to manage flows at end-hosts and switches respectively. DeTail [49] reduces network latency by detecting congestion and selecting alternative uncongested paths to reduce queuing delay. Deadline Driven Delivery (D^3) [46] is a 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 network underutilization.

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

2.2.5 Topology

Network topology is a critical factor in determining a DCN's equipment and deployment cost, wiring complexity, scalability, and performance characteristics (bisection and end-to-end latency). In this section, we outline the latency characteristics of different topologies in turn.

Tree Networks: Tree topologies, such as the standard multi-root tree structure [1] and Fat-Tree [13], organize the network into multiple switching 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. Even for Fat-Tree, where there is abundant 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 [14].

Server-Centric Networks: DCell [26], BCube [25] and CamCube [12] are networks that use servers as switches to assist in packet forwarding. 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 [12, 39].

Randomized Networks: SWDC [39] and Jellyfish [40] 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 $\mathrm{O}(n^2)$ connections requirement greatly limits scalability and increases wiring complexity.

2.3 Optical Network Technologies

A number of recent DCN proposals [18, 22, 45] include various optical network elements, such as fiber-optical cables and optical switches, in their design. Optical fiber permits transmission over longer distances, with higher speed and even lower energy consumption, than electrical cables. For instance, current state-of-art optical fibers support transmission at up to $400 Gb/s$ [24] over a single channel, and each optical fiber can carry many independent channels. Quartz uses the following optical equipment:

1. **Optical Transceiver**: An optical transceiver is an active device that converts electrical signals to/from optical signals.

2. **Wavelength Division Multiplexer/Demultiplexer (WDM)**: A WDM consists of a multiplexer and a demultiplexer. The multiplexer combines optical signals of different wavelengths into a single fiber for transmission. Conversely, the demultiplexer splits a WDM signal into different single wavelength channels, with each channel on a separate fiber cable.

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 1.1 (taken from [17]) shows that the cost of 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.

2.4 Related Optical Proposals

Current optical proposals are trying to increase bandwidth between specific nodes to address the needs of elephant flows. For example, C-through [45] and Helios [22] propose to solve the bandwidth problem in over-subscribed datacenters by augmenting electrical networks with optical connections. Optical links are added to each of the ToR switches and these optical links are connected to a MEMS optical circuit switch (OCS). An OCS switch can be reconfigured at real-time to build an optical circuit connection between two links. OSA [18] proposes a pure optical datacenter architecture. Each ToR switch has several optical links to the OCS switch. By reconfiguring the OCS switch, OSA can make online network topology changes, allowing on-demand bandwidth changes between racks. Unfortunately, all of these optical proposals require OCS switches, which increase latency and limit scalability. Most OCS switches require tens of milliseconds to reconfigure an optical circuit, although there has been recent work [36] that reduce this delay by $2-3$ orders of magnitude. Nevertheless, even with this improvement, the optical switching latency is still more than 10 μs, which is much larger than current cut-through switches. Furthermore, the scalability of these optical proposals are limited by the port density of optical switches (320 ports [45]).

In addition to latency and scalability, traffic demand estimation is required to configure OCS. The most common solution is to predict the highly utilized links to determine the configuration of the optical switches. However, traffic estimation is highly unreliable because datacenter network traffic is typically bursty [16].

Chapter 3

Architecture

In this chapter, we describe the architecture of Quartz and explore replacing different portions of traditional DCNs with a Quartz ring.

3.1 Quartz

Quartz is a *WDM-ring network* that implements a full mesh as a single physical ring. Each Quartz switch has, in addition to n standard ports, k *optical transceivers*[1]. 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 a *WDM*. 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 form λ_{s*} to the ring. Thus implementing a full mesh requires only *two* physical cables to connect to each Quartz switch.

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

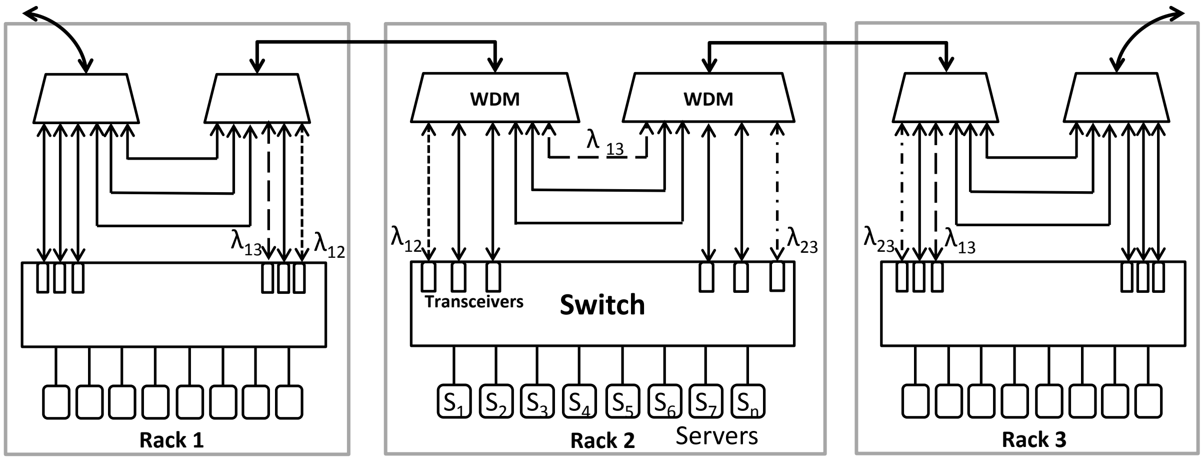


Figure 3.1: Quartz switches with $n = 8$ and $k = 6$. Each switch is only physically connected to two nearby switches by an optical cable. Switch 1 and switch 2 are connected using channel λ_{12}. Switch 1 and switch 3 are connected using wavelength channel λ_{13}.

Figure 3.1 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.

3.1.1 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 80 channels. 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

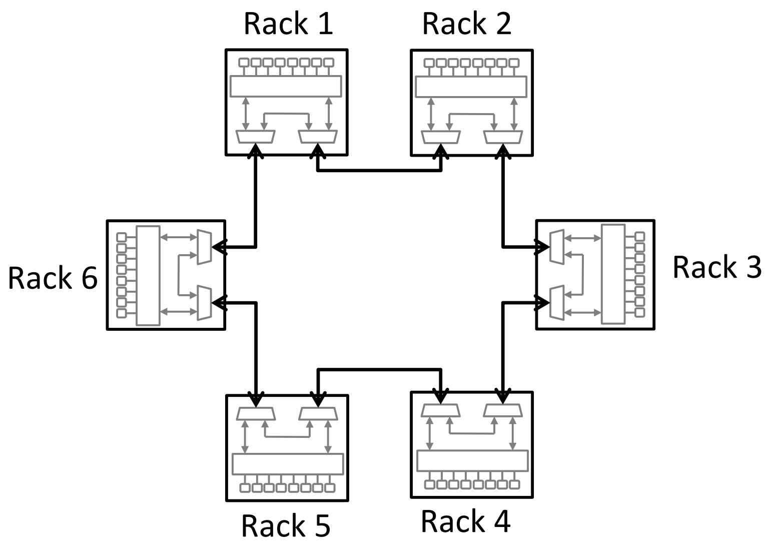


Figure 3.2: Quartz ring of size 6. The logical topology is equivalent to a mesh structure where any two switches are directly connected. By using WDM, only adjacent switches are connected with a optical fiber.

and t, there is no other channel using the same wavelength λ_{st}. For example, in Figure 3.2 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 [48]. 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.

ILP Formulation

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. 3.2).

