AN3: A Low-Cost, Circuit-Switched Datacenter Network

Eric Chung, Andreas Nowatzyk, Tom Rodeheffer, Chuck Thacker, Fang Yu
Microsoft Research, Silicon Valley

Abstract

In this paper, we present AN3—a low-cost, circuit-switched datacenter network. AN3 replaces expensive IP switches with custom hardware that supports circuit-based switching efficiently and with low cost. AN3 is enabled by a new speculative transmission protocol that (1) enables rapid multiplexing of links to efficiently support many flows in a datacenter-scale computer, and (2) establishes setup and teardown of circuits within tens of microseconds—well below the TCP handshake delay. In simulations, AN3 achieves over 90% link utilization in synthetic and real-world applications. A significant portion of AN3 has been built using low-cost, commodity FPGAs—our cost analysis shows that AN3 reduces the amortized per-server cost of the network by 1.7-2.4X relative to a conventional datacenter network.

1. Introduction

The datacenter network forms the critical infrastructure for enabling robust and scalable communication between hundreds of thousands of datacenter servers and storage nodes. Existing datacenter network architectures are predominantly packet-switched networks built out of expensive, commodity IP-based switches. These switches provide many (largely unused) capabilities and execute millions of lines of code, contributing to software failures and lowered reliability. Including the base costs, repair, and cabling, the datacenter network alone can incur nearly 19% of the entire datacenter's cost of ownership [20].

Our goal in this work is to develop a new datacenter network architecture that can simultaneously improve performance, efficiency, and reliability while drastically reducing networking costs. In a radical departure from conventional practice, this paper argues for the merits of **circuit-switched networks** in the design of future datacenter networks. Unlike packet-switched networks that carry traffic flows in packets, circuit-switched networks establish connections that allocate network resources along a path from the source to the destination. After a connection is established, cells are transferred without interruption until the connection is torn down. A circuit-switched network offers several key benefits in a datacenter:

- **Predictable performance**. In circuit-switched networking, flows are transferred with guaranteed bandwidth, bounded latency, low jitter, and no cells can be dropped due to congestion.
- **Performance isolation**. Flows do not interact with one another. This is particularly important to the operator of a large datacenter. The actions of one customer's application cannot be allowed to interfere with performance received by another customer.

Despite their benefits, traditional circuit-switched networks fell out of favor due to their perceived disadvantages relative to packet-switched networks: (1) *High setup overhead:* In a circuit-switched network, a connection must be established before data is transmitted—this is expensive in a large-scale network such as the Internet, (2) *Low link efficiency:* Since traffic is inherently bursty, a circuit-switched network has low link utilization due to its inability to perform statistical multiplexing, (3) *Complex switch design:* Circuit-switched networks require maintaining per-flow states at each router or switch, and (4) *Less robustness:* If a router or link in a packet-switched network fails, flows need to re-establish new circuits before they can be rerouted. These are plausible reasons why packet-switched networking is used for the Internet today.

In "re-thinking" the network architecture for a datacenter-scale computer, we argue that many of the perceived disadvantages of circuit-switched networking do not arise as they would in the Internet. We observe that: (1) A datacenter is typically set up and maintained by one central organization that has full knowledge of the datacenter topology and link capacities. For a circuit-switched network, this vastly simplifies the complexity and task of rerouting traffic during failures. (2) Unlike the Internet, there are a bounded number of hosts in the datacenter. Therefore, links are short, so the round trip delay is much lower when compared to the Internet.

Thus, the major challenge that remains is how to support many concurrent flows efficiently to fully utilize link bandwidth. We observe that if one can build a network that supports fast circuit setup and teardown, then statistical multiplexing of the links and low latencies can be achieved in a circuit-switched network exactly as it would in a packet-switched network.

The AN3 Network. In working towards the goal of building an efficient, low-cost datacenter network, we present the **AN3 network**—a custom network architecture that scales to 90,000 nodes. AN3 overcomes the limitations of traditional circuit-switched networks using a novel speculative transmission scheme, where cells can be transmitted before a connection is fully established between source and destination. Speculative transmission is critical for enabling rapid setup and teardown of circuits, which (1) increases link utilization efficiency, (2) reduces per-flow states needed per router, and (3) avoids slowing down applications that exhibit short or mixed flow patterns. The AN3 architecture also imposes less complexity and cost than existing datacenter networks by re-

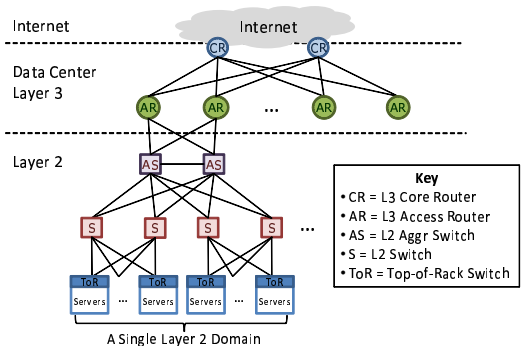

Figure 1: Baseline network (figure adapted from [16]).

placing expensive top-of-rack switches (TOR) with building blocks that can be implemented cheaply in commodity FPGAs. Based on extensive cost analysis, simulations, and prototyping, we demonstrate the following properties of AN3:

- **Excellent Scalability.** We show that the AN3 topology scales up to 90,000 nodes in one geographic location. [1]
- **High performance.** When simulating a variety of synthetic and real-world applications, we show that AN3 achieves very high bandwidth utilizations (90% and above). We show that circuit setups can be achieved in tens of microseconds, well below the TCP handshake delay and two to three orders of magnitude faster than traditional ATM switches [24]. Our experimental results further show that our design supports both long and short flows efficiently (Section 8).
- **Low complexity.** We show that the AN3 building blocks are simple to implement on commodity FPGAs and incur much lower cost in chip area and buffering compared to traditional packet-switched routers (Section 6).
- **Low cost.** Based on a detailed comparison study, we show that AN3 can achieve a 1.7-2.4X reduction in per-server networking costs relative to a standard datacenter network (Section 7).

2. AN3 Network Architecture

Before presenting AN3, we first describe the baseline network architecture commonly deployed in commercial datacenters today [11]. As Figure 1 shows, a traditional datacenter network comprises a tree-based hierarchy of servers and switches, where servers form the leaves, and switches are aggregated in multiple layers to provide full end-to-end connectivity and bisection bandwidth. At the lowest layer, servers (typically a rack's worth, 20-40) are aggregated to top-of-rack (TOR) switches. Above the switches, TORs are connected to aggregation switches, that are in turn connected further by core and Internet switches. The dominant cost of a datacenter network today are in the Network Interface Card (NIC)s, the switches, and the cables.

[1] Beyond that, it is better to have geographic diversity to protect against catastrophic events.

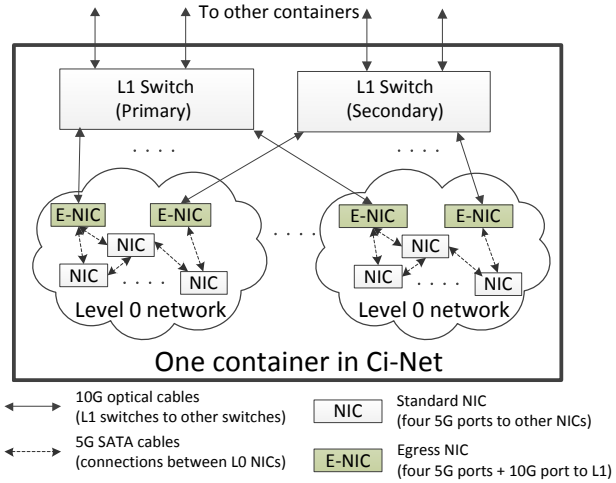

Figure 2: Topology of switches in a single container.

