Small-World Datacenters

Ji-Yong Shin† Bernard Wong‡ Emin Gün Sirer†

†Dept. of Computer Science
Cornell University
{jyshin, egs}@cs.cornell.edu

‡David R. Cheriton School of
Computer Science
University of Waterloo
bernard@uwaterloo.ca

ABSTRACT

In this paper, we propose an unorthodox topology for datacenters that eliminates all hierarchical switches in favor of connecting nodes at random according to a small-world-inspired distribution. Specifically, we examine topologies where the underlying nodes are connected at the small scale in a regular pattern, such as a ring, torus or cube, such that every node can route efficiently to nodes in its immediate vicinity, and amended by the addition of random links to nodes throughout the datacenter, such that a greedy algorithm can route packets to far away locations efficiently. Coupled with geographical address assignment, the resulting network can provide content routing in addition to traditional routing, and thus efficiently implement key-value stores. The irregular but self-similar nature of the network facilitates constructing large networks easily using prewired, commodity racks. We show that Small-World Datacenters can achieve higher bandwidth and fault tolerance compared to both conventional hierarchical datacenters as well as the recently proposed CamCube topology. Coupled with hardware acceleration for packet switching, small-world datacenters can achieve an order of magnitude higher bandwidth than a conventional datacenter, depending on the network traffic.

Categories and Subject Descriptors

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

General Terms

Design, Management, Performance

Keywords

Datacenter network, Network topology, Small-world network

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

1. INTRODUCTION

Network topology is one of the key determinants of application performance in datacenters. Traditional datacenters are built around a hierarchical topology using multiple layers of switches [8]. While such hierarchical topologies are modular, composable, and easy to set up, they suffer from performance bottlenecks due to oversubscription at the higher layers of the hierarchy. High-bandwidth connection technologies at the upper layers of the tree can mitigate oversubscription, but often entail high costs.

Consequently, much recent research has focused on alternative datacenter topologies to achieve high bisection bandwidth, improve failure resiliency, and provide low latency. For instance, DCell [14], BCube [13], and CamCube [1] provide richer topologies with higher performance. In these systems, costly network switches are replaced, and the routing functionality is shifted to the datacenter nodes, often equipped with additional NICs. Such a shift in functionality can improve bisection bandwidth and provide new functionality, such as content addressing, that might be a better fit for common datacenter applications such as key-value stores. Yet these past approaches have their own shortcomings; namely, they rely critically on a rigid connection pattern that is not easy to set up, they incur high latencies, and, while they provide higher bandwidth, they may still fall short of what is practically achievable.

In this paper, we propose a novel, and unorthodox, topology for datacenters. Like previous work, we eschew hierarchical switches in favor of a rich topology in which the datacenter nodes aid in routing. But unlike prior work, we propose that the datacenter network be wired, essentially, at random. Specifically, we propose networks based on a locally-regular foundation that is easy to wire, such as a ring, grid, cube or torus, and amend this regular structure with a number of network links placed throughout the datacenter. These links in the latter category follow a small-world-inspired [39] pattern, such that nodes have a relatively large number of links to the nearby nodes around them so as to be authoritative in their region, and possess sufficient links of the right lengths such that they can cover large distances when needed. The result is a datacenter topology with low stretch and high bandwidth that is very easy to provision.

This random topology can greatly simplify the construction of high performance datacenter networks. We envision that a Small-World Datacenter (SWDC) can be constructed

simply by pre-cutting cabling ahead of time according to the appropriate distribution. A larger number of short cables provide the regular grid structure, which ensures that the network is not partitioned and that all nodes are reachable. The instructions for the provisioning team consist of implementing a grid followed by establishing connections at random.

Small-world networks lend themselves naturally to scalable construction from pre-wired rack components. We describe designs where each rack comes pre-wired with the regular links. It can be dropped into a network of any size, and fully integrated into the fabric simply by connecting links at random through a patch panel.

The random links in SWDCs have the potential to reduce network latency and improve bandwidth, especially in comparison to previous work that limits all routes to a regular, rigid pattern. Furthermore, the combination of random and regular links enables a simple greedy algorithm to effectively route packets within the datacenter. SWDCs also enable content addressable networks, just as with the state of the art datacenter topologies such as CamCube [1].

SWDCs rely on a collection of NICs in each node to provide the rich topology that achieves low latency and high bandwidth. To fairly compare to past work, we investigate topologies based on 6 NICs per node. In an ideal scenario, the NICs would be integrated onto the same daughter board or coupled with hardware accelerators such as GPUs and NetFPGA [17, 27, 32] in order to avoid store-and-forward delays when a node is routing from NIC to NIC. We investigate systems based both on unintegrated, commodity NIC hardware, as well as systems with such optimizations.

Overall, this paper makes three contributions. First, we propose a novel topology based on small-world networks, and show that the judicious use of random wiring can yield competitive datacenter networks. Second, we investigate different small-world network topologies based on a ring, 2-D torus, and 3-D hexagonal torus and quantify their benefits. Finally, we provide a detailed evaluation of SWDCs, including comparisons with conventional datacenter topologies as well as CamCube. Our evaluation shows that SWDC can outperform CamCube on both bandwidth and latency, while providing the same functionality benefits stemming from content addressing. In addition, SWDC can provide up to 16 times higher bandwidth than conventional hierarchical datacenters, and can preserve over 90% of network flows among live nodes even if up to 50% of nodes fail.

The remainder of this paper explores the design of SWDCs, and evaluates and compares it with other datacenter designs. Section 2 provides background information on small-world networks. Section 3 presents the proposed SWDC designs, including different small-world topologies and physical datacenter packaging. Section 4 discusses suitable applications for SWDC, and the evaluation of SWDC is presented in Section 5. Related work is provided in Section 6 and finally, Section 7 concludes our work.

2. SMALL-WORLD NETWORKS

Small-world networks were initially inspired by the phenomena observed in Milgram's experiments [30, 35] in human social networks. Watts and Strogatz proposed the name and a mathematical simplification and formalization for such networks [39]. Examples of small-world networks are shown in Figure 1. In general, small-world networks exhibit an average pairwise distance that is proportional to the logarithm of the nodes in the network. Kleinberg examined small-world networks based on a d-dimensional lattice coupled with additional random links at each node, and showed that the network can route efficiently when the random links are chosen with probability that is proportional to the dth power of distance between two nodes, where d is the dimension of the lattice network [21, 22]. The lattice network connects the nodes to their nearest neighbors, in effect rendering them authoritative for hosts in their region, while the random links provide short cut-through paths across the network. Hereafter, we refer to links forming the lattice network *regular* links, while the randomly assigned links are known as *random* links.

For SWDCs, we adopt Kleinberg's small-world model. This network configuration guarantees that all nodes are reachable, that the network scales well, and, with high probability, the path length between any two nodes is $O(\log n)$ [24, 23].

3. SMALL-WORLD DATACENTER DESIGN

This section describes the topologies, address management, routing methods, and packaging issues regarding the design and implementation of SWDCs.