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, $L_{s,t,i,m} = 1$. This is guaranteed by Eq. 3.3.

On each link m, a single channel i should be only used at most once. We ensure this set of constraints by Eq. 3.4. To count the total number of channels used, variable λ_i is created to show whether channel i is used in the ring. Eq. 3.5 makes sure that λ_i equals 1 if channel i is used in the ring.

$$\text{minimize: } \sum_i \lambda_i \tag{3.1}$$

subject to:

$$\forall s < t,\ \sum_i C_{s,t,i} + \sum_i C_{t,s,i} = 1 \tag{3.2}$$

$$\forall s,t,i,m,\ \ L_{s,t,i,m} = P_{s,t,m} C_{s,t,i} \tag{3.3}$$

$$\forall m,\ \ i, \sum_{s,t} L_{s,t,i,m} \leq 1 \tag{3.4}$$

$$\forall i,\ \sum L_{s,t,i,m} \leq M\lambda_i \tag{3.5}$$

$$\forall\ variable \in \{0,1\} \tag{3.6}$$

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.

Greedy Channel Assignment

We outline a simple, greedy algorithm to solve the channel assignment problem for larger rings. 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.

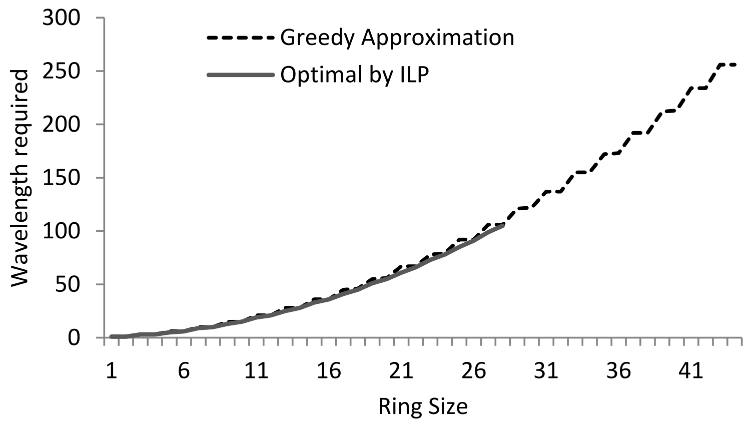


Figure 3.3: Optimal wavelength assignment

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.

To evaluate the greedy algorithm, we compare the results with the ILP solution. Figure 3.3 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.

3.1.2 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 3.2.

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

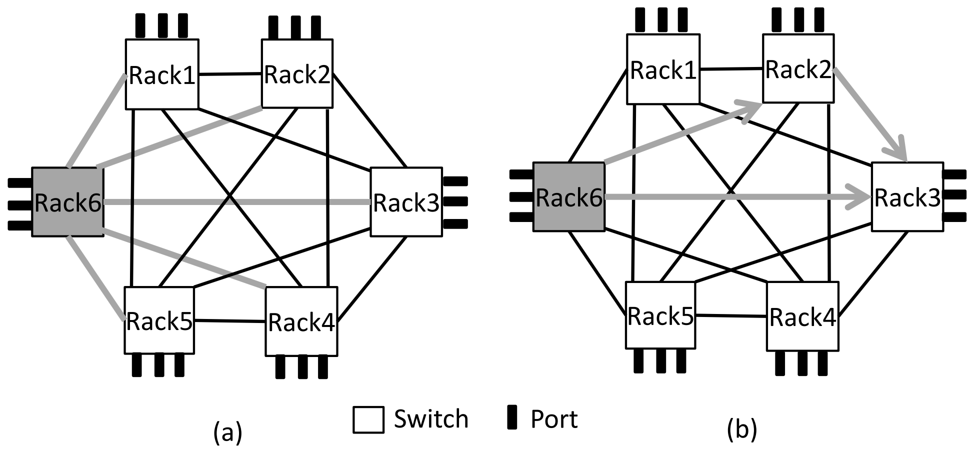


Figure 3.4: A flat mesh network where ToR switches are directly connected. (a) direct routing (b) two-hop routing.

servers is still two switches. However, this configuration can support up to 2080 (32×65) ports at the cost of an additional switch per rack.

3.1.3 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 [23] and MapReduce [19], are unaffected by this kind of independent, rack-to-rack bandwidth overscription.

For workloads that concentrate traffic between two racks, one can significantly reduce rack-to-rack oversubscription by using Valiant Load Balancing (VLB) [20,34] 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 3.4(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 4.2.

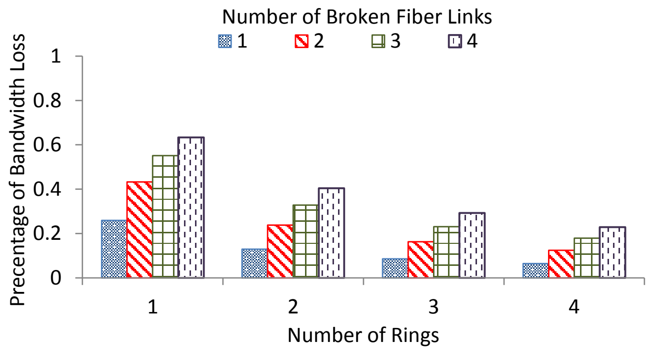
3.1.4 Fault Tolerance

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 mux/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.

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 3.5 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.

3.2 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.



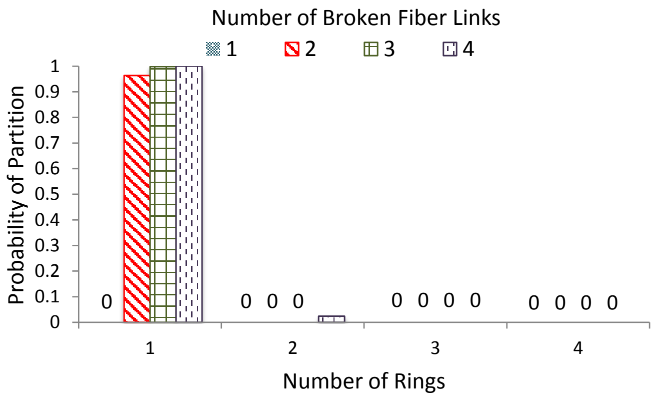


Figure 3.5: The top graph shows the percentage of bandwidth loss from broken fibre links with different ring sizes. The bottom figure shows the probability of network partitioning. With two rings, four link failures can partition the network at a very low probability of 0.0024.

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 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.

3.2.1 Quartz in the Edge

Current DCNs typically employ either 2 or 3-tier tree topology [13]. Figure 4.9 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 4.12. 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 [31].

3.2.2 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 [5]. Therefore, although these switches provide modular scalability, the large upfront cost of the chassis means that 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 4.11. 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 4.5.

3.2.3 Quartz in Random Topology Networks

Random topology networks, such as Jellyfish [40] and SWDC [39], 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 3.2.1. 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.

3.2.4 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 3.1, which summarizes the cost of using Quartz in various network configurations. We consider different datacenter sizes, ranging from 500 servers to 100,000 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 4.4 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 5% compared to a two-tier tree structure. However, we can achieve a latency reduction of at least 33%

Datacenter Size	**Utilization**	**Sample Topologies**	**LRQ**	**Cost/Server**
Small (500 Servers)	Low	Two-tier tree	33%	\$589
		Single Quartz ring		\$616
	High	Two-tier tree	50%	\$589
		Single Quartz ring		\$616
Medium (10K Servers)	Low	Three-tier tree	20%	\$530
		Quartz in edge		\$628
	High	Three-tier tree	40%	\$530
		Quartz in edge		\$628
Large (100K Servers)	Low	Three-tier tree	70%	\$525
		Quartz in core		\$525
	High	Three-tier tree	74%	\$525
		Quartz in edge and core		\$601

Table 3.1: Approximate cost and latency comparison. Costs include all the hardware expenses except servers. Note: LRQ represents Latency Reduction with Quartz.