2.1. AN3 Architecture: Single Container

The AN3 network replaces traditional TOR and core switches with two simple building blocks: (1) a specialized, 5-port NIC for building small, local networks between servers in a rack, and (2) a 128-port crossbar switch for aggregating the local servers into a larger global network.

Physically, the AN3 network adopts the popular shipping container model for the datacenter [4]. Figure 2 shows various AN3 components within a single container. In contrast to conventional TORs, the server NICs within the same half-rack (typically 22 servers) are connected to each other to form a Level-0 (L0) network. Each NIC exposes four 5G ports that can be connected via low-cost SATA cables to other NICs within the same L0 network.

Within each container, two redundant 128-port crossbar switches are used to provide full end-to-end connectivity between all L0 networks within the container and between containers throughout the datacenter. In Figure 2, these switches are designated as L1 primary and L1 secondary. Each L1 switch exposes 128 10G ports—half of them used for the L0 networks belonging to one container, and the other half connecting to other L1s belonging in other containers. Within the L0 networks, two NICs (designated as *E-NIC* in Figure 2) have a 5th port that connects to the L1 switches directly. Generally, the AN3 configuration is flexible—more L1 switches or E-NICs could be included in a container to provider higher bisection bandwidth.

2.2. AN3 Architecture: Multi-Container

As shown in Figure 3, in a large-scale AN3 network, there are up to 64 containers connected to each other using L1 switches. For each L1 switch, 63 ports are connected to other containers and 1 port connects to the Network Operations Center. The remaining 64 ports connect up to 64 L0 networks hosted within the same container. In a typical L0 network with 22 nodes per half-rack, one AN3 network can support up to $64 \times 64 \times 22 = 90,112$ servers.

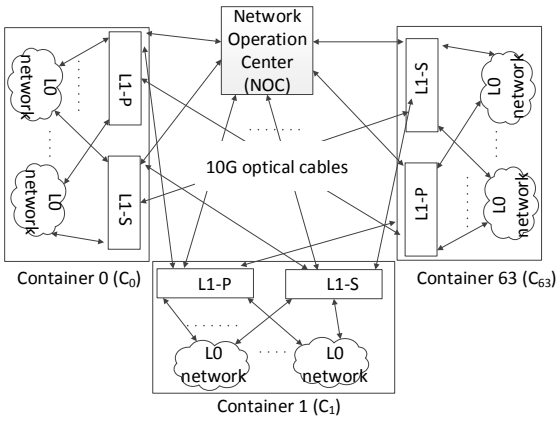

Figure 3: AN3 network with multiple containers. Containers C_i and C_j are connected with two 10G cables from port $64+j$ of C_i to port $64+i$ of C_j through the primary and secondary L1s. Adding a new container does not require rewiring of existing containers.

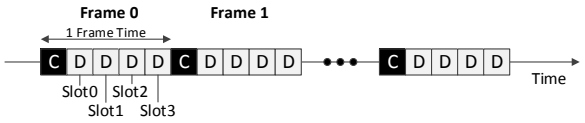

Figure 4: Example of framing on the link with 1 control cell and 4 data cells per frame.

The Network Operations Center shown in Figure 3 provides a AN3-to-TCP/IP bridging function and connections to external routers. The AN3 network employs Valiant Load Balancing (VLB) to provide more bandwidth than a single pair of links between a pair of containers can provide. When the bandwidth of the two links between a pair of containers is saturated, additional circuits can be routed through another L1 switch rather than being routed directly to the destination. When there are less than 64 containers in the data center, unused ports can also be connected to each other to provide higher bandwidth between containers.

3. Circuit-switched Network

The most significant departure from standard networking practice in AN3 is the use of circuit-based switching rather than packet-based switching.

In AN3, there are two types of cells flowing through the network: the control and data cells. Control cells are used to set up and tear down circuits and data cells are used to carry the actual data. More specifically, bits over the link are transmitted as a stream of repeating *frames*, each consisting of 128 8-byte control cells and 128 64-byte data cells. A data cell within the frame is identified by a *slot number*, and a frame has 128 slots. Unused slots are marked with null data cells. On a 10G link with 128 slots and 66/64B encoding, each port sends out exactly one frame of data during each frame time (7.32us). Figure 4 illustrates a simplified example of framing on a link.

Any given data cell belongs to some part of an existing *flow*.

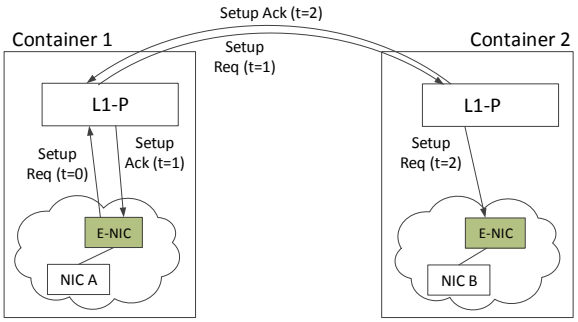

Figure 5: Example flow setup and tear down topology. After L1-P in Container 1 processes the setup request at time 0, it forwards the request to the next L1 switch. Meanwhile, it sends an ACK to the previous switch (E-NIC), telling the source NIC that it is ready to accept data.

The client NIC at the source is responsible for controlling injection of data cells into the network at one cell per frame time for each flow. Flows are set up unidirectionally since traffic patterns may not be symmetric. Data cells flow through the switches in the forward direction, and ACKs (one ACK for a number of data cells) travel along the reverse path. For symmetric flows, two circuits are set up, one from each end.

When a path through the network is set up, each switch on the path assigns a local slot in its frame to this flow. All subsequent data cells belonging to this flow occupy this slot at this switch and an outgoing link towards the next switch. The slot numbers are carried in the data cell headers and are used to determine the input buffer for the data payload at the next switch. The slot number is modified at the output unit in each switch along the path in a manner similar to the modification of the virtual circuit identifier in an ATM network. As we will discuss in Section 4, the use of framing and slots in the L1 network vastly simplifies the design complexity and dynamic scheduling of a 128×128 crossbar.

3.1. Circuit Setup and Teardown

Since flows can be short in datacenter workloads, AN3 creates circuits dynamically for each flow, rather than trying to predict the aggregate bandwidth for multiple flows and setting up long-lasting circuits. The setup request reserves a default (low) bandwidth for each flow. When an application requires higher bandwidth, multiple flows are requested independently to achieve a higher aggregated bandwidth allocation.

To mitigate the path setup overhead for short flows, data cells can be transmitted *speculatively* by the source NIC as soon as the path setup process begins. Figure 5 illustrates the setup process using an example topology, where the source NIC A tries to establish a flow to the destination NIC B.

In similar vein to the RSVP protocol [40], the source NIC A initiates a setup request, which travels through all intermediate switches to reserve bandwidth, and finally reaches the receiver (NIC B). However, unlike RSVP, the ACKs in AN3 are generated differently. In RSVP, ACKs are initiated by the destination NIC, so the source must wait a full round trip delay

before it can receive an ACK and start sending data. In contrast, the L1 switches in AN3 respond with ACKs immediately after processing setup requests, when the flow has yet to be established all the way to the destination. The ACK travels to the previous L1 switch or NIC, informing it that the next switch is ready to accept data.

In the example shown in Figure 5, after the L1 switch of Container 1 processes the setup request, it simultaneously forwards the setup request to the next hop switch (L1 of Container 2) and sends an ACK to the previous hop, which relays it to the source NIC. When the source NIC receives a setup ACK, it begins sending data (speculatively) on the connection. In this protocol, the source NIC does not wait for a full round trip delay before pumping data into the network.