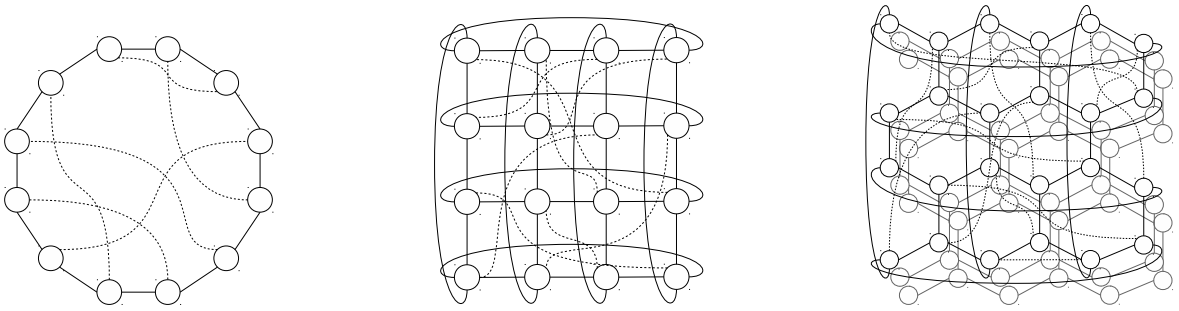
3.1 Network Topologies

The SWDC network topology admits many different instantiations. The dimension of the underlying lattice, the number of random links, degree of each node and so on can vary depending on the configuration. In this paper, for purposes of realistic deployment, low-cost, and fair comparison to past work, we limit the degree of each node to 6 and construct the following small-world networks for SWDCs:

- **Small-World Ring (SW-Ring)**: 2 links form a ring topology and 4 links connect each node to 4 random nodes.
- **Small-World 2-D Tours (SW-2DTorus)**: 4 links form a 2-D torus topology and 2 links connect each node to 2 random nodes.
- **Small-World 3-D Hexagonal Torus (SW-3DHexTorus)**: 5 links form a 3-D hexagon torus and 1 link connects each node to a random node.

Figure 1 illustrates the topology of each of these networks. For clarity, the figure shows only one of the random links per node and omits links in the third dimension for the SW 3-D hexagonal torus.

The split between the number of regular versus random links has the potential to yield significantly different properties for these topologies. A topology such as SW-Ring, with a relatively high number of random links and a smaller number of regular ones, can render the topology better suited for data-center-wide search, while the opposite weighting, as in the case of SW-3DHexTorus, can yield better performance when searching local nodes. In essence, this breakdown represents the tradeoff between local expertise versus better global knowledge. We show in Section 5 that having strong local connections yields better results when using greedy routing, while shortest path routing algorithm works well with a larger number of random links.



(a) SW Ring (b) SW 2-D Torus (c) SW 3-D Hexagonal Torus

Figure 1: Small-world datacenter topologies. Solid lines indicate regular links while dotted lines represent random links. Only one random link per node is drawn and links in the 3rd dimension in SW-3DHexTorus are omitted for simplicity.

Port #	SW-Ring	SW-2DTorus	SW-3DHexTorus
1	right (2)	east (2)	top (2)
2	left (1)	west (1)	bottom (1)
3	random (3)	north (4)	east (4)
4	random (4)	south (3)	west (3)
5	random (5)	random (5)	north or south (5)
6	random (6)	random (6)	random (6)

Table 1: Examples of port assignments for SWDC topologies. Table entries describe the relative server location and the destination port for each source port.

3.2 Physical Structure

Since routing in SWDCs is performed by the server nodes that comprise the network, there is no need for any additional expensive switch hardware. The functionality of top-of-rack, access, aggregation and core switches, which often require expensive interconnect technologies and/or pose bottlenecks due to oversubscription, are effectively distributed throughout the servers. Because of this redistribution of costs, SWDCs require every node to possess multiple network cards.

To provide a direct comparison to past work, this paper focuses on SWDC topologies with degree 6. We name ports from 1 to 6 in each server and consistently use the port with same number to connect servers in the same relative direction in the network. Since connections are bidirectional, both ends of a link always connect the predefined ports. For example, link from port 1 of a server always connects to port 2 of the adjacent server on the right in the lattice network of SW-Ring and vice versa. Table 1 describes examples of port assignments for each SWDC topology. This static assignment of ports simplifies the assignment of logical addresses, which we explain in the next subsection.

3.3 Geographical Identity and Address Assignment

Because small-world networks are built on top of a lattice, the network lends itself easily to address assignment based on topology. We call this new address g_k of a server k its *geographical identity*. For a SW-Ring, a single index corresponds to the position of the server on a ring; for a SW-2DTorus, two indexes corresponding to x and y coordinates of a server in 2-D torus; and for a SW-3DHexTorus, three indexes corresponding to x, y, and z coordinates of a server in the 3-D hexagonal torus network form the geographical identity.

When the network port connections on all servers consistently conform to the wiring configuration described in Table 1, the servers can autonomously determine their geographical identities with minimal human intervention. The network administrator needs to only define the geographical identity of a single landmark server. The rest of the servers can determine their geographical identities through the regular links, by simply determining which, if any, of their neighbors have an address assigned, and over which regular link they are connected to the neighbor with the assigned address. This process can also discover and identify wiring errors through simple sanity checks. Once all the geographical identities are set in a wave over the regular links, the servers can easily figure out which servers they can directly reach through the random links.

The regular lattice structure that underlies SWDC lends itself naturally to efficient automated tools for monitoring the network and detecting failures. Miswirings are easy to detect through a simple lattice consistency checker at construction time, and efficient failure detectors can be employed on the lattice throughout production use.

Note that it is not strictly necessary to use geographical identities in a small-world network; the topology provides benefits even if the address assignment is performed using opaque, de facto MAC addresses. However, the regular structure of the network, coupled with geographical identities, enables SWDCs to facilitate more efficient datacenter applications. Specifically, geographical routing and addressing APIs, proposed in CamCube [1], can enable datacenter applications, such as key-value stores, map/reduce operations and coordination protocols, to be implemented more efficiently.

3.4 Routing Methods

Since the geographical identities assigned in the previous phase identify the rack location of each of the servers, they enable a range of efficient routing strategies to be used in SWDCs. Two interesting choices for geographical routing in a datacenter are to (1) find the optimal shortest paths for all destinations, or (2) route the packets through a less-expensive greedy approach. We discuss these design options in turn and identify their benefits and inherent costs.

3.4.1 Optimal Shortest Path Routing

One possible choice of a routing scheme in a SWDC is for all nodes to compute the shortest optimal path to every destination. A node may use a well-established protocol, such as OSPF [29], to discover the shortest path to all destinations. Precomputing shortest paths will minimize the path stretch that is possible with a given topology, reduce congestion, and thus increase bandwidth.

There are a few potential downsides to computing the optimal shortest routing table. First, every node needs to maintain a routing table whose size is linear to the size of the datacenter. Given the size of typical datacenters (in the 100,000s of machines, or fewer), and the amount of memory typically found on server nodes, these requirements (at 1MB for 10K servers) are quite modest. Second, running OSPF may incur a bandwidth cost, but given the relatively low rates of mobility and failure in the datacenter and the high bandwidth interconnect, these overheads are miniscule. Finally, the primary cost of precomputing an optimal routing table is the TCAM cost of storing that table on the daughterboard that holds multiple NICs. While the regular structure of a SWDC can significantly reduce the size and cost of the TCAM required, these costs can potentially be high. For this reason, we investigate a simpler, greedy alternative for the routing layer.

3.4.2 Greedy Geographical Routing

Greedy geographical routing determines the next hop for a packet based solely on the packet's current location and its destination. Assume that a distance is the minimum network hop count between the two server nodes, the destination server node of a packet is dst, the server currently holding the packet is $curr$, and the set of all next hop neighbors of $curr$ is $N(curr)$. The regular network pattern in the small-world network always enables the routing algorithm to find a server $n \in N(curr)$ that has smaller distance from n to dst compared to that from $curr$ to dst. This ensures that the greedy routing in small-world network guarantees packet delivery. The random links help reduce the distance more dramatically in a single step.

This geographical routing method only requires each server to keep track of the identities of its directly reachable servers. The computation for the routing consists solely of numerical address comparisons, which can be efficiently implemented in hardware. However, to maximize the use of potential shortcuts to the destination, we propose a slightly modified approach wherein every node discovers the identities of its neighbors up to k hops away. This enables every node to make effective use of the random links of its neighbors to degree k. Our experiments show that for $k = 3$, which requires maintaining $\sum_{i=1}^{3} 6^i = 258$ identities per server, this scheme yields short routing path lengths. Because greedy routing, when coupled with this enhancement, requires only a small, fixed number of computation and comparisons, this approach lends itself well to a hardware implementation.