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 18% 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 13% 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.

Chapter 4

Evaluation

We have previously described the architecture of Quartz and its use as a design element in large DCNs. In this chapter, we analyse Quartz's path diversity and bisection bandwidth to determine the theoretical performance benefit of using a full mesh network as a design element. We also evaluate Quartz's performance using both simulations and a small working prototype.

4.1 Wiring Complexity and Path Diversity

We analyze the properties of five representative network topologies (2-tier tree, Fat-Tree [13], BCube [25], Jellyfish [40] 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 define wiring complexity as the number of cross-rack links. We compute the path diversity of each topology using the metric defined in [41]:

Definition (Path Diversity) Given a node pair (source s, destination d), the path diversity PD between them is the number of disjoint paths for a packet to transit between s and d.

For example, even though there exits 3 different paths between node pair (s, d), the path diversity is only 2 due to a shared link for two of these paths in Figure 4.1. 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. Table 4.1 shows a summary of the key properties of these topologies.

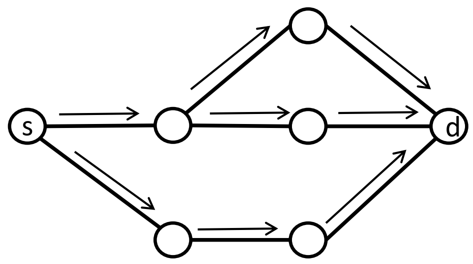


Figure 4.1: Sample graph for path diversity.

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 [13], 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.

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 lowest hop count and latency when using direct routing, and relatively high bisection bandwidth and the highest path diversity when using indirect routing with VLB. It also has the highest 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.

4.2 Bisection Bandwidth

Given the use of two-hop paths in Quartz using VLB, it is difficult to analytically estimate its bisection bandwidth. Instead, we use simulations to compare the aggregate throughput

	Switching Latency	**NoS**	**Wiring Complexity**	**PD**
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 VLB)
			32 (with WDMs)	

Table 4.1: Summary of different network structures with 1k servers. Note: NoS represents the Number of 64-port Switches and PD stands for Path Diversity.

of a Quartz network normalized 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]:

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 4.2 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 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.

[1]Note that all the traffic goes to different racks, that is the worst case comparison.

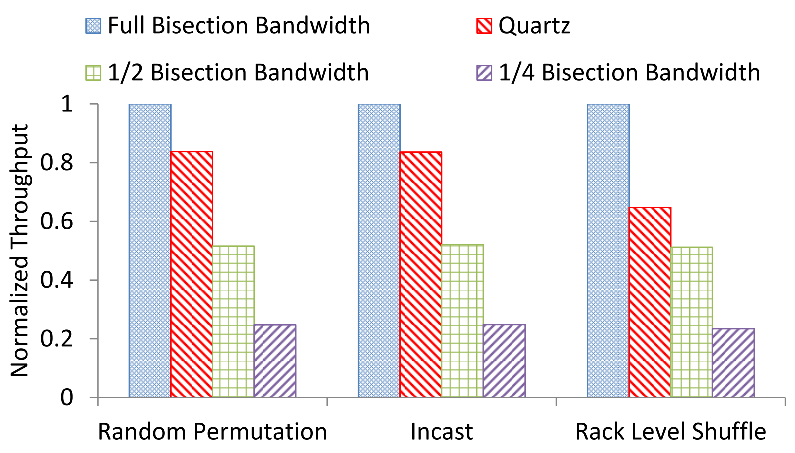


Figure 4.2: Normalized throughput for three different traffic patterns

4.3 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 4.4. 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 4.3(a) shows the logical connectivity between the switches. By using WDM muxes/de muxes, we can simplify the network cabling to Figure 4.3(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.

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.

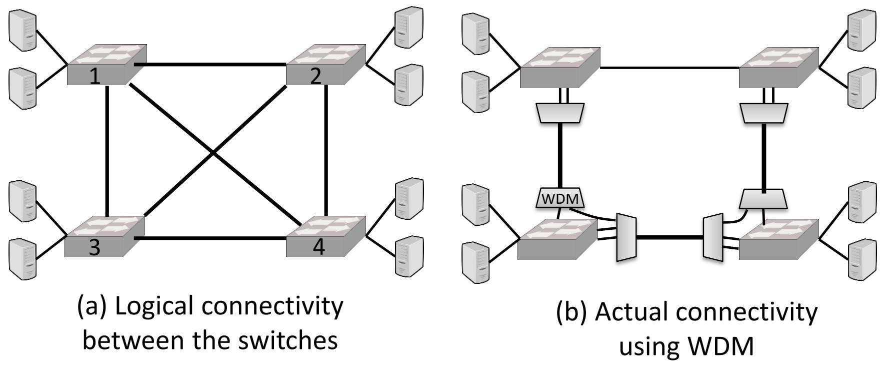

1
2
3
4
(a) Logical connectivity between the switches
WDM
(b) Actual connectivity using WDM

Figure 4.3: Topology of our testbed.

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.

4.3.1 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 4.3 for evaluating Quartz, and rewired the switches from the Quartz prototype into a 2-tier tree (1 aggregation and 3 ToR switchs) to perform our 2-tier tree experiments. Each ToR switch is connected to two servers and we use six total servers in our experiments.

Our cross-traffic workload consists of a "Hello World" RPC written in Apache Thrift [10] 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 [11] from three servers connected to S_1 and S_2 to a server connected to S_3. Figure 4.5 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

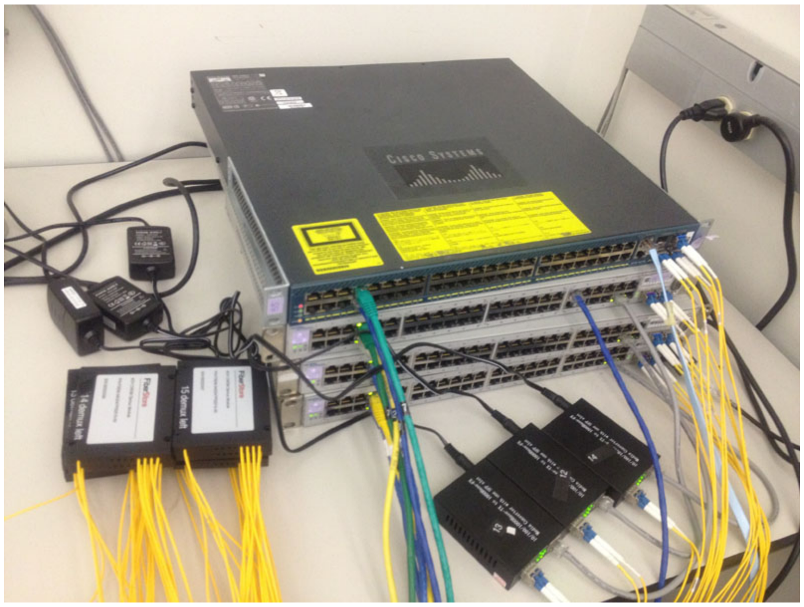


Figure 4.4: Quartz with 4 switches, connected by WDMs

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 4.6 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 4.1, and demonstrate the impact of the network topology on reducing network congestion from cross-traffic.