If a connection is rejected by some switches along the route (or by the destination NIC itself), a teardown is sent along the reverse path to the source NIC, releasing resources allocated in the switches it passes. Similarly, when a switch or a link fails, the neighboring switch detects the error and tears down all existing connections. When the source NIC receives the teardown message, it attempts to use the alternate L1 switch to create a new connection or to use VLB routing.

Data cells that are injected into a connection where a setup request subsequently failed are discarded by the network. They are re-sent by the source NIC after another circuit is successfully set up. To ensure correctness, the setup and teardown protocols were specified in TLA+ [23] and verified using the TLC model checker [38]. Due to page limitations, we refer the reader to a technical report for these details [3].

3.2. Sending Data

Once a setup ACK is received, the sender NIC begins sending one data cell per frame time for this flow. Its first cell has a distinguished type, so the destination knows whether it is missing initial cells. If so, it sends a Restart message back to the source and the source will restart the flow. In practice, this case is extremely rare (only when the control channel is heavily loaded). This is because data cells experience at least one frame of delay at intermediate switches, while the control cells do not, hence the probability of data cells catching up to control cells is low.

A source NIC sends a Chunk Mark cell on a connection every 128 data cells (a 4 KB "chunk"). The Chunk Mark includes a checksum over the 4KB of data calculated by the source NIC, as well as a sequence number for the chunk. The Marks flow on the data connection and arrive after the last data cell corresponding to the chunk. The resulting Data ACKs, which contain the chunk sequence number, are sent in control slots from the receiver back to the sender. A chunk forms the unit of error checking and retransmission for a flow. If a destination NIC receives a checksum error upon receiving a Chunk Mark, it does not respond with a Data ACK, and the source will retransmit the chunk after a (short) timeout.

When the source wishes to end a flow, it sends a Last Chunk Mark. When the destination NIC sees a Last Chunk and the checksum is correct, it responds with a Teardown message. The destination NICs also provide a mechanism to suspend a flow temporarily in order to multiplex the limited number of simultaneously active flows among hundreds to thousands of logical connections. These suspended flows can be restarted at the point of suspension. The suspension mechanism allows us to provide a device driver that matches the characteristics of the AN3 hardware to the familiar sockets abstraction provided by the operating system to clients. Clients can open many simultaneous logical connections, which are transparently multiplexed between the limited number of simultaneous flows supported by the network. Similarly, long connections can be torn down after a fixed period and re-setup to avoid starving other connections. If there are no other connections that need to be served, long connections can continue to utilize available resources.

3.3. Compatibility with Existing Networking Protocols

The AN3 network provides fast connection setup and teardown as a basic communication primitive. Applications can use these primitives to send data directly or tunnel higher-level legacy protocols such as TCP. Because cells flow at a fixed rate with no jitter or packet losses due to congestion, AN3 does not rely on TCP to conduct flow control and avoids many TCP-related congestion problems such as TCP incast [27, 10]. Other functions provided by TCP, such as data sequencing, error retransmission and rate control, are provided by the client NICs. NICs, together with the OS driver, expose a simple circuit "socket" abstraction that can support existing datacenter workloads. External connections can also be easily tunneled to exit routers using circuits, just as many IP links are now built upon ATM links.

Note that broadcast and multicast functions are not directly supported by AN3. End-host-based solutions could be used to facilitate such functions [39, 22].

4. L1 Switch

The L1 switch architecture forms the heart of the AN3 network and is a critical building block for creating large-scale, low-diameter networks. In a traditional packet-based design, dynamic scheduling of packets in a large crossbar is prohibitively expensive and complex due to the matching problem. In AN3, the use of slots and framing dramatically simplifies this process, enabling the practical implementation of a large, 128-port crossbar with minimal logic.

Figure 6 shows the overall architecture of the L1 switch. In total, there are 16 line cards, each containing 8 bidirectional 10G ports for an aggregate throughput of 2.56Tbits of bandwidth. Each 10G port contains an input unit for receiving cells and an output unit for sending cells. Together, we call them I/O units. When framed data arrives on an I/O unit, data cells are separated from control cells, as they go through different paths in the L1 switch. Control cells are sent to the ringmaster as shown in Figure 6, whereas data cells are sent to the frame

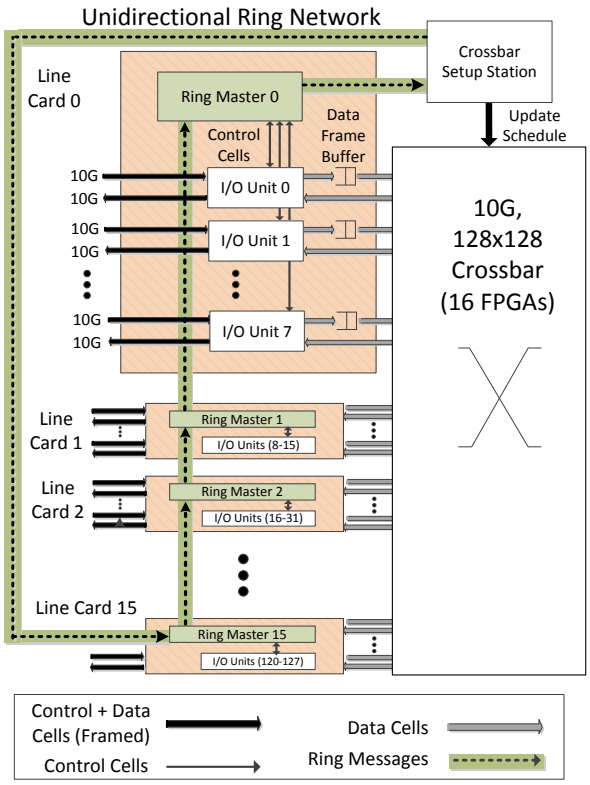

Figure 6: The L1 switch. The switch consists of 128 ports and 16 ringmasters. Control cells travel through the ring. Data cells go through the frame buffer and the crossbar.

buffer preceding the actual crossbar. Next, we describe the crossbar and the ring masters in detail.

4.1. L1 Crossbar

AN3 pre-configures the switching instructions within the crossbar chips during circuit setups. This avoids the dynamic matching computation and allows the AN3 switch to scale to large port numbers. The AN3 128×128 crossbar (right, Figure 6) is bit-sliced over 16 crossbar cards, each on its own PCB. Each crossbar chip contains $128 \times 128 : 1$ multiplexers, one for each output port. The switching instructions within each crossbar chip determines the input line selected by each multiplexer during each slot in a frame. Each crossbar chip cycles through the 128 slots in a frame, matching inputs to outputs according to the instructions. After finishing all 128 slots in a frame time, the crossbar repeats the process from the beginning.

Figure 7 uses a 2X2 crossbar as an example to illustrate the switching process. At time $t = 0$, multiplexers of both output ports process data cells in slot 0 in parallel. In this example, the output port A selects the port B according to its switching instruction and transfers the data D_2 from port B, while the output port B transfers data D_0 from port A. At time $t = 1$, these multiplexers load the next round of switching instructions and process the data cells in slot number 1. At time $t = 2$, this process repeats and the crossbar returns to processing slot 0 of the next frame.

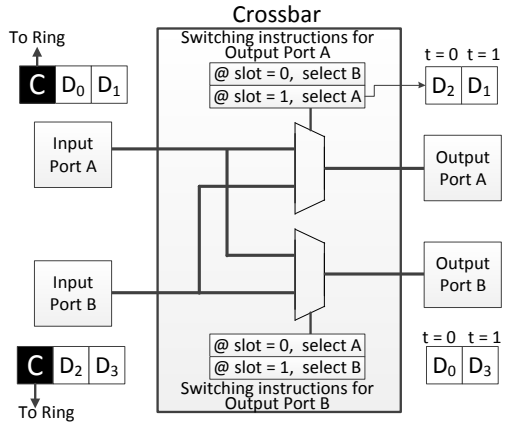

Figure 7: Example 2x2 L1 Crossbar.