3.4.3 Routing with Failures

The large scale of a modern datacenter guarantees that there will be failed nodes somewhere in the network at all times. Geographical routing in the presence of failures is a well-studied topic, and due to space constraints, we do not discuss potential remedies in detail except to note that the regular structure of the underlying grid greatly simplifies routing around failed nodes. Techniques such as face routing [20] are well-established for routing around failures. Compared to a conventional datacenter, the high interconnectivity and in-degree make it far less likely for an SWDC to be partitioned.

3.4.4 Interaction with Other Network Protocols

Throughout this paper, we assume that we use deterministic single-path routing to avoid packet reordering and to provide compatibility with other network protocols such as TCP. However, SWDCs have multiple paths with the same length that connect the same node pairs. Therefore, a multipath routing protocol that takes advantage of the presence of alternative paths can further improve the performance of SWDC.

3.5 Composability and Scaling

Wiring up a datacenter is a costly undertaking. Intricate topologies based on complex connection patterns can require extensive human effort. In contrast, SWDCs are based on relatively simple blueprints. The underlying grid topology requires a very regular, mostly localized wiring pattern. We focus this discussion on the wiring of the random links, which are responsible for the unique properties of SWDCs.

Random point-to-point links are feasible even for relatively large datacenters. Our discussion above alluded to point-to-point wiring without repeaters. Such deployments are feasible with commercial off-the-shelf technology. Gigabit ethernet links can be 100 meters long. Consider a datacenter with 10K servers in standard 42U Racks. Usually the width, depth and height of the rack are 0.6, 1, and 2 meters respectively [19]. Thus, even if we locate 10K server in racks of 16 by 16 grid with 1.5 meters distance in-between, a 100 meter link can connect any servers in the datacenter through the ceiling or the floor. For installations above these dimensions, one can employ repeaters on long paths.

The resilience of the small-world network tolerates many kinds of errors in the construction of the datacenter. The crew can simply be provided with wires of appropriate lengths and instructions to extend the wires completely and pick a random port. Small-scale, one-off errors will have negligible impact on the performance of the resulting network. If wires cannot be precut, the construction crew can be sent with spools and a handheld random number generator application, in lieu of strict blueprints, to wire up a datacenter.

Adding new nodes in an SWDC requires some care to ensure that the resulting random graph has good routing properties. Establishing the new regular links is straightforward, as they simply need to be wired to establish a ring or torus. Once the regular links are handled, we run the random algorithm to determine the random link destinations for the new nodes. To connect the random links while preserving the small-world characteristics, a pair of new nodes should work together in each step. First, we unplug the existing random links from the destination pairs and connect the random links from the new node pairs to the unplugged ports. Now there are two nodes that were connected to the destination pairs and have unconnected random ports. We simply connect the ports of these two nodes. Algorithm 1 describes the process of connecting the random links.

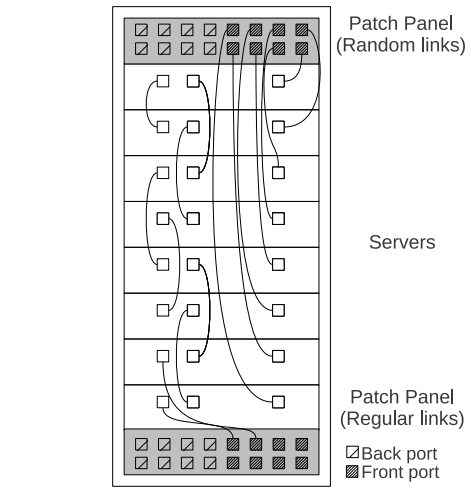
The self-similarity of small-world networks provides many opportunities for scaling datacenters easily. A compelling scenario is to construct datacenters from pre-fabricated small-

Algorithm 1 Random link construction when adding new node pairs to SWDC.

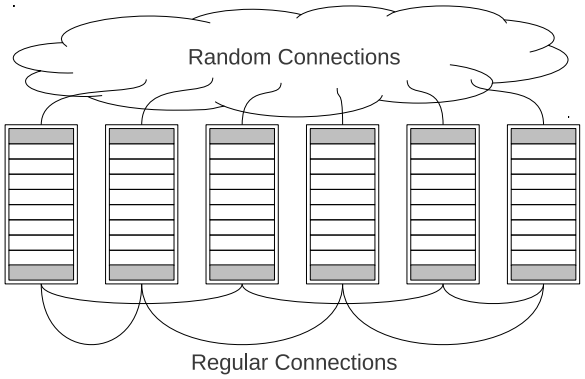
Require:
Functions
$Port(S, n)$: returns nth port of server S.
$EPort(S, n)$: returns other end of the link connected to $Port(S, n)$.
$Disconnect(P)$: disconnects wires from all port $p \in P$.
$Connect(p_1, p_2)$: connects wires between ports p_1 and p_2.
$RandSel(S)$: returns random server which is selected with decaying probability in the dth power of distance from server S, where d is the dimension of the lattice network.
Variables
$NewPairs$: set of new server node pairs to add to SWDC.
$RandPortNum$: set of port numbers connecting random nodes.
Ensure: Add random links to new nodes in small-world datacenter
for all $p \in NewPairs$ **do**
$S_1 = p.first$
$S_2 = p.second$
for all $r \in RandPortNum$ **do**
$D_1 = RandSel(S_1)$
$D_2 = RandSel(S_2)$
$sp_1 = Port(S_1, r)$
$sp_2 = Port(S_2, r)$
$dp_1 = Port(D_1, r)$
$dp_2 = Port(D_2, r)$
$ep_1 = EPort(D_1, r)$
$ep_2 = EPort(D_2, r)$
$Disconnect(\{dp_1, dp_2, ep_1, ep_2\})$
$Connect(sp_1, dp_1)$
$Connect(sp_2, dp_2)$
$Connect(ep_1, ep_2)$
end for
end for

world racks. In such a scenario, each rack would have the regular lattice framework pre-wired. One way to perform this wiring easily, without the need to resort to long wires that span across the full length of the datacenter, is to connect each host not to the physically adjacent host, but to alternate and double back at the edge of the network. Thus, instead of wiring five nodes in a 1-2-3-4-5-1 pattern, where the 5-1 connection would have to be very long, the nodes can be wired in a 1-3-5-4-2-1 pattern, where every wire that is part of the regular lattice is approximately the same length, and no wires cross the datacenter. It is possible to create highly scalable, reusable, componentized racks with the aid of patch panels placed in each rack. One edge of the regular lattice would connect to the patch panel, enabling multiple racks to be easily patched into a string, and the random links emanating from each rack to other racks would be routed to the patch panel as well. Random and regular links can use different patch panels for ease of maintenance. Once racks are internally wired, datacenter administrators can wire the back ports of patch panels among racks to complete the construction of the network. Figure 2 illustrates a simplified example based on the SW-Ring structure.

Connecting a SWDC to the outside world can be performed with the aid of switches. The full connectivity provided by the SWDC enable any number of switches to be connected to any location within the datacenter. However, in the rest of this paper, we focus exclusively on the network connection and traffic inside datacenters, since the traffic internal to datacenters accounts for more than 80% of the total traffic even in client-facing web clouds [12].



(a) Wiring within a rack



(b) Wiring among racks

Figure 2: Wiring SW-Ring using patch panels.

4. CONTENT ROUTING IN THE SMALL-WORLD

One of the strengths of SWDC is that the physical network structure can correspond to the logical network structures of the applications. This enables efficient content addressing, and enables applications such as distributed hash tables (DHTs) [16, 11], multicast, aggregation and many others. Conventional hierarchical networks are not a good substrate for content addressing. Consider the case of deploying CAN [34] in the datacenter. Every communication would be subject to layers of processing through the standard communication stack. All packets would have to go through multiple layers of switches to get to their destination because the physical location and connection of nodes would not correspond to the logical structure of the system. Thus, adjacent nodes in the virtual space can require packets to travel through the whole hierarchy of switches for data exchange. Although the switches provide fast forwarding of packets to far nodes, the top level switches can pose bottlenecks to the overall bandwidth.