4.4 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. With the simulator, we can determine the latency performance of any

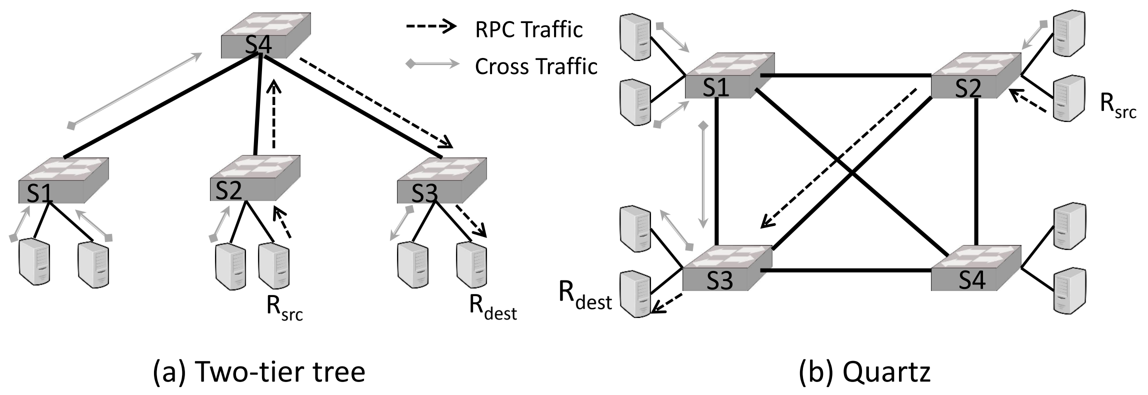


Figure 4.5: Traffic flows of cross-traffic experiment

network topology and any network devices. In this section, we first present our simulator verification process. We then discuss the configuration details of our experiments. Finally, we present our experimental results that compare Quartz with other network topologies.

4.4.1 Verification Process

Our simulator focuses on modelling switching latency, propagation delay and queueing latency. Since the switching latency and propagation delay are independent of the system load, which can be validated using simple test cases, we focus on verifying our simulator's queueing latency.

Our simulator models switches that have an output-queued switching architecture. Therefore, each of our switches has a switching-processor that makes the forwarding decisions, and an output buffering queue for each port as shown in the Figure 4.7. We have performed extensive validation of our simulator to ensure that it produces queueing latencies that match queueing theory [7]. Our validation test uses an M/M/1 queueing model which requires: (1) the packets arrive at rate λ according to a Poisson process; (2) service time for each packet has an exponential distribution with parameter μ; (3) buffer size is unbounded and is served in FIFO order. With this model, the average response time (waiting and service time) t in the buffer should be:

$$t = \frac{1}{\mu - \lambda}.$$

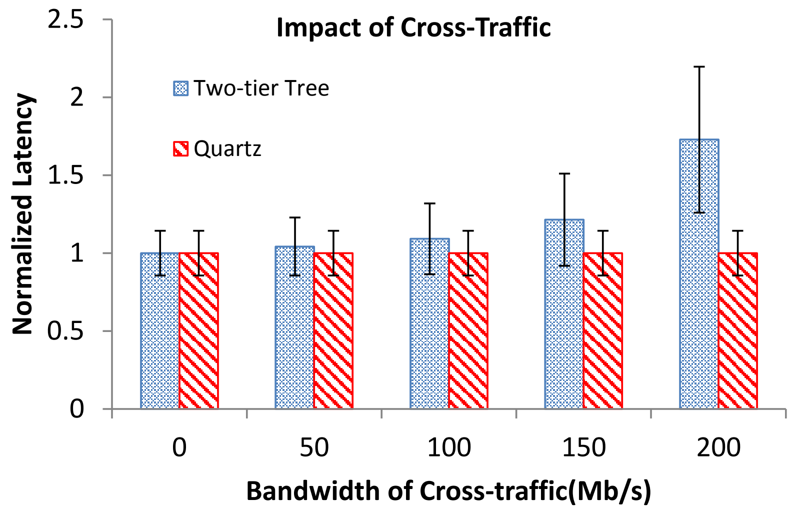


Figure 4.6: Impact of Cross-traffic on different topologies

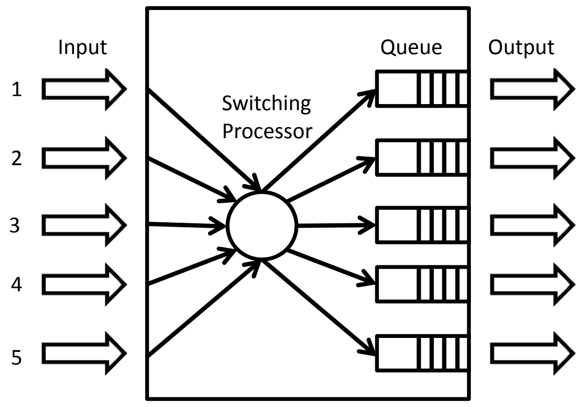


Figure 4.7: The switching fabric of switch in the simulator. Note: Routing decisions are made by micro-processor.

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 4.2: Specifications of switches used in our simulations.

Our validation process involves generating a large number of packets with exponentially distributed sizes, and sending them into the network according to a Poisson process. We vary the packet size and arrival rate to verify that the response time matches queueing theory. Figure 4.8 demonstrates two sample verification results with different packet sizes. As expected, the cumulative average delay of packets quickly converges to the M/M/1 expected result.

4.4.2 Configuration Details

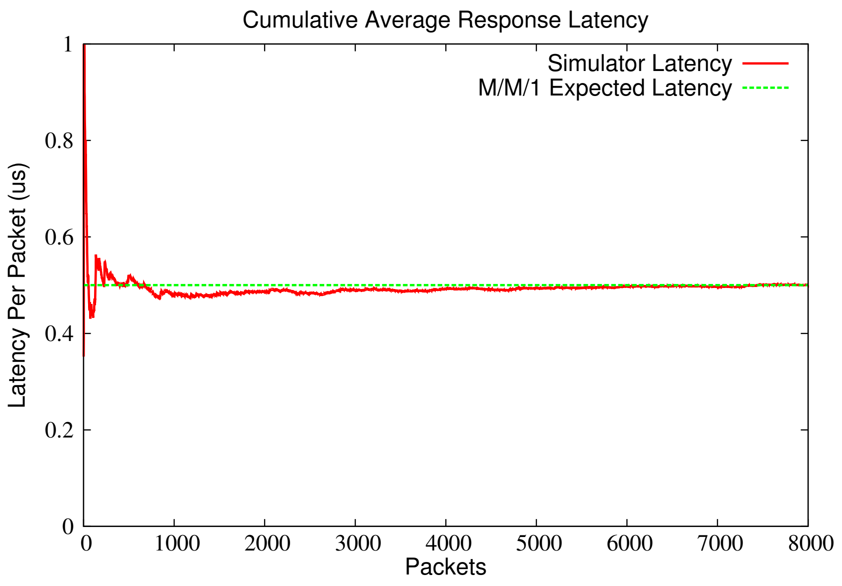
We make the simplifying assumptions that servers send packets according to a Poisson process and packet sizes are exponentially distributed with a mean of 400 bytes. We model two state-of-the-art switches in our simulator:

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

The specifications of these switches are summarized in Table 4.2. In our simulated architectures, we use ULL exclusively in fat-tree and Quartz, while we use ULL for both ToR switches and aggregation switches, and CCS as core switches for the other architectures. 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 [2] :

1. **Three-tier Tree (Figure 4.9):** 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] Jellyfish employs *k-shortest-path*, while the others adopt Equal-Cost Multi-Path (ECMP) as routing algorithm.