The switching instructions are set by the Crossbar Setup Station (top-right, Figure 6). The Crossbar Setup Station sits along the path of the ringmasters and watches for ACKs that are transmitted along the ring during the setup process, which is described next.

4.2. L1 Ring

Since the crossbar can only ship data cells from one port to another in a common slot—for any new setup request, the L1 switch must identify a common free slot for both the input and output ports. To perform this quickly in the L1 switch, both input and output ports maintain 128b bitmasks that track the slot utilization of their respective links. We refer to these bitmasks as input and output schedules, respectively. An input schedule records free slots for flows that arrive at a given port, while an output schedule records free slots for flows departing at a given port.

In a straightforward setup, one can use a centralized processing unit to maintain all schedules and process all requests. However, this approach easily creates bottlenecks. Thus, we adopt a distributed approach: 16 distributed ringmasters (one per linecard) are responsible for processing control cells and orchestrating the setup and teardown of flow resources. Each ringmaster services the 8 I/O units within each linecard. Ringmasters are connected to each other in a ring that passes between the line cards on a large back-plane.

Figure 8 gives a detailed step-by-step example of how setup requests are processed. When a setup request control cell reaches the ringmaster from one of its 8 I/O units, a common unused slot must be identified for both the source port and destination port. This typically requires two ringmasters of the input and output ports to coordinate this request. When the input ringmaster services a setup request ((1) in Figure 8), it first locks the input schedule for the input port. It then forwards the setup request together with the input schedule along the ring to the output port ringmaster (2,3). When the ringmaster of the output port receives the message, it ANDs the received input schedule with its own output schedule to identify a common free slot (4). If a common slot is found, the ringmaster of the output port deasserts the corresponding bit

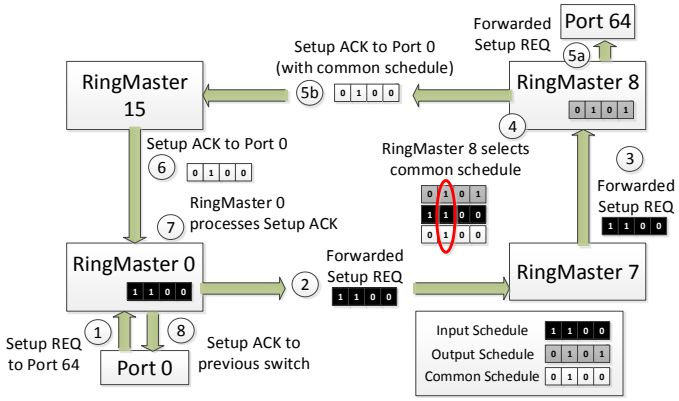

Figure 8: Setup process through the ring.

in the output schedule and sends an ACK to the input port with the common slot (5b). Simultaneously with the ACK (5a), the ringmaster of the output port forwards the setup request to the next switch along the path. When the next switch responds with an ACK later, a small table in the output ringmaster associates its own common slot with the local slot of the next switch (not shown).

At the same time within the same L1 switch, when the input ringmaster receives the ACK (6), it records the output slot and associates it with the original setup request (7); this is followed by an ACK to the previous switch (8). Within the original input ringmaster, a small table records the mapping between the local slot of the previous switch to the newly allocated common slot (not shown).

With distributed ring processing, our prototype AN3 switch can handle 156 million setup and teardown requests per second, several orders of magnitudes higher than ATM switches that can only set up hundreds to thousands calls per second [24].

5. L0 NIC

L0 NICs are in charge of maintaining the flow status and sending and receiving cells. They must be aware of framing. During any one-frame interval, a NIC may not send more than one cell to an L1 switch for a given L1 slot number. At the egress NIC, data and control cells are combined in a frame to send to the connected L1 port.

In AN3, the L0 networks are arranged differently than the L1 because they are much smaller. The NICs route cells through the L0 network, rather than using the customary TOR switch. The L0 NICs use routing tables based on the known topology of the network. These tables are only updated by a lightweight NIC CPU if a node or link in the local L0 network fails. The NICs use cut-through forwarding to minimize latency. The per-hop latency through an unloaded NIC is 192 ns, so traffic traverses an L0 network in much less than a frame time. Hence, the total latency through the network is dominated by the frame delay at the L1 switches.

The topology of the L0 network and the number of egress NICs per L0 network are flexible. Different data centers, or even different L0 networks can have different configurations. Common topologies such as a 2-D torus or mesh can be used. In our experiments, we use a 22-node fat-tree with cross-links. Each NIC has four 5G link to four other NICs. Two egress NICs have extra 10Gb/s links to the L1 switches. In our L0 topology, all nodes reach an egress NIC within two hops and can reach any other nodes in the L0 network with an average of 2.04 hops.

Component	LUT Area	FF Area	Memories
L0 NIC Logic	9368, 4%	9276, 2%	33%
L1 Ringmaster	2400, 1%	670, <1%	2, <1%
L1 I/O Unit	1136, <1%	314, <1%,	1, <1%
L1 Line-card	11544, 5%	2638, <1%,	6, 1%

Table 1: L0 and L1 Components Mapped to Xilinx Kintex-7 LX325T-3 FPGA.

6. AN3 FPGA Protoyping

In this section, we report on the prototyping efforts of AN3 on commodity FPGAs. Samples of the AN3 L0 NIC were recently received from our manufacturer and tested successfully on commodity PCI express Gen-3 motherboards. The 5-port L0 NIC comprises a single Xilinx Kintex-7 LX325T-3 FPGA along with 4 SATA ports (to operate as 5G links) and a single 10G optical transceiver. The L0 design was implemented in synthesizable Verilog and has been validated in actual hardware. Table 3 shows area statistics about the design, with under 5% of the total FPGA being utilized. The SATA-3 links are able to sustain the full 5Gb/s rate with extremely low error rates (a single bit error every few days).

In addition to the L0 NICs, the L1 crossbar has been fully implemented in synthesizable Verilog. Table 3 shows the synthesized costs for the various sub-components, including the ringmaster, the I/O unit, a single line-card, and a single crossbar chip. For the relatively modest FPGA, the line-card (which includes eight I/O units and a single ringmaster) fits comfortably within 5% area of the FPGA. It is noteworthy that the L1's modest area consumption was achieved without any logic optimization due to the simplicity of the L1's circuit-switched architecture. Furthermore, the lack of extensive buffering contributes significantly to a lower area cost.

The L0 NIC is measured to consume about 12W of power, comparable to standard off-the-shelf 10G NICs. We do not have measured power numbers for the L1 switch but we conservatively estimate that with 33 LX325T-3 FPGAs operating at worst-case consumption, the total L1 power should not exceed 400W, comparable to TOR switches [34].

The L1 switch components have been validated in RTL simulation with a combination of test cases and assertions. In addition to the test cases, the L1 setup and teardown protocols were described using a formal specification written in TLA+ [23] and verified using the TLC model checker [38].

		Baseline			AN3	
Servers	Oversub	NICs	TORs	Agg Sw	L0 NICs	L1 Sw
20K	2.5	20K	500	143	20K	125
20K	5.0	20K	429	71	20K	63
40K	2.5	40K	1000	286	40K	250
40K	5.0	40K	857	143	40K	125
60K	2.5	60K	1500	429	60K	375
60K	5.0	60K	1286	214	60K	188

Table 2: Datacenter Network Configurations. The number of TOR, Aggregated, and L1 switches are calculated based on the number of servers and the oversubscription rate.