In contrast, the physical topology of SWDCs can mirror the logical structure of a multidimensional space, simplifying naming and routing. Nodes can be addressed not by an opaque IP address, but by an assigned geographical iden-

Topology	# Nodes	Switching Delay	Links per server	Note
SW-Ring SW-2DTorus SW-3DHexTorus CamCube (3D Torus)	10.24K	64B Pkt: 4.3 (15.8) μs 1KB Pkt: 11.7 (150.0) μs	1 GigE × 6	NetFPGAs are used as default. Numbers in parentheses are for cases without NetFPGA support.
CDC (2,5,1) CDC (1,7,1) CDC (1,5,1)	10.24K	TOR: 6 μs, AS: 3.2 μs, CS: 5 μs	1 GigE × 1	Delays are independent of packet sizes. 10 GigE links are used among switches.

Table 2: Base configurations for SWDC, CamCube, and CDC. SWDCs and CamCube have 6 1GigE ports on each server and NetFPGA accelerates the switching performance. 3 level switches with different oversubscription rates connect CDC nodes.

tity or other customized addressing scheme that captures their location. Since this location corresponds to the node's logical position, this enables data-addressing: a key-value pair can be routed to the responsible home node by key, for example. Content routing in the datacenter was proposed and examined in depth by CamCube, from which we derive our motivation and the API for content addressing. Unlike CamCube, which relies solely on a rigid torus structure for its routing, the additional random links in a SWDC provide significant shortcuts that can greatly improve performance and distribute load.

5. EVALUATION

In this section, we further explore and evaluate the characteristics of SWDCs including network hop counts, latency and bandwidth, load distribution, and fault tolerance in comparison to CamCube and conventional hierarchical datacenter (CDC) models.

5.1 Test Environment

We evaluate the performance characteristics of SWDCs through simulations of realistically-sized datacenters. We use a fine-grained packet-level simulator that considers delays in links, servers, and switches, reflects link and buffer capacities, and models topology and routing protocols. The simulator can measure evaluation metrics such as packet delivery latency, network bandwidth, and network hop count.

We compare SWDCs to conventional datacenters (CDCs) as well as CamCube, targeting datacenters with 10,240 servers. CDCs may have different oversubscription rates in each level of switches. The oversubscription rate x represents the bandwidth ratio $x:1$ between the lower and upper level switches. We consider three levels of switches that connect 10,240 servers, and we use three numbers to indicate the oversubscription rate in each level: top of rack switch, aggregation switch, and core switch. For example CDC (2,5,1) indicates that the oversubscription rates in top of rack, aggregation, and core switches are 2, 5, and 1, respectively.

Table 2 summarizes the base configurations we use for each topology. For SWDCs and CamCube, we use NetFPGAs to accelerate the switching performance. SWDCs and CamCube servers have limited parallelism to serve multiple ports, but switches in CDCs are capable of serving all ports simultaneously as long as there is no contention within the internal switching logic. The delays in the table are based on [9, 2, 6, 5, 7]. Unless otherwise stated, we use greedy routing in SWDCs. We briefly compare the performance with other combinations of hardware settings and routing methods in a later subsection. We do not tackle cost issues in this paper because semiconductor costs are highly dependent on economies of scale and tend to fall rapidly, sometimes exponentially, over time. A hypothetical cost discussion would necessitate market analysis, a topic beyond our expertise, though we do note that the cost of the central component, the NetFPGA, went down four-fold during the period that this paper was under review.

In keeping with prior work in the area, we use three synthetic benchmarks and employ different packet sizes for evaluation:

- **Uniform random**: Source nodes select random destination nodes in datacenters with uniform random probability. This resembles the shuffle phase of a MapReduce task with uniform distribution.
- **Local random**: Source nodes select random destination nodes within a cluster with uniform random probability. A cluster in SWDCs and CamCube consists of 640 server nodes that are neighbors in the lattice network. For CDCs, a cluster is the server nodes under same aggregation switches (2nd level). In comparison to the uniform random workload, this workload better characterizes a real datacenter application because most of the applications generate traffic that travels short distance from the source [3, 25]
- **MapReduce**: (1) A source node sends messages to the nodes in the same row of its rack. (2) The nodes that receive the messages send messages to the nodes in the same columns of their racks. (3) All the nodes that receive the messages exchange data with the servers in the same cluster and outside the cluster with 50% probability each. This benchmark resembles the MapReduce application used in Octant [38], where server nodes compute and exchange information with other nodes during the reduce phase.

5.2 Path Length Analysis

Before evaluating the performance of running applications in SWDCs, we first evaluate the network path length, a key factor that influences network latency and bandwidth.

5.2.1 Dijkstra Path Length

To characterize shortest possible path lengths under different datacenter topologies, we first determine the shortest paths from each node to the others using Dijkstra's algorithm [10], and take the average of this metric across all nodes. Since CDCs only have paths of length 2, 4, and 6, where the paths involve climbing up and down up to three levels of switching hierarchy, the discussion below focuses on SWDCs and CamCube.

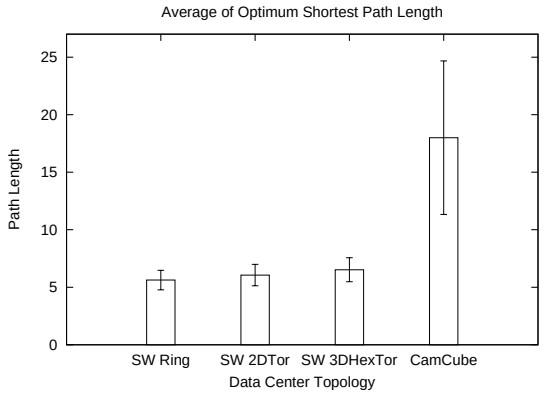


Figure 3: Dijkstra path length. Topologies with larger number of random links have shorter path length.

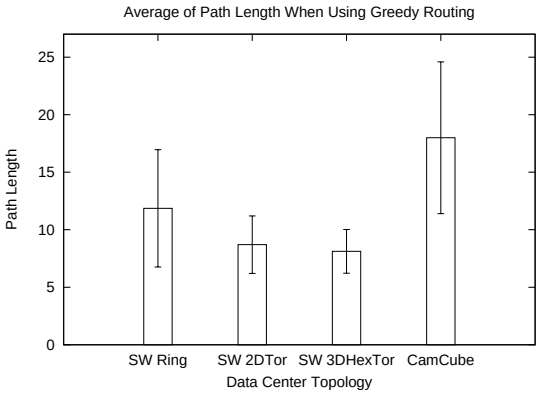


Figure 4: Greedy path length. SWDC topologies based on lattices of larger dimensions achieve shorter path lengths.

Figure 3 compares the Dijkstra path lengths achieved by small-world networks to CamCube. The Dijkstra path lengths in SWDCs are 2 to 3 times shorter, on average, than CamCube. Among SWDCs, the topologies with larger number of random links achieve shorter path lengths. Surprisingly, path lengths in SWDCs are comparable to conventional datacenters. This implies that, depending on the performance of the switching logic, SWDC can provide comparable or better network service than CDCs.

5.2.2 *Greedy Path Length*

To evaluate the realistic path lengths that SWDCs would achieve when employing the cheaper greedy routing technique, we measure the path length between all nodes when using greedy routing with uniform random traffic on each topology. Each node using the greedy routing method is aware of the geographical identities of its three hop neighbors.

Figure 4 shows the resulting greedy path length. CamCube achieves almost identical results as the Dijkstra path length experiment since greedy routing leads to the shortest path in a 3D torus. However, all SWDCs topologies achieve shorter greedy path lengths than CamCube.