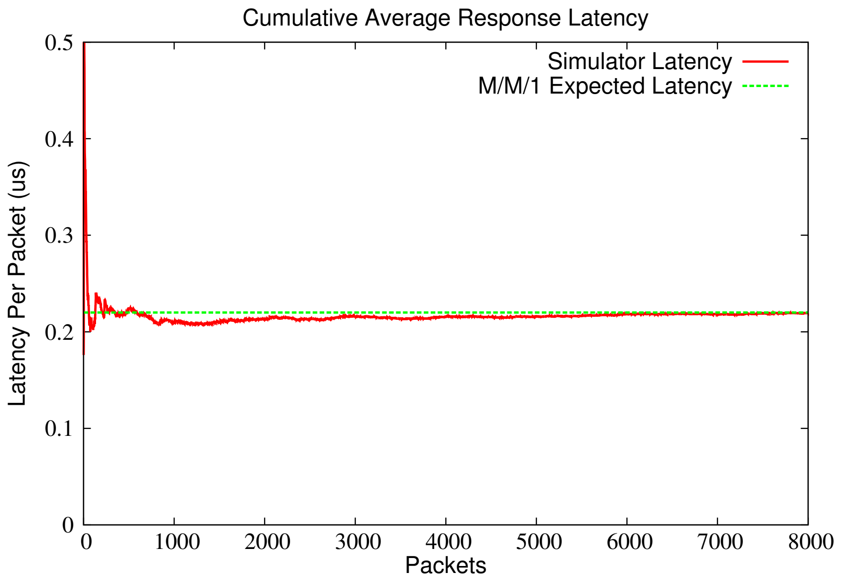


Figure 4.8: M/M/1 queueing verification results. Note: packets in both figures have an arrival rate of 0.5 packet/μs with different service times (top: 2.5 packets/μs, bottom: 5 packets/μs). The M/M/1 expected delays are 0.5 μs and 0.22 μs respectively.

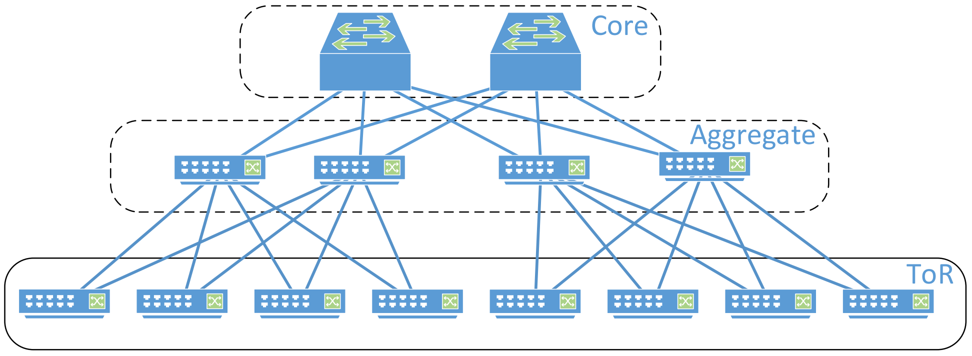


Figure 4.9: Multi-root 3-tier tree.

2. **Three-tier Fat-tree (Figure 4.10):** The network is divided into 4 pods, where four switches are connected over 10Gbps links for each pod, and each aggregation switch connects to two ULL switches over 40Gbps links.

3. **Quartz in Core (Figure 4.11):** Each core switch is replaced with a Quartz ring. The aggregation switches connect to the Quartz ring over 40 Gbps links.

4. **Quartz in Edge (Figure 4.12):** 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.

5. **Quartz in Edge and Core (Figure 4.13):** Both core and edge layers are replaced with Quartz rings.

6. **Jellyfish:** A random topology consisting of 16 ULL, with each switch dedicating four 10 Gbps links to connect to other switches.

7. **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.

4.4.3 Results

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

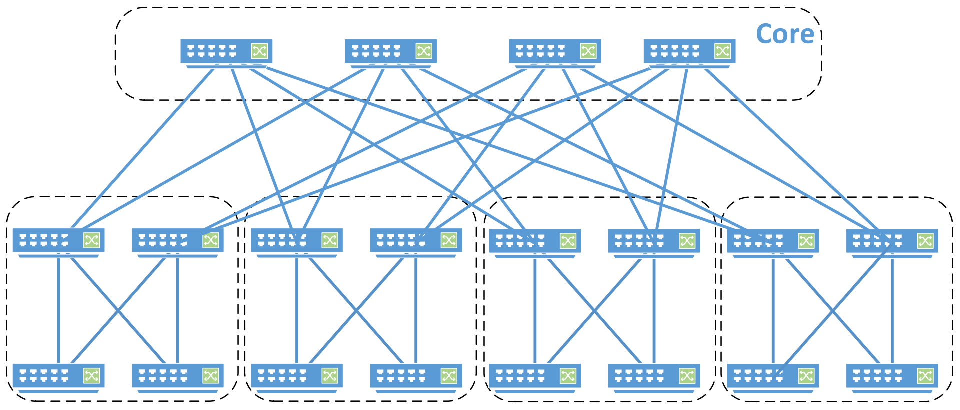


Figure 4.10: Fat-tree with 4 pods.

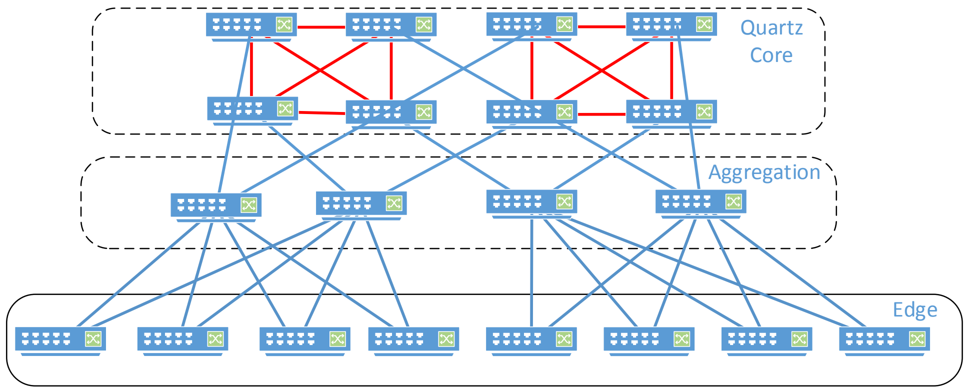


Figure 4.11: Quartz in core tier of 3-tier tree.

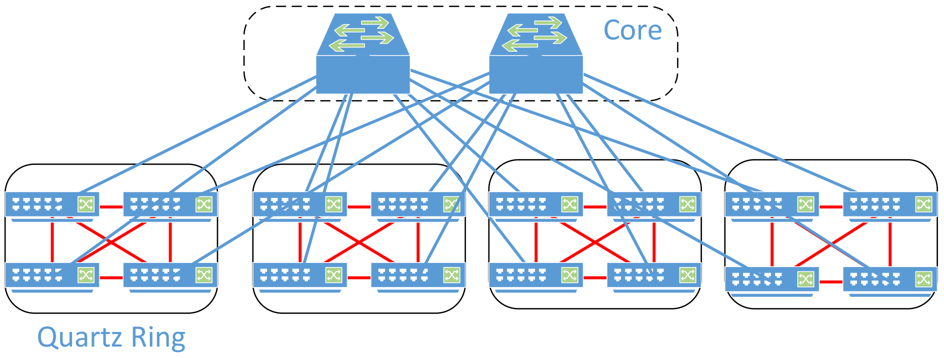


Figure 4.12: Quartz in edge tier of 3-tier tree.