	Unit Price ($)	NRE ($)	Discounts (0.5, 0.75)
10G NIC	400	-	200, 100
56-port 10G Agg Switch	34,500	-	17250, 8625
L1 Switch (FPGA)	6,600	100,000	-
L0 NIC (FPGA)	90	100,000	-
L0 NIC (ASIC)	10	1,000,000	-
L0 SATA Cable	0.10	-	-
10G Copper+Cable (20m)	25	-	-
10G Optics+Cable (20m)	110	-	-

Table 3: Datacenter Component Costs (Baseline and AN3).

7. Cost of AN3 vs. Traditional Networks

A major goal of AN3 is to significantly reduce the cost and overhead of building a datacenter network. As shown by Hamilton et al. [20], the network can incur up to 19% overhead relative to server costs. Our analysis will show that AN3 can reduce this component by 1.7-2.4X, even when factoring in volume discounts. To estimate costs, we compare AN3 to a representative baseline datacenter network discussed in Section 2. To ensure a fair comparison between the two radically different architectures, our cost analysis follows the methodology by Popa et al. [29], where both network architectures are configured to have approximately equivalent bisection bandwidth and delay.

For a conservative estimation, we select network parameters that minimize the overall cost of the baseline network. We use a standard, 56-port 10G access layer switch [1] recommended by Cisco [11] for **both** TOR and aggregation switches[2]. We conservatively ignore the cost of higher-level core switches and only count server NICs, TORs, aggregation switches, and the cables used between them. An important parameter that has a first-order effect on switch costs is oversubscription at the TORs. In our study, we select oversubscription values based on recommendations from Cisco [11] and for various datacenter workloads [2].

To mimic the baseline topology in AN3, we configure each L0 network to have the same number of servers aggregated to each TOR. To provide the same level of bandwidth between TORs, we scale the number of egress NICs within the L0 to match the oversubscription on the baseline TOR switches. The egress NICs are connected to 64 inward facing ports of the 128-port L1 switches, and the remaining 64 outward facing ports are fully connected to other L1 switches. In our cost model, we consider two variants of AN3, one using higher-cost FPGAs for the L0 NICs and one with ASICs.

Table 3 shows estimated per-component costs based on publicly available listing prices and internal estimates. Given that the prices can vary dramatically based on volume discounts, market forces, etc., our analysis assumes discount factors of 50% and 75% for the baseline switches and NICs. The component costs of AN3 are based on vendor-provided volume pricings of FPGAs and ASICs, board design, and manufacturing. For the baseline, low-cost copper cables are used for intra-rack connections, while more expensive optical cables and transceivers are assumed for inter-rack communication. In AN3, we assume SATA cables for intra-rack communication within the L0 network and optical cables and transceivers for L1 port connections. In AN3, we also include the NRE design costs of implementing custom hardware.

Discussion. Figure 9 shows the per-port costs of AN3 and the baseline networks for various server configurations (20K, 40K, 60K), levels of oversubscription (2.5, 5), and discount factors (50%, 75%). With an oversubscription of 2.5, the AN3 per-port costs are significantly better than the 75% discounted baseline by a factor of 2.4X at best (60K servers, ASIC L0s) and 1.7X at the minimum (20K servers, ASIC L0s). As expected, the per-port cost of AN3 with ASIC L0s improves with increased volume. However, even in a low-volume scenario using more expensive FPGA L0 NICs, the AN3 still achieves a 1.8X improvement in cost relative to the baseline network with a 75% discount factor. We observe similar trends at a higher level of oversubscription (5).

Figure 10 shows the cost breakdown between the baseline network (with a 75% discount) and AN3. The AN3 incurs a much lower cost in switches and NICs due to the very simple, circuit-switched hardware deployed on low-cost FPGAs and/or ASICs. Note, our cost analysis is highly conservative and favorable towards the baseline network because we do not include the cost of expensive core switches (we assume all aggregations switches are fully populated using the lowest-cost switch). Overall, our cost analysis shows that AN3 is practical to implement using commodity technology available today and can offer sigificant cost savings in the datacenter under conservative assumptions (1.7-2.4X).

8. Evaluation

We rely on simulation to measure the performance of large AN3 configurations under a variety of workloads. Our simulator implements all details of the L0 and L1 switches including ports, buffers, ringmasters and crossbars, as well as the full setup, teardown, and data transmission. All parameters such as processing times and delays were calibrated from our FPGA-based implementation.

[2]Note: we make the conservative assumption that the lowest-cost switch can be used at any level of the hierarchy. Although we could use fewer, higher-radix switches for aggregation, the average per-port cost increases substantially.

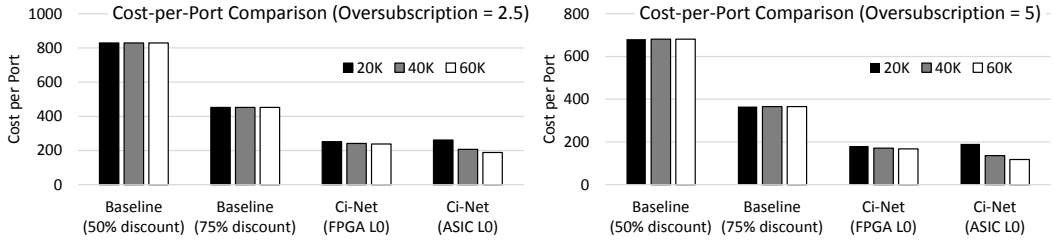

Figure 9: Cost-per-port (AN3 vs. Baseline).

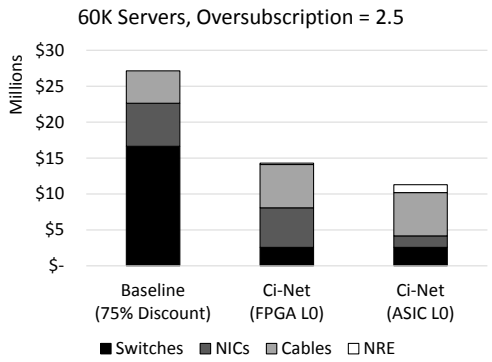

Figure 10: Cost Breakdown (AN3 vs. Baseline).

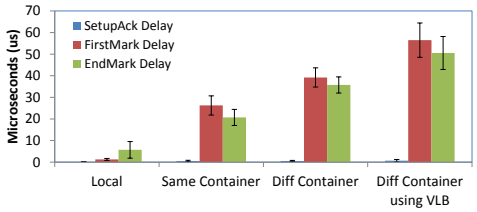

Figure 11: Setup and teardown delay.

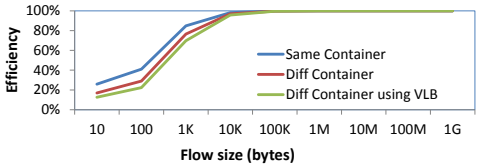

Figure 12: Setup and teardown overhead.

8.1. Experimental Setup

We evaluate three types of traces: (1) randomly generated flows to test the circuit setup and teardown latencies, (2) a well-known sorting benchmark (used in [16, 19, 15, 8] etc.) to test whether the AN3 network can maintain high throughput for all-to-all traffic, and (3) traces from a commercial data center network, also used in [3] to study the impact of uneven distribution of node loads.

In our experiments, each NIC sustains at most two outstanding remote setup requests (one towards each egress link), so that one aggressive node will not starve other neighboring nodes. Subsequent requests are buffered in a request queue at the NIC. An ACK is received from the first L1 switch indicating whether the request is granted or not. If the request is granted, the NIC starts sending data cells speculatively. If not, the request is inserted at the tail of the request queue and is retried again later. Upon receiving the ACK, the NIC immediately sends out another setup request.

8.2. Performance on Random Flows

To test the speed at which AN3 can setup and teardown flows, we use a topology containing three containers of nodes (4224 nodes in total) to test local, same-container, cross-container, and VLB traffic. Each node in the system randomly issues setup requests to other nodes in the system. The duration of a flow is randomly selected, with the shortest flow being just 10 bytes. We generate enough flows to saturate the network.