Since each SWDC nodes use a small amount of information for routing, the path length increases compared to the optimal Dijkstra path lengths which are determined using global information. Whereas topologies with larger number of random links achieve shorter Dijkstra path lengths, this trend is reversed for greedy paths. SW-Ring's greedy path length is significantly higher than its Dijkstra path length, because greedy routing is unable to fully take advantage of random links with limited local information. On the other hand, SW-3DHexTorus's greedy path lengths are only moderately longer than its Dijkstra path length because the higher dimension lattice network provides relatively shorter paths when random links are unavailable.

5.3 Packet Delivery Latency

Because the average path lengths of SWDCs exceed that of CDCs, one can expect the latency of a SWDC to be larger than CDCs. Figure 5 shows the average packet delivery latencies for comparable SWDC, CDC and CamCube configurations. Each server in this evaluation generates 300 packets per second and MapReduce traffic launches multiple jobs simultaneously to fulfill the packet rate.

The measured packet delivery latencies reflect the trends of the path lengths discussed in the previous subsection. The path length partially accounts for the higher latency of SWDCs, but the performance of the switching logic also plays a significant role.

While the packet switching latency in SWDC varies depending on the packet sizes, datacenter switches in CDCs have short constant latencies due to cut-through routing and can process multiple packets at the same time. Therefore, the latency gap between SWDCs and CDCs becomes larger when using 1KB packets compared to using 64B packets. When using uniform random 1KB packets in SW-3DHexTorus, the average latency is 5.3 times greater than the latency of CDC(1,5,1).

Interestingly, SWDCs achieve lower latencies than CDCs for certain benchmarks. In the MapReduce benchmark with 64B packets, the benchmark generates packets for each phase of MapReduce in a short time interval and these packets are subject to queuing. CDC servers have one network port per server whereas SWDC servers have six ports. Therefore, the wide bisection bandwidth helps SWDC to minimize the queuing delays and outperform conventional datacenters for this benchmark.

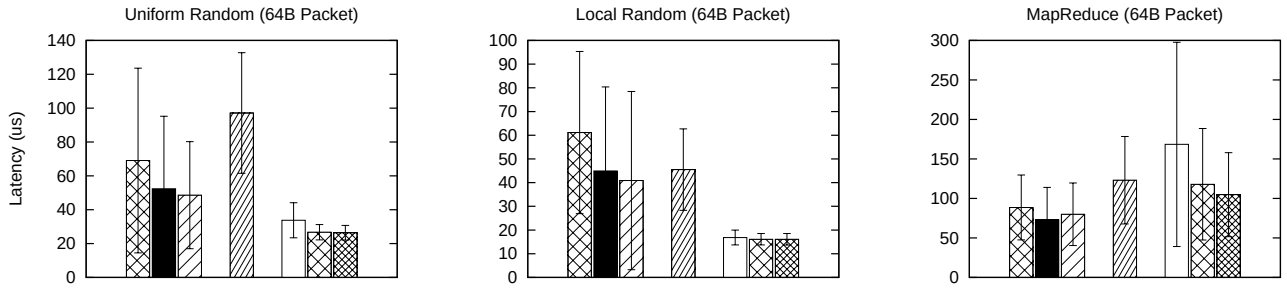
SWDCs generally exhibit lower latencies than CamCube. The only cases that CamCube outperforms SWDCs – only for SW-Ring and SW-2DTorus – are when using local random benchmark. This is because the packets from the benchmark travels short distance and SW-Ring and SW-2DTorus have lower dimension lattice than CamCube.

To summarize, SWDC outperforms CamCube for most cases. Due to longer path lengths and increased switching delays, SWDCs typically show larger latencies than CDCs.

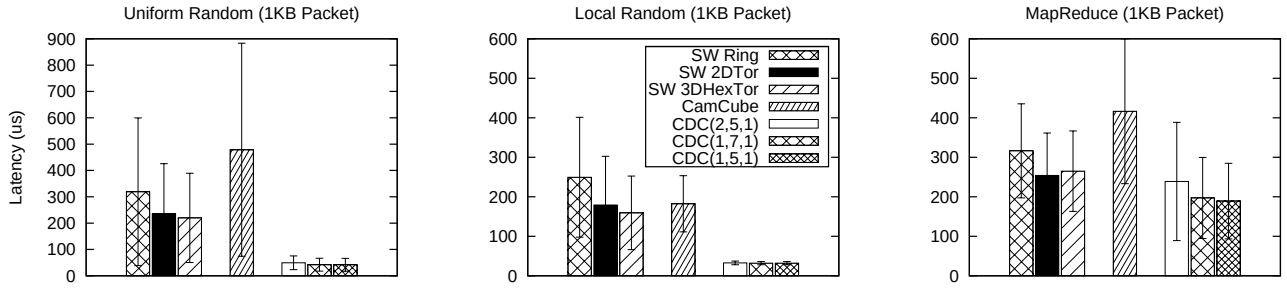
5.4 Maximum Aggregate Bandwidth

To measure the maximum aggregate bandwidth, we measure the aggregate bandwidth of the network when every node pair is sending a burst of 500 packets. The results for the different benchmarks are collated in Figure 6.

Although the packet latencies of SWDCs were larger than CDCs for most cases, SWDCs achieve higher or comparable maximum aggregate bandwidth than CDCs. For the 64B packet workloads (Figure 6(a)), SWDC outperforms CDC

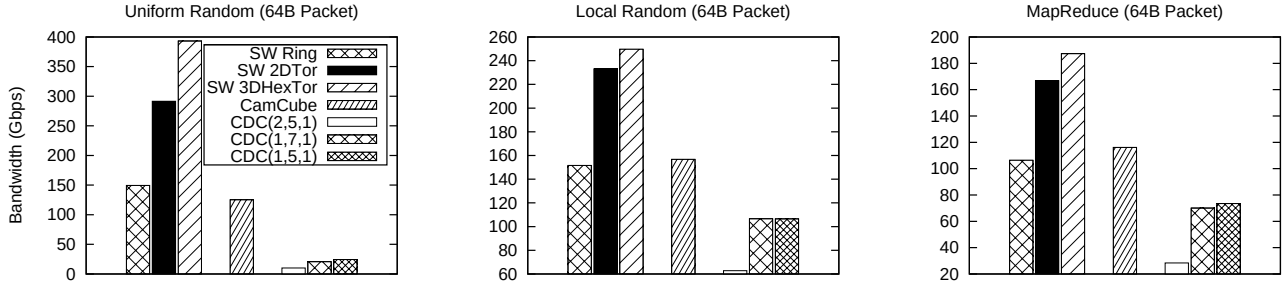


(a) Using 64B packets. SW-3DHexTorus always outperforms CamCube and CDCs suffers from queuing delay for MapReduce.

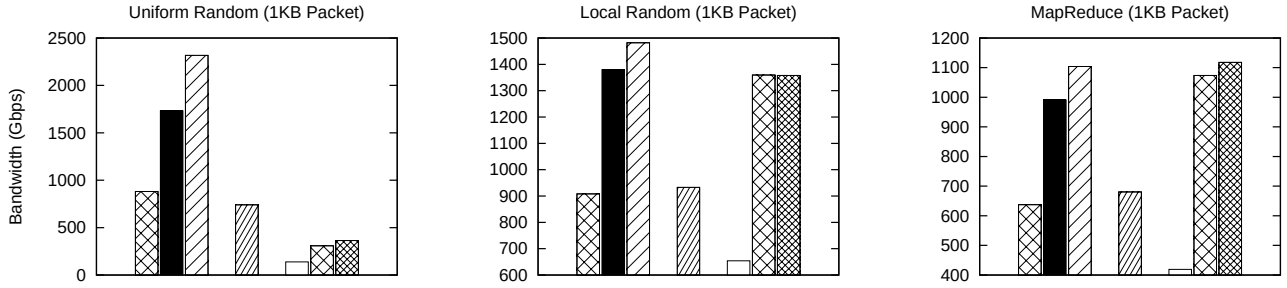


(b) Using 1KB packets. Absence of physical switches in SWDCs makes larger latency gaps between SWDC and CDCs.