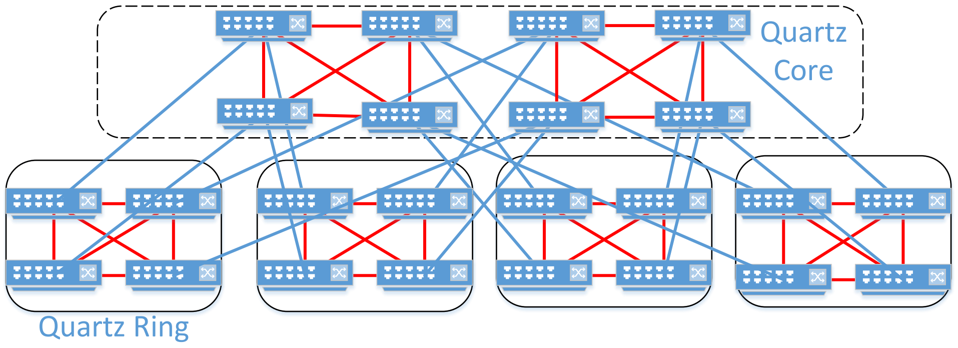


Figure 4.13: Quartz in both core and edge tier of 3-tier tree.

- **Scatter:** One host is the sender and the others are receivers. The sender concurrently sends a flow of packets in Poisson process to the receivers.
- **Gather:** One host is the receiver and the others are senders. The senders concurrently send a flow of packets in Poisson process to the receivers.
- **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 [15], and are also common in high-performance computing applications, with MPI [23] 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 4.14, 4.15 and 4.16 show 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.

Fat-tree has similar latency characteristics as Jellyfish, since it also uses low-latency cut-through switches instead of high-latency core switches in the core layer. Furthermore, since Fat-tree provides non-blocking throughput, it does not exhibit the latency jump from adding the fourth task.

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 tasks 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 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 4.17, 4.18 and 4.19 show 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 have randomly distributed senders and receivers and are used to generate cross-traffic. The three-tier tree has significantly lower latency in this experiment because the local task traffic does not have to traverse the core tier. However, it still exhibits a linear increase in latency with additional tasks. Unlike the global traffic patterns experiment, Fat-tree does not benefit from its cut-through core switches for localized traffic. Therefore, its latency is identical to the multi-root tree until the links in the multi-root tree become saturated.

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, and scatter/gather operations to fewer targets than the non-local tasks. This accounts for the slight 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 remains 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.

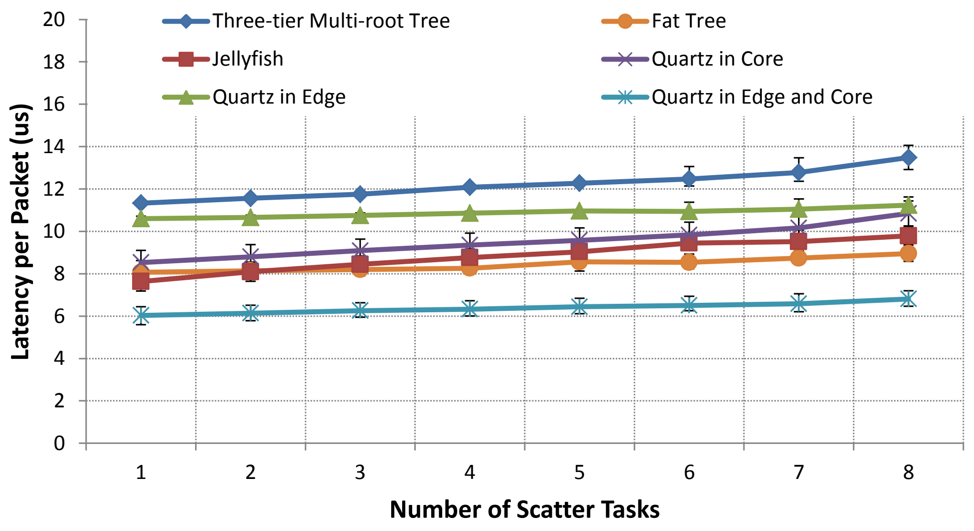


Figure 4.14: Average latency for global scatter traffic. This graph is best viewed in colour.

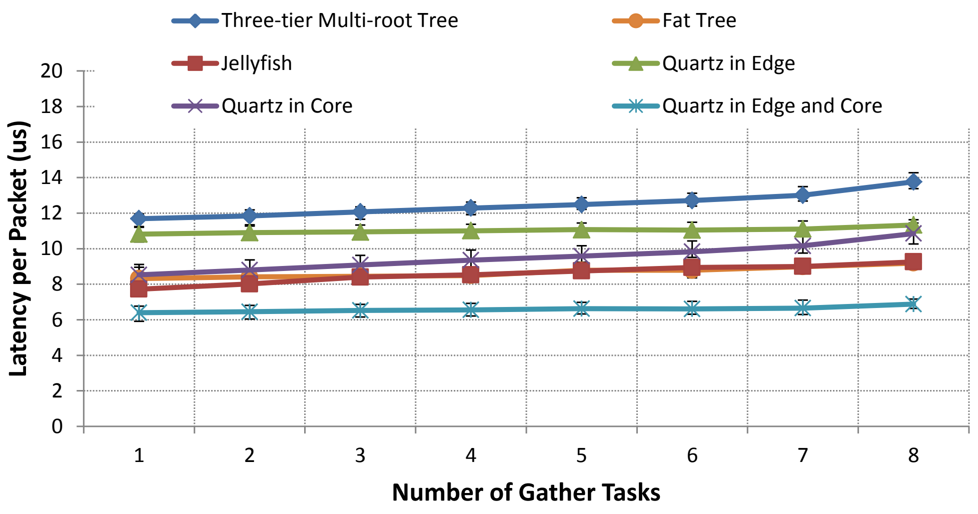


Figure 4.15: Average latency for global gather traffic. This graph is best viewed in colour.

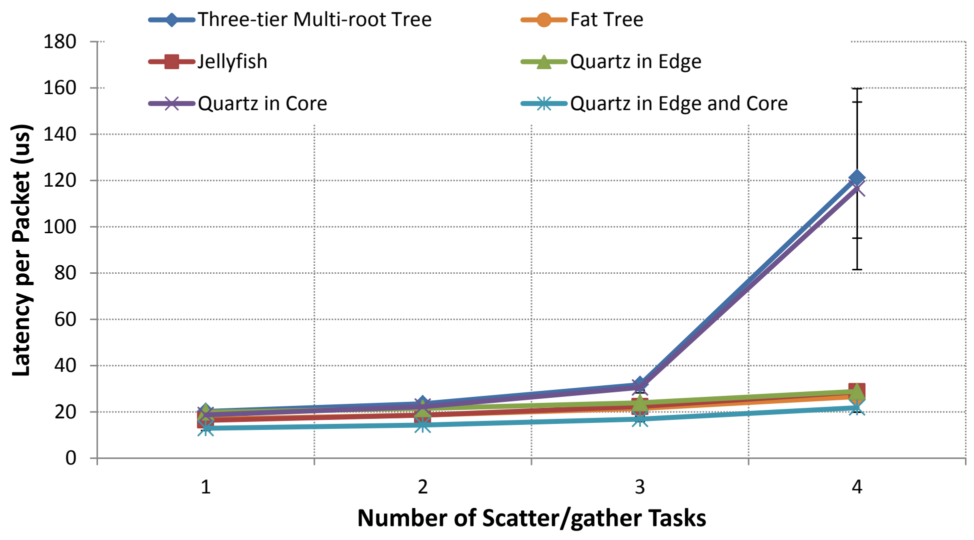


Figure 4.16: Average latency for global scatter/gather traffic. This graph is best viewed in colour.