Figure 11 presents the setup and teardown overhead, plotting the three types of delay a connection experiences. The first is the Setup ACK delay, which records the time from when a source NIC starts sending out a setup request to the time it receives a Setup ACK, either from the destination NIC (local traffic) or from the first L1 switch (remote, cross-container, and VLB traffic). Figure 11 shows that the Setup Ack delay is very short (below 1 us) for all four types of traffic.

The second type is FirstMark delay, which records the time from when a source NIC sends out the setup request to the time the first data cell arrives at the destination NIC. Figure 11 shows that the FirstMark delay increases as the path gets longer because the data cells experience frame delays at each input port of the L1 switch. However, unlike packet networks, where the buffer delay can become very long if there is a congested output link, the frame delay in AN3 is bounded. Every cell experiences one frame delay at the input frame buffer, plus up to one frame delay imposed by the scheduling of crossbar (the major source of the error bar in Figure 11). So, a cell can experience a maximum of two frames of delays (7.3 us*2 = 14.6 us) at each L1 switch and the output port of an egress L0. As we limit the maximum number of hops a data cell travels in the AN3 network to be three L1 switches (VLB case), the frame delay a data cell can experience is bounded. Figure 11 shows that even with VLB, the FirstMark delay is only about 60 us, well below a typical Round Trip Time (RTT) in a packet-switched data center network [37].

The last delay we examine is the EndMark delay, which records the time when a source NIC sends out the last data cell to the time it receives a teardown acknowledgement from the receiver. Figure 11 shows that it follows the same trend as the FirstMark delay, but its value is slightly lower. This is because when the teardown control cell travels back to the source, it is given a higher priority to access the ring than the setup request cells. Therefore, it experiences less delay.

Next, we study the impact of setup and teardown delay on the overall flow transmission time. Since flows have variable sizes, the setup and teardown delay is more critical to small flows. Figure 12 plots the efficiency of different flow sizes, where efficiency is calculated as the total time the NIC spends on raw data transmission only over the entire time the network spends on the flow (including the setup and teardown overhead). As we can see from the figure, the efficiency of very small flows with only one cell can be quite low. But when the flow size increases to 1KB (16 frames), the efficiency grows to 69%-84%. For flow size greater than 10KB (157 frames), the efficiency is 96-98%. Previous studies of data center workloads report that the vast majority of background traffic is greater then 50KB and query traffic is 1.6KB to 2KB [5]; thus, AN3 is able to handle data center workloads with very little overhead.

8.3. Performance on a Sorting Benchmark

Sorting is a popular benchmark for testing data center performance. In MapReduce-style implementation with N machines, there are two phases [30]. In Phase 1, a local processing step is performed where N mappers process their inputs locally. Each mapper divides the input into N ranges and produces N output files. At the end of phase 1, the network shuffles $N*N$ output files to N reducers. In phase 2, each reducer is in charge of one specific range and produces the sorted output. In some implementations, each reducer duplicates the results in several other nodes for fault-tolerance.

We test the AN3 network using phase 1 (all-to-all) and phase 2 (all-to-many) traffic. In phase 1, we test two settings with each node sending to all other nodes in a random order and a fixed order. The former is commonly used in other tests [15], where the load of both the source and the destination nodes is roughly balanced throughout the test, while the latter causes more contention. We use two topologies: a small network with 5 L0 networks (110 nodes) and a larger network with a full container (1408 nodes). In both, each node sends 10KB files to every other node in the system.

All-to-all traffic with random ordering. For the all-to-all traffic, the links from the L0 egress NICs to the L1 switches (10Gb/s) are the bottleneck. Figure 13 shows the throughput on these links in the 110-node test. Almost all of the 128 available slots are used during the test so the throughput is close to 10Gb/s. There are 7 small dips during the test as flows complete and new flows start. We call the transmission between dips a *round*. Each node sends flows to other 109 nodes, with 21 of them being local and 88 of them being remote connections. As one L0 network has two egress links with 256 slots in total, each node in the L0 network gets an average of 11.63 slots in each round. To send all 88 remote connections, the first 7 rounds operate nearly at the full capacity, while there are not enough flows to saturate the full bandwidth in the last round, as shown in the last dip of the solid line in Figure 13. If NICs are allowed to increase their slot sizes for existing

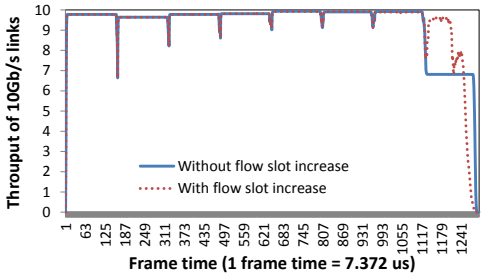

Figure 13: Throughput of a 110 nodes all-to-all traffic.

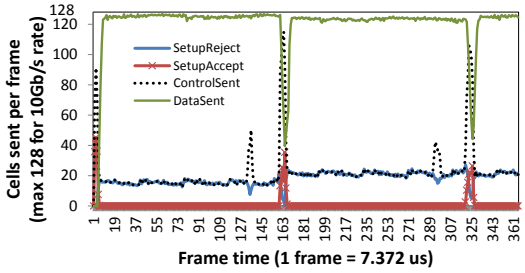

Figure 14: Different types of cells sent during the first two rounds (110 nodes all-to-all traffic).

connections by setting up new slots when there are no more requests in its queue, we can improve the utilization of the last round as shown in the dotted line of Figure 13.

Figure 14 zooms in the first two rounds and plots the number of data and control cells transmitted on the 10Gb/s link. We can see that, in the first few frames, there are many control cells for setting up flows and many setup requests being accepted. Shortly thereafter, data cells begin to flow on the link. For every 128 data cells, source NICs send chunk marks, which trigger small peaks of ACK control cells around frame time 136 and 294. Even when the network is almost fully utilized, there are still setup requests flowing through the network, trying to use the final unused slots. Most of them are rejected at the L1 switch either because the slots are full or no common slots can be found. But when there are flows releasing slots, these requests can quickly acquire the free slots. Therefore, the gap between rounds are tiny with less than 8 frames (< 6 us)—two to three orders faster than the speed of commercial optical switches reported in [15].

We also test the all-to-all performance on a larger network with 1408 nodes sending 10KB files to each other. The all-to-all traffic on 1408 nodes finishes in 142.0 milliseconds (ms). Compared to the NIC's pure data sending time (123.8 ms, which does not take into account of link delay and setup/teardown overhead), this is 87% of overall efficiency even with all the contentions from different nodes. Also note that these flows are fairly small with only 10KB each. For larger sorting benchmarks that have larger files to shuffle, the efficiency would be higher. We also performed a test on an even larger network with three containers of nodes with similar results. Due to space limitation, we omit the results here.

All-to-all traffic with fixed identical ordering. The results presented above are based on the setting in which each NIC

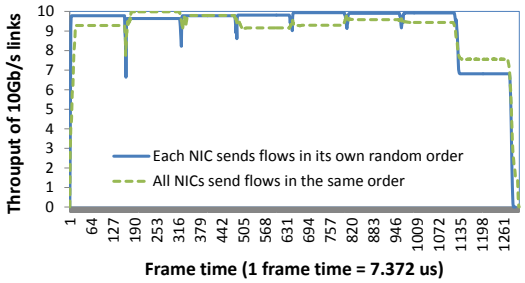

Figure 15: All-to-all traffic with random v.s. fixed flow ordering.