Figure 5: Average packet delivery latency. While relatively longer path lengths and larger switching delays pose problems for SWDCs compared to conventional datacenters, they achieve comparable delays for the MapReduce benchmark.



(a) Using 64B packets. SW-2DTorus and SW 3DHexTorus outperforms CamCube and CDCs for all cases.



(b) Using 1KB packets. SW-3DHexTorus shows 1.5% less bandwidth for MapReduce than CDC, but outperforms for all cases.

Figure 6: Maximum aggregate bandwidth. SWDCs achieve high bandwidth due to the additional, mostly-disjoint paths provided by the random links.

for almost all configurations. There are several reasons for this: the SWDC overhead for switching small packets is small; the number of network ports per server in SWDC is greater than CDC; and the bandwidth of CDC is bounded by the top level switches. The bandwidth of SW-3DHexTorus is, at its maximum, 1600% greater than CDC(1,5,1) for the uniform random benchmark. The more the benchmark limits the packet travel distance, the smaller the bandwidth gap between SWDC and CDC becomes. This is because SWDC is effective at transferring packets to far away nodes and CDC is suitable for sending packets locally without going up and down the oversubscribed network hierarchy.

SW-2DTorus and SW-3DHexTorus outperform CamCube for all cases due to their shorter path length. The bandwidth of 3DHexTorus is 312% larger than that of CamCube for the uniform random benchmark while SW-Ring achieves comparable performance.

For large packet workloads (Figure 6(b)), CDCs provides comparable bandwidth to SWDCs for local random and MapReduce benchmarks. This is due to the minimal utilization of random shortcuts in SWDCs and limited usage of the top level switches in CDC. However, SW-3DHexTorus demonstrates bandwidth that is higher than or comparable to CDCs for both benchmarks and higher bandwidth for uniform random benchmarks: SW-3DHexTorus shows 1.5% lower bandwidth for MapReduce benchmark, but 637% higher bandwidth for uniform random benchmark than CDC(1,5,1). CamCube generally shows lower bandwidth than SW-2DTorus and SW-3DHexTorus: SW-3DHexTorus operates at maximum 314% greater bandwidth than CamCube. CDC(2,5,1) performs the worst for all cases due to oversubscription in two levels of switches, namely, top of rack and aggregation switches.

Overall, SWDCs generally provide higher bandwidth than CamCube. They also provide bandwidth that is higher than or comparable to CDCs. Similar to the latency results, high dimension lattice networks such as SW-3DHexTorus lead to higher bandwidth. Although the packet latencies are greater than CDCs, SWDC is appealing to applications where communication across datacenter is frequent and requires high bandwidth.

5.5 Hardware Acceleration

In this subsection, we briefly compare the performance of SWDCs and CamCube with different hardware and routing configurations. We compare the latency and bandwidth of our base case using NetFPGA and greedy routing with the cases of using shortest path routing with and without NetFPGAs. Although we only show the cases for the uniform random benchmark with 1KB packets in this paper, the performance of other SWDC configurations shows qualitatively similar results.

Figure 7 illustrates the latency of SWDCs with and without hardware assistance for inter-NIC routing. The forwarding latencies are at least three times larger without hardware acceleration. In CamCube, the effect of switching delay is more obvious since the packets travel through larger number of hops. Routing purely in software effectively transforms the node into a store-and-forward platform, where the forwarding can take effect only when the packet has been received in full, and thus the switching delay is highly dependent on the packet size [2]. Hardware accelerators such as NetFPGAs or GPUs provide a performance boost by mitigating this effect. If shortest path routing is used in SWDCs in conjunction with NetFPGA assistance, we can achieve significantly lower latencies.

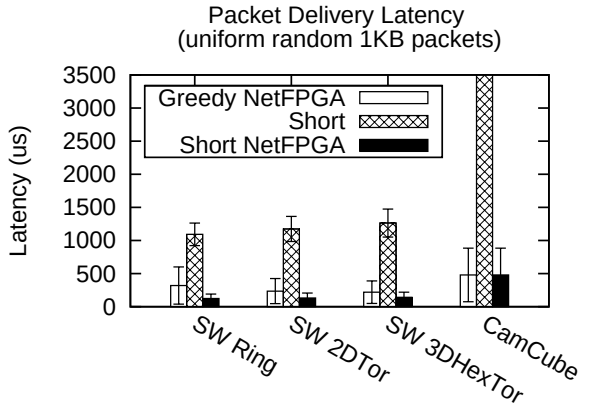


Figure 7: Latency of SWDC and CamCube with different hardware and routing configurations.

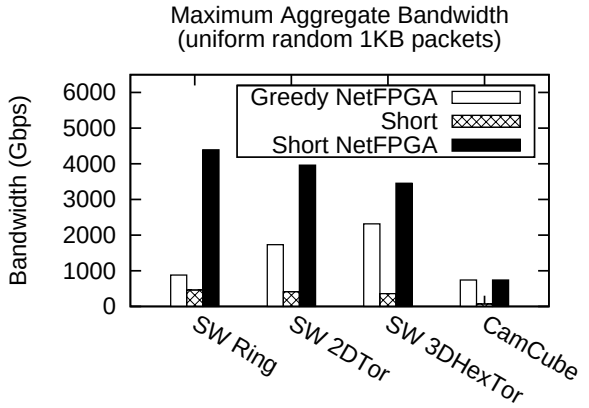


Figure 8: Aggregate bandwidth of SWDC and CamCube with different hardware and routing configurations.

Figure 8 shows that bandwidth measurements show similar trends. Our base case with hardware acceleration enables SWDC to achieve 1.9 to 6.5 time higher bandwidth than shortest path routing without NetFPGAs. Again, shortest path routing and NetFPGA combinations yield the best results. SW-Ring performs the best for this case since the Dijkstra path length is the shortest as shown in Figure 3.

Both the latency and bandwidth comparisons demonstrate the need for hardware acceleration. Further, in environments with abundant CPU resources, the use of shortest path routing will lead to further latency and bandwidth improvements.

5.6 Load Distribution

Compared to CamCube and CDC, SWDC utilizes random links as shortcuts. Thus, packets may overload certain links or servers. Thus, we analyze the traffic load on each server under different benchmarks for SWDC and CamCube to measure the amount of average packet load per server and the load distribution. Figure 9 illustrates the average

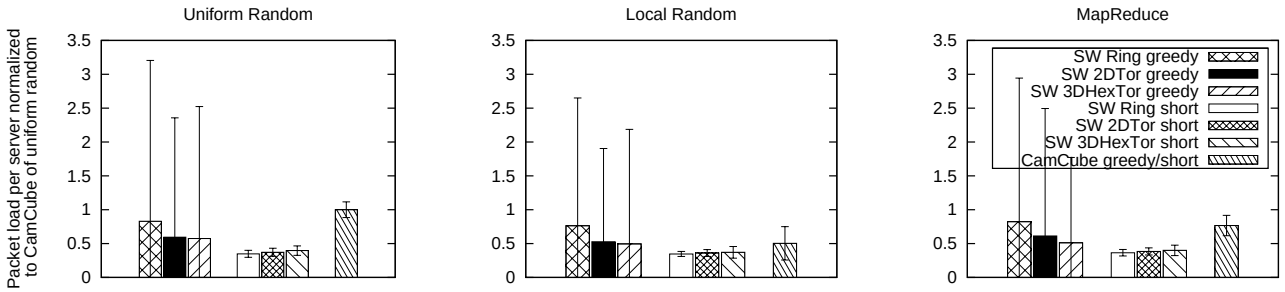


Figure 9: Packet load distribution. SWDC with greedy routing imposes less overall load than CamCube.

and the standard deviations of the numbers of packets each server processes. We normalize the bars to CamCube with the uniform random benchmark. We also include the results of using shortest path routing for comparison.