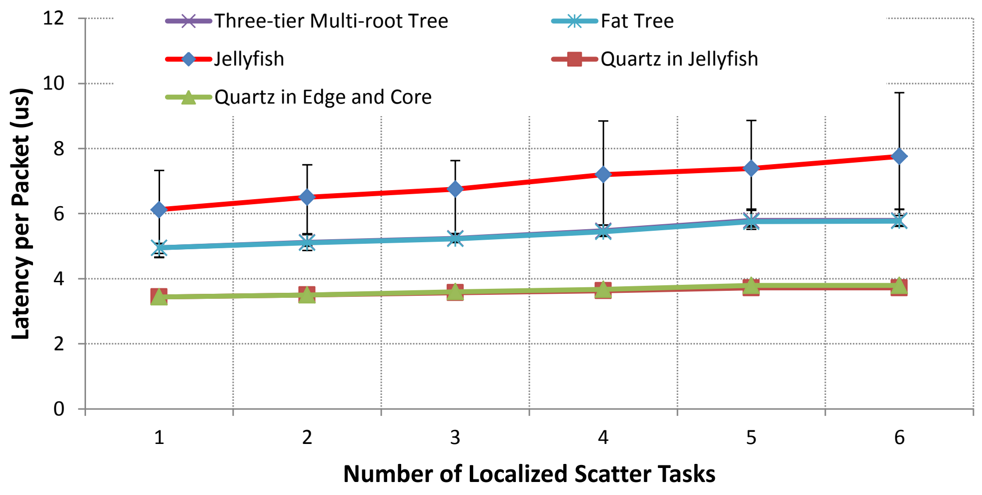


Figure 4.17: Average latency for local scatter traffic. This graph is best viewed in colour. Note that Fat-tree is identical to multi-root tree in this figure.

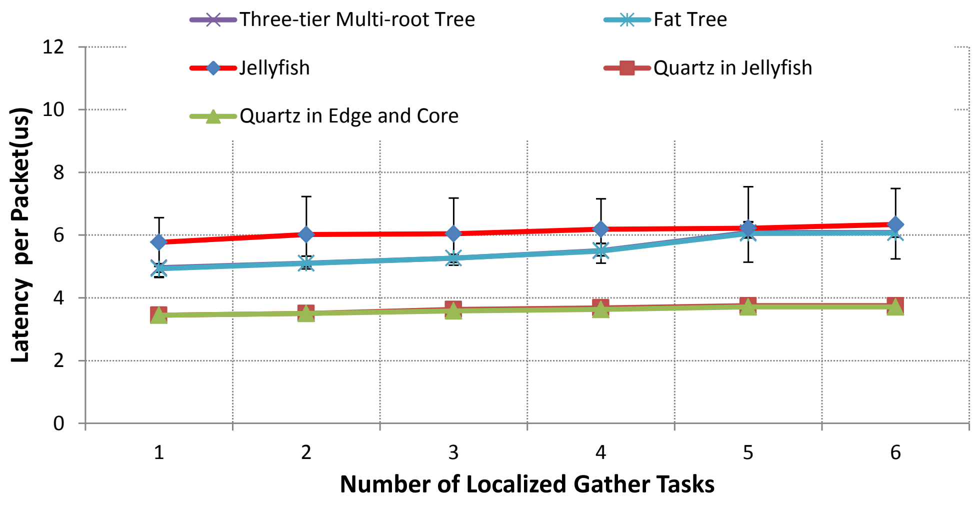


Figure 4.18: Average latency for local gather traffic. This graph is best viewed in colour. Note that Fat-tree is identical to multi-root tree in this figure.

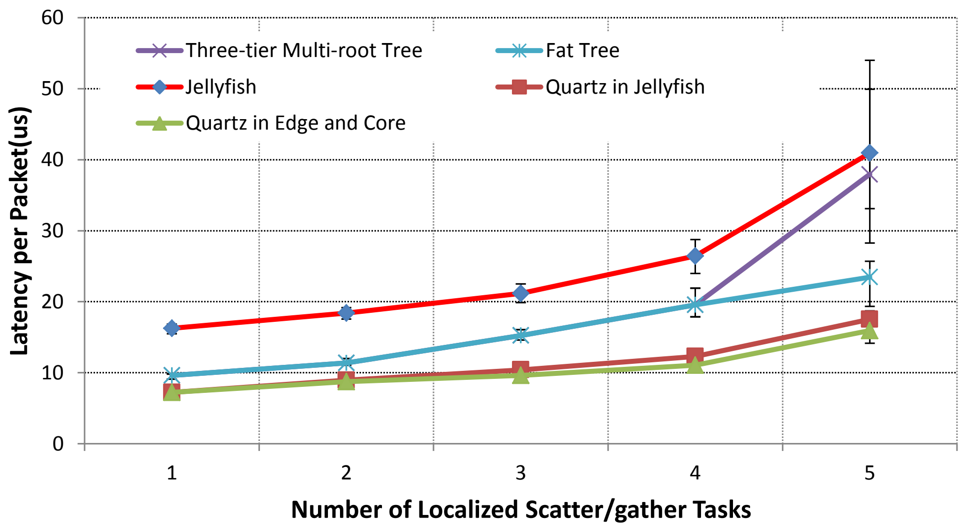


Figure 4.19: Average latency for local scatter/gather traffic. This graph is best viewed in colour.

4.5 Pathological Traffic Pattern

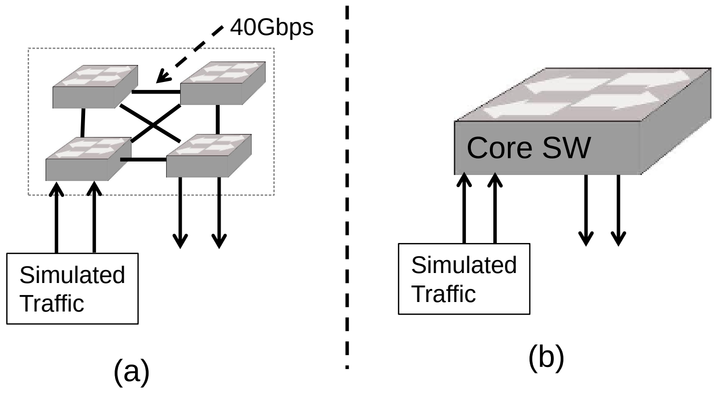


Figure 4.20: Pathological traffic pattern.

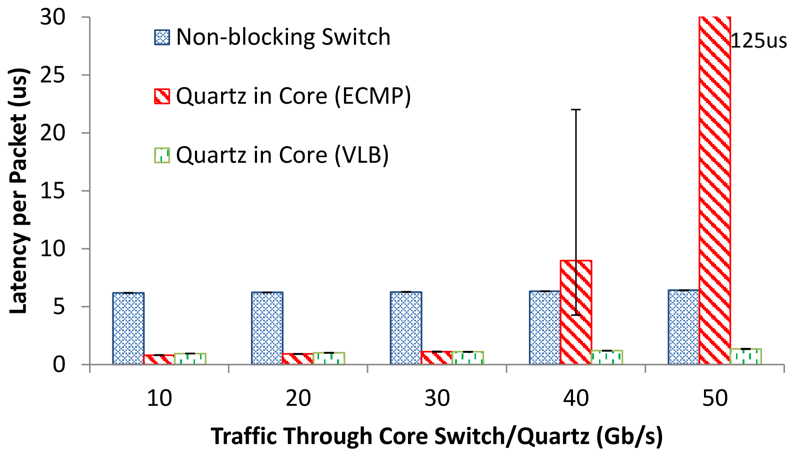


Figure 4.21: Average latency comparison for pathological pattern.