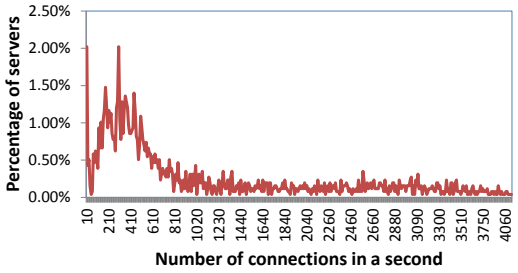

Figure 16: Distribution of connections per second per server.

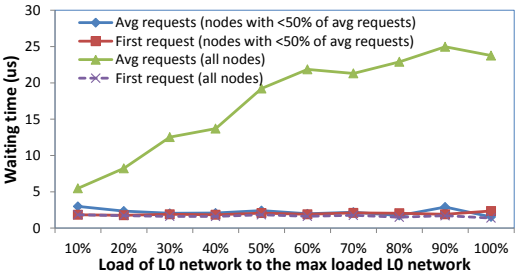

Figure 17: The waiting time for nodes with few requests.

sends to other NICs in its own random order, so the incoming requests to each NIC are evenly distributed throughout the test. This setting is commonly required for many existing networks to achieve high throughput on the all-to-all benchmark. In this subsection, we present the results of another setting, where NICs communicate to others in a fixed order, i.e., from 1 to N, to maximize the contention. In this setting, all nodes send data to the same destination node simultaneously, which is known to produce the infamous TCP Incast problem in packet-switched networks.

Figure 15 shows the comparison of the two settings. The result of the fixed order sending is only slightly worse than the random-order test in the AN3 network. This shows that the limited number of slots in AN3 network allows the network to reject flows, which essentially reorders the flows to achieve a high overall throughput automatically. Even if the application writer does not optimize the flow sending order to take advantage of the network's characteristics, the AN3 network can still accommodate these requests efficiently.

All-to-many traffic. In some systems, the final phase of the sorting benchmark requires each merge node to duplicate results to other nodes for fault-tolerance purposes. A duplication factor of two or three is often used in practice. We test the AN3 network with each node sending 50KB of traffic to two other nodes randomly selected from the entire 110-node pool. Under this workload, since each node in the L0 network only has 2 connections through two egress NICs, the outgoing link from one egress NIC to the L1 switch only carries roughly 22 connections, which could not fully saturate the 10G link. The dynamic flow slot increase scheme, which allows the NIC to request more slots for existing long lasting flows if it does not have any waiting requests, resolves this issue. It is able to utilize the unused bandwidth and finishes the test 3.55 times faster.

The sorting benchmark shows that the AN3 network performs gracefully on all-to-all traffic and all-to-many traffic, where nodes have symmetric workloads. Next, we study the performance of AN3 on workloads with uneven distribution of loads across nodes.

8.4. Performance on Real Data Center Workloads

We use flow-level traces from a real datacenter with more than 2000 nodes. Nodes in the trace have different loads as shown in Figure 16. A majority of the nodes emit several hundred flows per second, while a small fraction of nodes send out several thousands of flows per second. Because the topology and the bandwidth of AN3 are significantly different from the original datacenter network, we do not directly compare the throughput and efficiency of two networks. Instead, we focus on the performance of different nodes to demonstrate that the AN3 network can work efficiently for nodes with thousands of requests per second without starving nodes with very few connection requests.

Response time of nodes with few requests. We analyze the waiting time, i.e., the time from when the application issues a flow setup request to the NIC to the time the NIC gets an ACK from the network. In other words, the waiting time includes both the time spent in the NIC's queue and the time for the network to generate a response. We compare the waiting time of "light" nodes that send out less than 50% of the requests to the waiting time of "average" nodes.

Figure 17 shows the load of the L0 network vs. delay. As various L0 networks experience different loads, we categorize the L0 network according to the total number of requests compared to the maximally loaded one. Unsurprisingly, nodes in a lightly loaded L0 network experience shorter waiting time compared to nodes in a heavily loaded L0 network. Interestingly, this only applies to the average request case. The first request of all nodes stays very low (less than 3 us) regardless of the L0 network load. This shows that for all types of nodes, whenever they want to send one request, they can always send it out quickly and get a response from the network. It is only the case where a node wants to send many requests and the neighbors are busy sending out many requests as well, that subsequent requests need to be buffered at the NIC. We believe that the admission control strategy imposed at the NIC (one outstanding request per egress direction) and the egress NIC (limited slots) is better than letting all connections go through

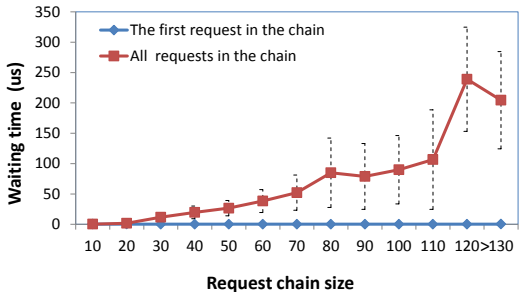

Figure 18: Waiting time for nodes with many concurrent requests.

simultaneously into the L1 network, which could cause serious congestion.

Performance of nodes with many requests. Figure 17 shows that for nodes that send out many connections in a loaded L0 network, they can experience on more than 20 us of waiting time. Is this an acceptable delay, especially for nodes that wish to set up thousands of concurrent connections?

To answer this question, we divide the concurrent connections into two cases: the external and internal data center connections. For external connections, even if a server has thousands of concurrent connections, they go through the same path in the local data center, i.e., from the node to the router (co-located with NOC). Therefore, all these connections can be tunneled through just one circuit.

For internal connections, we study the prevalence of such problem in our trace. In particular, we look at the size of concurrent requests a node generates within a short window (1us), which we call the request chain size. In our trace, vast majority of the chains are short: 90% of them have less than 3 concurrent requests and 99% of the chains have less than 40 concurrent requests.

Figure 17 plots the waiting time of nodes with different request chain sizes. Waiting time goes up when the request chain size increases, as we limit each NIC to have at most two outstanding remote setup requests in our setting. However, given 99% of request chains are smaller than 40 requests, the delays those nodes experienced are below 20 us, which is well below the typical round trip delay we saved using our speculative transmission protocol.

For the few nodes that have long request chains in busy L0 networks, they may experience hundreds of microseconds of delay, which is still smaller than the hundreds of microseconds RTT delay in the current packet-switched network. Therefore, applications would hardly notice the effect.

It is possible to improve the waiting time for long chains: We could allow NICs to have more outstanding requests. NICs can also incorporate better scheduling heuristics to cut down the waiting time. For example, they could tear down long lasting flows temporally and serve new requests. Further, one could design better heuristics or adopt CPU scheduling algorithms from operating systems to achieve better results. This is beyond the scope of this paper.

To summarize, our experimental results have showed that the AN3 network can set up and tear down flows within tens of microseconds, which is well below the TCP handshake delay. In addition, we are able to support large all-to-all traffic gracefully and also real-world data center workloads efficiently.

9. Related Work

The idea of circuit switching is not new and was widely used in traditional ATM [13, 21] and telephone networks [14]. AN3 improves the performance of circuit-switching with speculative transmission, whereas in ATM and telephone networks, circuits are long lasting and must be set up before cells can flow. AN3 also employs a new scheduling method in the crossbar, enabling scalability to much larger port numbers (e.g., 128) than possible in traditional ATM switches (e.g., 16 [33]). Compared to ATM switches that can only set up up to thousands of calls per second [24] (a major reason why ATM was unpopular), AN3 handles 156 million setup and teardown requests per second per L1 switch (orders of magnitude higher).

AN3 uses a simple and efficient framing scheme, whereas traditional circuit-switching networks such as SONET employ complex two dimensional framing formats [35]. AN3 can achieve this simplicity because the datacenter network is simple with a known topology. Thus the traffic monitoring and failure recovery schemes can be separated from framing. In AN3, frames are used to carry data and control cells directly, whereas SONET's framing provides other functionalities such as monitoring traffic quality, detecting failures, and supporting different higher protocols, etc [35].