The average number of packets processed per server follows the trends of path length. The longer the path that a packet travels, the larger the number of servers that processes the packet. Therefore, for all benchmarks, SW-3DHex-Torus processes the same job with the least amount of work among all others – excluding the shortest path routing cases – and CamCube shows better performance than SW-Ring for local random benchmarks.

However, greedy routing in SWDCs can lead to load imbalance under certain circumstances. Because greedy routing maintains no global state and picks only locally optimal routes, it may end up favoring small number of random links. On the contrary, an optimal routing algorithm will select the random links with the balanced random distribution, similar to how random links were built, based on the global information of the network. This explains the relatively high standard deviation bars in Figure 9. This imbalance represents a tradeoff between performance and the simplicity of the routing algorithm. SWDCs still complete the benchmarks with less total overhead than CamCube in most cases.

The experiment also shows how much a better routing algorithm can improve the load balance and performance of a small-world network. As shown in the second set of bars, both the amount of load and its distribution are all better when shortest-path routing is employed. There are other techniques, such as pseudo-randomization when making greedy decisions and routing packets adaptively based on link load counters, that can be employed to improve the performance of greedy routing.

5.7 Fault Tolerance

The failure characteristics of SWDC and CamCube are different from CDCs. Switch failures can disconnect many nodes in CDCs, while SWDC and CamCube, which are not dependent on a switch fabric, exhibit tremendous failure resilience. Since each server node acts as a switch in SWDC and CamCube, node and link failures can cause path disruption, but the abundance of alternative paths often provide alternative paths for flows. To evaluate the fault tolerance characteristics, we simulate random node failures and measure the path lengths and connectivity among live nodes.

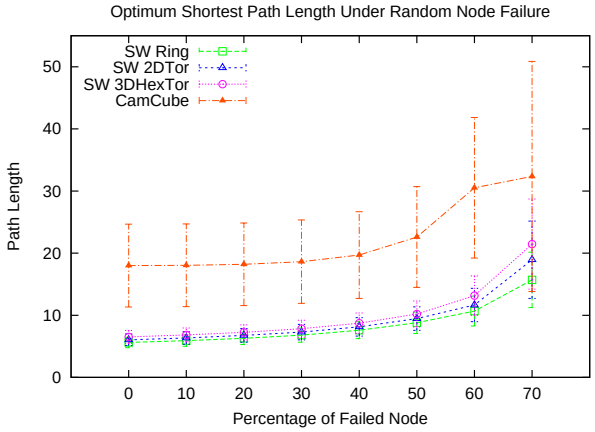


Figure 10: Path length under failure. The increase in path length under failure is moderate.

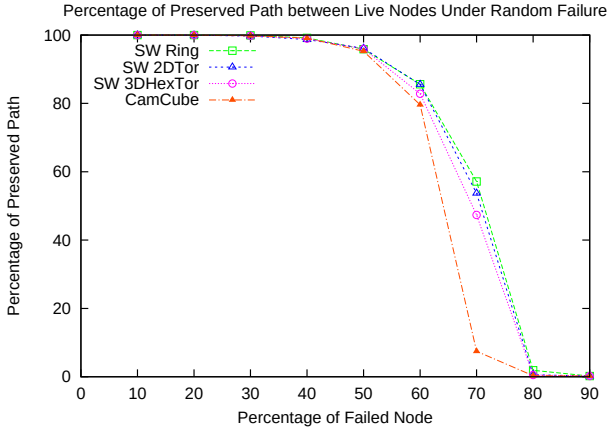


Figure 11: Connectivity under failure. SWDCs can maintain strong connectivity in the presence of many failures.

Figure 10 shows the average path length to connect all live node pairs. For both SWDC topologies and CamCube, the path stretch increases gradually as the percentage of failed node increases. Because SWDC and CamCube topologies are degree 6 graphs, where servers can easily forward packets through alternative routes, random node failure does not greatly affect network performance. The amount of increase

in path stretch is greater for the topologies with fewer number of random links.

The percentage of preserved connections under failure also shows similar trends (Figure 11). All topologies maintain over 90% of connections until 50% of nodes fail. SWDCs maintain more connections than CamCube, while SW-Ring, which has the largest number of random links, keeps the greatest number of connections under equivalent failure conditions.

5.8 Summary

The summaries of the findings throughout this section are as follows:

- **Path length**: Average path lengths in SWDCs can be less than half of CamCube. The random links in the topology contribute to the reduction of the shortest path lengths, while the regular links impact the greedy path lengths.
- **Latency**: Average packet delivery latencies of SWDCs is less than CamCube but greater than CDC. Depending on the benchmark, SWDCs can achieve smaller latencies than CDC by taking advantage of multiple network ports.
- **Bandwidth**: Depending on the application and packet sizes, SWDCs' maximum aggregate bandwidth can be 6 to 16 times greater than CDC and 3 times greater than CamCube.
- **Hardware support**: Hardware acceleration is critical to providing low latency and high bandwidth in SWDCs.
- **Load distribution**: Greedy routing can cause uneven distribution of load in SWDC. The load to process the same job on SWDCs is typically less than in CamCube.
- **Fault tolerance**: SWDCs can maintain over 90% of connectivity among nodes until 50% of total nodes fail. The random links contribute to stronger connectivity and node failures degrades the path length between node pairs very moderately.

The simulations in this paper use a fixed number and degree of nodes, which reflect realistic hardware constraints. Past work on mathematical models of small-world networks [18, 33, 26] provides insights on how SWDCs can scale and perform when these fixed parameters are changed.

6. RELATED WORK

Small-world phenomena and networks have been studied since 1960s. Kleinberg provides extensive theoretical analysis and a survey of past work on small-world networks [21, 22]. The theoretical aspects of SWDCs build on this work.

Small-world networks provide the foundation for certain distributed hash tables and peer-to-peer systems. Freenet [40] and Symphony [31] provide mechanisms to achieve small-world properties in wide-area networks. Viceroy [28], a peer-to-peer DHT, builds a butterfly network inspired by small-world networks. The Meridian [37] and Cubit [36] systems create lightweight overlays in wide-area networks where each node is an expert for other nodes and data in its own region, and possesses just the right number of outgoing links to far regions to perform efficient routing. Our design differs from this work in that we propose to use small-worlds for the actual physical topology of a datacenter, as opposed to the structure of an overlay.

More recently, a large body of work has emerged on how to improve bisection bandwidth, failure resilience, and latency of datacenter networks through innovative topologies. VL2 [12] uses flat addressing to separate names and locations of servers and performs load balancing and flexible routing to provide each server with the illusion of a large level 2 switch over the entire datacenter. This structure enables VL2 to support efficient content-addressing. Unlike SWDC, VL2 depends on an expensive switched fabric. Further, since the physical location of addresses are hidden from application programmers and the network is complex, the costs of data access are nonuniform and opaque to application programmers. BCube [13] and DCell [14] are similarly novel datacenter topologies that are based on a hierarchical cell organization. Both approaches aim to reduce path length while improving bisection bandwidth through a rich inter-cell connection network. They require a complex regular network pattern, and depend on numerous commodity switches. SWDC requires a relatively simple regular, switch-free point-to-point network and random links can be wired with some imprecision. Scafida [15] is another datacenter network architecture inspired by scale-free networks [4]. Similar to SWDC, it uses randomness to connect nodes. However, due to lack of regularity in the network, Scafida cannot efficiently support content-based and greedy routing. Perhaps the closest work to ours is CamCube [1], a content-addressable network based on a 3D torus. Simple regular connections and APIs that expose the topology enable application programmers to program content-addressable applications. However, the torus topology suffers from large network hop counts as we have shown in this paper. SWDC can provide the same APIs as CamCube while improving performance through its random links.