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 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 4.20(a) and we use a standard non-blocking core switch as shown in Figure 4.20(b) for comparison.

Figure 4.21 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 taffic, there is no noticeable increase in packet latency when performing VLB routing.

Chapter 5

Conclusion

The goal of this thesis is to reduce the latency for certain applications that require low latency in DCNs. Past work to reduce network latency has a number of limitations, for example, some of them rely on protocol changes [15, 49], or require network-wide flow scheduling [14] or strict resource reservations [30]. Our focus is on using an architectural approach to reducing latency that is complementary to previous efforts. The key insight behind our design, Quartz, is that we can implement an O(n^2) mesh as an O(n) physical ring by using commodity WDMs. Furthermore, with the exponentially drop in the cost of optical devices, Quartz will become more cost-effective with time.

The main limitation of a WDM-ring is its limited scalability. We overcome the scalability limitation by using Quartz as a design element that can replace *portions* of current DCNs. Quartz can be used to replace the edge and aggregate tier, or the core tier, or both. 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 increases, Quartz becomes more scalable.

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.

In this thesis, we have evaluated these ideas extensively. Through the analysis of path diversity and bisection bandwidth, we found that WDM mesh ring is the most suitable design element for low latency DCNs compared to several representation topologies. Our experimental results corroborate our analysis; the results show that Quartz can significantly reduce the latency of a DCN under several realistic workloads. In particular, we found that using Quartz in both the core and edge can reduce latency by 50% in typical scenarios.

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

[5] Cisco Chassis Price. December 2013. http://bit.ly/1dOBFHr.

[6] Hadoop Map-Reduce Tutorial. December 2013. http://bit.ly/WZP6Hs.

[7] M/M/1 Queueing theory, December 2013. http://bit.ly/1a1eGYc.

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

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

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

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

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

[13] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the SIGCOMM Conference*, Seattle, Washington, August 2008.

[14] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *Proceedings of the USENIX Conference on Networked Systems Design and Implementation(NSDI)*, San Jose, California, April 2010.

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

[16] theophilus Benson, Aditya Akella, and David A. Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the Internet Measurement Conference(IMC)*, Melbourne, Australia, November 2010.

[17] Joseph 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.

[18] Kai Chen, Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang, Xitao Wen, and Yan Chen. OSA: an optical switching architecture for data center networks with unprecedented flexibility. In *Proceedings of the Symposium on Networked System Design and Implementation(NSDI)*, San Jose, California, April 2012.

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

[20] Mihai Dobrescu, Norbert Egi, Katerina Argyraki, Byung-Gon Chun, Kevin Fall, Gianluca Iannaccone, Allan Knies, Maziar Manesh, and Sylvia Ratnasamy. Routebricks: Exploiting parallelism to scale software routers. In *Proceedings of the Symposium on Operating Systems Principles(SOSP)*, Big Sky, Montana, October 2009.

[21] Franz Färber, Sang Kyun Cha, Jürgen Primsch, Christof Bornhövd, Stefan Sigg, and Wolfgang Lehner. SAP HANA database: data management for modern business applications. *SIGMOD Rec.*, January 2012.

[22] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *Proceedings of the SIGCOMM Conference*, New Delhi, India, August 2010.

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

[24] Binbin Guan, Ryan P. Scott, Chuan Qin, Nicolas K. Fontaine, Tiehui Su, Carlo Ferrari, Mark Cappuzzo, Fred Klemens, Bob Keller, Mark Earnshaw, and S. J. B. Yoo. Free-space coherent optical communication with orbital angular, momentum multiplexing/demultiplexing using a hybrid 3d photonic integrated circuit. *Optics Express Magazine*, January 2014.

[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 the SIGCOMM Conference*, 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 the SIGCOMM Conference*, Seattle, Washington, August 2008.

[27] Andrew G Haldane. Patience and Finance. *Bank of England*, November 2010. http://bit.ly/11dMLz0.

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

[29] 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.

[30] Nan Jiang, D.U. Becker, G. Michelogiannakis, and W.J. Dally. Network congestion avoidance through speculative reservation. In *Proceedings of the Symposium on High Performance Computer Architecture (HPCA)*, New Orleans, Louisiana, Feburary 2012.

[31] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. the nature of data center traffic: Measurements & analysis. In *Proceedings of the Internet Measurement Conference(IMC)*, Chicago, Illinois, November 2009.

[32] R. Kapoor, G. Porter, M. Tewari, G.M. 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.

[33] Heiner Litz, Holger Froning, Mondrian Nuessle, and Ulrich Bruning. A HyperTransport Network Interface Controller for Ultra-low Latency Message Transfers. *Hyper Transport Consortium*, Feburary 2008.

[34] Huan Liu and Rui Zhang-Shen. On direct routing in the valiant load-balancing architecture. In *Proceedings of the Global Telecommunications Conference(GLOBECOM)*, St. Louis, Missouri, November 2005.

[35] Jayaram Mudigonda, Praveen Yalagandula, Mohammad Al-Fares, and Jeffrey C. Mogul. Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In *Proceedings of the USENIX Conference on Networked Systems Design and Implementation(NSDI)*, San Jose, CA, April 2010.

[36] George Porter, Richard Strong, Nathan Farrington, Alex Forencich, Pang Chen-Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Integrating microsecond circuit switching into the data center. *SIGCOMM Computer Communication Review Magazine(SIGCOMM CCR)*, August 2013.

[37] Luigi Rizzo. Netmap: A novel framework for fast packet i/o. In *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, Boston, MA, June 2012.

[38] Stephen M. Rumble, Diego Ongaro, Ryan Stutsman, Mendel Rosenblum, and John K. Ousterhout. It's time for low latency. In *Proceedings of the Workshop on Hot Topics in Operating Systems(HotOS)*, Napa, California, May 2011.

[39] Ji-Yong Shin, Bernard Wong, and Emin Gün Sirer. Small-world datacenters. In *Proceedings of the ACM Symposium on Cloud Computing*, Cascais, Portugal, October 2011.

[40] A. Singla, C.Y. Hong, L. Popa, and P.B. Godfrey. Jellyfish: Networking data centers randomly. In *Proceedings of the USENIX Conference on Networked Systems Design and Implementation(NSDI)*, San Jose, California, April 2012.

[41] Renata Teixeira, Keith Marzullo, Stefan Savage, and Geoffrey M. Voelker. Characterizing and measuring path diversity of internet topologies. June 2003.

[42] Bhanu Chandar Udatha. Report on Direct Market Access and Ultra Low Latency Trading Facilities in India. March 2011. http://ssrn.com/abstract=1795782.

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

[44] Ashish Vulimiri, Philip Brighten Godfrey, Radhika Mittal, Justine Sherry, Sylvia Ratnasamy, and Scott Shenker. Low latency via redundancy. In *Proceedings of the Conference on Emerging Networking Experiments and Technologies(CoNEXT)*, Santa Barbara, California, December 2013.

[45] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T.S. Eugene Ng, Michael Kozuch, and Michael Ryan. c-Through: part-time optics in data centers. In *Proceedings of the SIGCOMM Conference*, New Delhi, India, August 2010.

[46] Christo Wilson, Hitesh Ballani, Thomas Karagiannis, and Ant Rowtron. Better never than late: meeting deadlines in datacenter networks. In *Proceedings of the SIGCOMM Conference*, Toronto, Canada, August 2011.

[47] L. XU and Y. WENG. Oracle timesten in-memory databases automatic data cleaning mechanism explored. *Computer Knowledge and Technology Journal*, October 2010.

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

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