The flow setup and bandwidth reservation of AN3 is related to traditional Quality of Service (QoS) features such as integrated [12] and differentiated services [9]. A key difference is that AN3 initiates flows directly from the source NICs; therefore, no traffic shaping is required at the intermediate switches, resulting in a simpler switch design. Speculative transmission also eliminates the round trip latency required in traditional bandwidth reservation protocols such as RSVP [40].

AN3 is not the first proposal to introduce circuits to datacenter networks. Two recent proposals (Helios [15] and c-Through [36]) employ hybrid electric/optical switch technologies to boost the performance of datacenter networks. Our approach is similar, but we eliminate packet switching altogether. In addition, we set up per-flow circuits, which are much more fine-grained than the circuits that are shared among many flows in hybrid switches. Our circuits are set up electronically within tens of microseconds—two to three orders faster than the commercial optical switches used in [15]. Hence, our switch supports short and bursty flows in a more flexible and efficient manner.

With AN3's fine-grained circuit switching, each flow knows its allocated bandwidth, and there is no congestion-related packet loss. Therefore, the network is much simpler, eliminating special efforts needed to improve TCP performance on datacenter environments, e.g., DCTCP [5], or additional

methods to isolate networking usage between tenants. Given that circuits provide predictive performance, resource allocation schemes such as Oktopus [7], SecondNet [18], Gatekeeper [31], SeaWall [32] and Faircloud [28] are easily applied to AN3. We also envision that it is easier to deploy software defined networks such as Open Flow [26] on top of AN3.

AN3 allows end hosts to communicate directly, which is similar to server switch [25], Dcell [19], and BCube [17]. AN3 takes the additional step of eliminating the expensive TOR switches, allowing egress NICs in a half-rack to communicate directly with the L1 switch.

AN3 can also be used to access and transfer data from remote storage. In the storage context, Infiniband [6] is a high-performance packet switched network developed for supercomputer clusters but has seen adoption for accessing remote storage in networked datacenters. Infiniband offers high performance, reliable delivery, remote DMA, and a multicast capability. Although it is feasible to construct a large Infiniband network, it is expensive because Infiniband employs a two-level structure that requires free-standing switches in the interior of the network to route cells between subnets. AN3 eliminates the interior switches (i.e. the usual TOR switches), and therefore, scales more gracefully and costs significantly less.

10. Conclusions

The AN3 network is a radical departure from current practice. We have shown that it works efficiently for different types of traffic patterns. Beside performance benefits, it provides a substantial reduction in the capital cost with the removal of top of rack switches, the replacement of optical cables with SATA cables, and the use of FPGA for switches. AN3 also simplifies operation, since the NOC has full visibility and complete control of all the elements in the network. We believe AN3 can be a candidate design for next generation data center networks.

References

[1] "Cisco Nexus 5020 Switch, http://www.cisco.com/en/US/products/ps9710/index.html."

[2] "Data Center Design Considerations, http://www.cisco.com/en/US/docs/solutions/Enterprise."

[3] "Removed for blind submission."

[4] "Data Center In a Box," in *Scientific American*, Aug 2007.

[5] M. Alizadeh *et al.*, "Data center tcp (dctcp)," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 63–74, 2010.

[6] I. T. Association, *InfiniBand Architecture Specification: Release 1.2.1*. InfiniBand Trade Association, 2008.

[7] H. Ballani *et al.*, "Towards predictable datacenter networks," in *ACM SIGCOMM*, 2011.

[8] H. Bazzaz *et al.*, "Switching the optical divide: Fundamental challenges for hybrid electrical/optical datacenter networks," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 30.

[9] S. Blake *et al.*, "An architecture for differentiated services," 1998.

[10] Y. Chen *et al.*, "Understanding tcp incast throughput collapse in datacenter networks," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 73–82.

[11] Cisco, "Data Center Design–IP Network Infrastructure," Nov 2011.

[12] D. Clark, R. Braden, and S. Shenker, "Integrated services in the internet architecture: an overview," 1994.

[13] M. de Prycker, *Asynchronous transfer mode: solution for broadband ISDN*. Upper Saddle River, NJ, USA: Ellis Horwood, 1991.

[14] L. Dryburgh and J. Hewett, *Signaling System No. 7 (SS7/C7): protocol, architecture, and services*. Cisco Press, 2005.

[15] N. Farrington *et al.*, "Helios: a hybrid electrical/optical switch architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 339–350, 2011.

[16] A. Greenberg *et al.*, "Vl2: a scalable and flexible data center network," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4. ACM, 2009, pp. 51–62.

[17] C. Guo *et al.*, "Bcube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.

[18] C. Guo *et al.*, "Secondnet: a data center network virtualization architecture with bandwidth guarantees," in *Proceedings of the 6th International COnference*. ACM, 2010, p. 15.

[19] C. Guo *et al.*, "Dcell: a scalable and fault-tolerant network structure for data centers," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 75–86.

[20] J. Hamilton, "Talk: Datacenter Networks Are In My Way - Principals of Amazon," Oct 2010.

[21] A. Joel, "Asynchronous transfer mode switching," *IEEE, New York*, 1993.

[22] K. Lakshminarayanan *et al.*, "End-host controlled multicast routing," *Computer Networks*, vol. 50, no. 6, pp. 807–825, 2006.

[23] L. Lamport, "Specifying systems: The tla+ language and tools for hardware and software engineers. 2002."

[24] S. Long *et al.*, "Call performance studies on the atm forum uni signalling implementations," 1999.

[25] G. Lu *et al.*, "Serverswitch: A programmable and high performance platform for data center networks," in *Proc. NSDI*, 2011.

[26] N. McKeown *et al.*, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[27] A. Phanishayee *et al.*, "Measurement and analysis of tcp throughput collapse in cluster-based storage systems," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies*. USENIX Association, 2008, pp. 1–14.

[28] L. Popa *et al.*, "Faircloud: Sharing the network in cloud computing," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM, 2012, pp. 187–198.

[29] L. Popa *et al.*, "A cost comparison of datacenter network architectures," in *Proceedings of the 6th International COnference*, ser. Co-NEXT '10. New York, NY, USA: ACM, 2010, pp. 16:1–16:12. Available: http://doi.acm.org/10.1145/1921168.1921189

[30] A. Rasmussen *et al.*, "Tritonsort: A balanced large-scale sorting system," in *USENIX NSDI*, vol. 11, 2011.

[31] H. Rodrigues *et al.*, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks," *USENIX WIOV*, 2011.

[32] A. Shieh *et al.*, "Sharing the data center network," in *USENIX NSDI*, vol. 11, 2011.

[33] C. Systems, "Catalyst 8500 csr architecture," http://www.cisco.com/en/US/products/hw/switches/ps718/products_white_paper09186a008009263d.shtml.

[34] C. Systems, "Cisco nexus 5000 series switches," http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9670/data_sheet_c78-461802.pdf.

[35] Telcordia, "Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria," 2009.

[36] G. Wang *et al.*, "c-through: Part-time optics in data centers," in *ACM SIGCOMM*, vol. 10, 2010, pp. 327–338.

[37] H. Wu *et al.*, "Ictcp: Incast congestion control for tcp in data center networks," in *Proceedings of the 6th International COnference*. ACM, 2010, p. 13.

[38] Y. Yu, P. Manolios, and L. Lamport, "Model checking tla+ specifications," *Correct Hardware Design and Verification Methods*, pp. 702–702, 1999.

[39] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: A framework for delivering multicast to end users," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2002, pp. 1366–1375.

[40] L. Zhang *et al.*, "Rsvp: A new resource reservation protocol," *Network, IEEE*, vol. 7, no. 5, pp. 8–18, 1993.