The software switching techniques used in SWDCs are similar to those in RouteBricks [9] and the Linux software router [2] in a small scale. PacketShader [17] and ServerSwitch [27] accelerate software switches and routers using hardware such as GPUs and NetFPGAs. These techniques are applicable to SWDCs as well.

7. CONCLUSIONS

In this paper, we introduced small-world datacenters, a datacenter topology based on small-world networks comprised of a regular short-distance lattice amended with longer-range random links. Such a topology is easy to wire, composes and scales well to large datacenters, supports content addressing, and achieves higher or comparable performance compared to both conventional and recently proposed network designs. SW-2DTorus and SW-3DHexTorus outperform CamCube for most of the evaluation metrics due to random links that provide efficient cut-through routes across the datacenter. Although SWDCs exhibit higher latencies than conventional datacenters, they achieve roughly an order of magnitude higher bandwidth. The latency gap between SWDCs and CDC stems from the packet processing efficiency in servers versus in switches. As software routing technology advances, we expect SWDC to become more competitive on the latency front while providing higher

bandwidth, greater failure resiliency and a more flexible addressing framework.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback, and Anthony Rowstron for insightful discussions on efficient wiring in datacenters. This material is based upon work supported by the National Science Foundation under Grant No. 0546568.

8. REFERENCES

[1] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, and A. Donnelly. Symbiotic Routing in Future Data Centers. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, pages 51-62, New Delhi, India, August 2010.

[2] R. Bolla and R. Bruschi. Linux Software Router: Data Plane Optimization and Performance Evaluation. In *Journal of Networks*, 2(3):6–17, June 2007.

[3] T. Benson, A. Akella, and D. A. Maltz. Network Traffic Characteristics of Data Centers in the Wild. In *Proceedings of the Conference on Internet Measurement*, pages 267-280, Melbourne, Australia, November 2010.

[4] A. L. Barabasi and R. Albert. Emergence of Scaling in Random Networks. In *Science*, 286(5439):509–512, October 1999.

[5] Cisco. Cisco Nexus 5000 Series Architecture: The Building Blocks of the Unified Fabric. http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9670/white_paper_c11-462176.pdf, 2009.

[6] Cisco. Cisco Catalyst 4948 Switch. http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps6021/product_data_sheet0900aecd8017a72e.pdf, 2010.

[7] Cisco. Cisco Nexus 7000 F-Series Modules. http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9402/at_a_glance_c25-612979.pdf, 2010.

[8] Cisco. Cisco Data Center Infrastructure 2.5 Design Guide. http://www.cisco.com/en/US/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCI_SRND_2_5_book.html, March 2010.

[9] 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 the ACM Symposium on Operating Systems Principles*, pages 15–28, Big Sky, MT, USA, October 2009.

[10] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. In *Numerische Mathematik*, 1:269–271, 1959.

[11] B. Fitzpatrick. Distributed Caching with Memcached. In *Linux Journal*, 124, August 2004.

[12] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: A Scalable and Flexible Data Center Network. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, pages 51-62, Barcelona, Spain, August 2009.

[13] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: A High Performance, Server-Centric Network Architecture for Modular Data Centers. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, pages 63–74, Barcelona, Spain, August 2009.

[14] 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 ACM SIGCOMM Conference on Data Communication*, pages 75–86, Seattle, WA, USA, August 2008.

[15] L. Gyarmati and T. A. Trinh. Scafida: A Scale-Free Network Inspired Data Center Architecture. In *SIGCOMM Compututer Communication Review*, 40(5):5–12, October 2010.

[16] S. D. Gribble. A Design Framework and a Scalable Storage Platform to Simplify Internet Service Construction. In *Ph.D. Thesis*, U.C. Berkeley, September 2000.

[17] S. Han, K. Jang, K. Park, and S. Moon. PacketShader: A GPU-Accelerated Software Router. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, pages 195–206, New Delhi, India, August 2010.

[18] M. D. Humphries and K. Gurney. Network 'Small-World-Ness': A Quantitative Method for Determining Canonical Network Equivalence. In *PLoS ONE*, 3(4):1–10, April 2008.

[19] HP. HP Rack 10000 G2 Series Quick Spec, DA12402. http://h18004.www1.hp.com/products/quickspecs/12402i_na/12402_na.pdf, 2011.

[20] E. Kranakis, H. Singh, and J. Urrutia. Compass Routing on Geometric Networks. In *Proceedings of the Canadian Conference on Computational Geometry*, pages 51-54, Vancouver, British Columbia, Canada, August 1999.

[21] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 163–170, Portland, OR, USA, May 2000.

[22] J. Kleinberg. Small-World Phenomena and the Dynamics of Information. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 431–438, Vancouver, BC, Canada, December 2001.

[23] J. Kleinberg. The Small-World Phenomenon and Decentralized Search. In *SIAM News*, 37(3), April 2004.

[24] J. Kleinberg. Complex Networks and Decentralized Search Algorithms. In *Proceedings of the International Congress of Mathematicians*, pages 1019–1044, Madrid, Spain, August 2006.

[25] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The Nature of Data Center Traffic: Measurements & Analysis. In *Proceedings of the Conference on Internet Measurement*, pages 202-208, Chicago, IL, USA, November 2009.

[26] R. V. Kulkarni, E. Almaas, and D. Stroud. Exact Results and Scaling Properties of Small-World Networks. In *Physical Review Letters*, 61(4):4268–4271, April 2000.

[27] G. Lu, C. Guo, Y. Li, Z. Zhou, T. Yuan, H. Wu, Y.

Xiong, R. Gao, and Y. Zhang. ServerSwitch: A Programmable and High Performance Platform for Data Center Networks. In *Proceedings of the Symposium on Networked System Design and Implementation*, Boston, MA, USA, March 2011.

[28] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In *Proceedings of the Symposium on PRinciples of Distributed Computing*, pages 183–192, Monterey, CA, USA, July 2002.

[29] J. Moy. OSPF Version 2. RFC i2328. April 1998.

[30] S. Milgram. The Small World Problem. In *Psychology Today*, 2:60–67, 1967.

[31] G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed Hashing in a Small World. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, pages 127–140, Seattle, WA, USA, March 2003.

[32] J. Naous, G. Gibb, S. Bolouki, and N. McKeown. NetFPGA: Reusable Router Architecture for Experimental Research. In *Proceedings of the ACM workshop on Programmable Routers for Extensible Services of Tomorrow*, pages 1–7, Seattle, WA, USA, August 2008.

[33] M.E.J. Newman, C. Moore, and D.J. Watts. Mean-Field Solution of the Small-World Network Model. In *Physical Review Letters*, 84(14):3201–3204, April 2000.

[34] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, pages 161–172, San Diego, CA, USA, August 2001.

[35] J. Travers and S. Milgram. An Experimental Study of the Small World Problem. In *Sociometry*, 32(4):425–443, 1969.

[36] B. Wong and E. G. Sirer. Approximate Matching for Peer-to-Peer Overlays with Cubit. In *Computing and Information Science Technical Report, Cornell University*, December 2008.

[37] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A Lightweight Network Location Service Without Virtual Coordinates. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, pages 85–96, Philadelphia, PA, USA, August 2005.

[38] B. Wong, I. Stoyanov, and E. G. Sirer. Octant: A Comprehensive Framework for the Geolocalization of Internet Hosts. In *Proceedings of the Symposium on Networked System Design and Implementation*, pages 313–326, Cambridge, MA, USA, April 2007.

[39] D. J. Watts and S. H. Strogatz. Collective Dynamics of 'Small-World' Networks. In *Nature*, 393:440–442, June 1998.

[40] H. Zhang, A. Goel, and R. Govindan. Using the Small-World Model to Improve Freenet Performance. In *Proceedings of the IEEE International Conference on Computer Communications*, pages 1228–1237, New York, NY, USA, June 2002.