# Scale-Out Networking in the Data Center

Scale-out architectures supporting flexible, incremental scalability are common for computing and storage. However, the network remains the last bastion of the traditional scale-up approach, making it the data center's weak link. Through the UCSD Triton network architecture, the authors explore issues in managing the network as a single plug-and-play virtualizable fabric scalable to hundreds of thousands of ports and petabits per second of aggregate bandwidth.

**Amin Vahdat**
**Mohammad Al-Fares**
**Nathan Farrington**
**Radhika Niranjan Mysore**
**George Porter**
**Sivasankar Radhakrishnan**
University of California, San Diego

Over the past decade, the scale-out model has replaced scale up as the basis for delivering large-scale computing and storage. In the traditional scale-up model, system architects replace a server with a higher-end model upon reaching the device's compute or storage capacity. This model requires periodically moving applications and data from one machine to another. Worse, it eliminates leveraging commodities of scale, meaning that the price per compute cycle or byte of storage can be a factor of 10 or more higher than the unit costs available from commodity components.

This additional cost could be justified in smaller deployments—that is, the simpler administration and software architecture of running on fewer, more powerful devices might reduce the absolute cost difference between scale-out and scale-up service deployment. However, planetary-scale infrastructures serving billions of requests daily[1] and cloud computing infrastructures centrally hosting millions of individuals' and businesses' computing and storage requirements have led to dense data centers with hundreds of thousands of servers, such as the Microsoft Chicago Data Center Container Bay. In this environment, the cost difference between leveraging commodity versus noncommodity components can amount to billions of dollars for the largest providers.

Such economics have justified necessary software architecture development to leverage incrementally deployable, horizontally scalable commodity servers and disks.[2-3] In the scale-out model, adding additional compute or storage resources should ideally be a matter of adding more servers or disks to the system. For instance, increasing a service cluster's compute capacity by 10 percent should require adding only 10 percent more servers to the cluster. The scale-out model requires parallelism in the underlying workload, but this required parallelism is trivially available for storage for services handling many small requests from a large user population, such as search, and increasingly available for various computational models.[2]

Although the scale-out model provides clear and compelling benefits, the last bastion of the scale-up model in data centers is the network. Delivering the necessary parallelism for both compute and storage requires coordination and communication among many elements spread across the data center. In turn, this cooperation requires a high-performance network interconnect at the scale of tens of thousands of servers. Today, however, delivering bandwidth to ever-larger clusters requires leveraging increasingly specialized, noncommodity switches and routers. For example, servers within a single rack can interconnect using a commodity 48-port nonblocking top-of-rack switch. All hosts directly connected to this first-hop switch can communicate with one another at their local network interface card's (NIC's) speed. However, high-speed communication at the scale of thousands of racks requires a hierarchical network. Although the hierarchy's leaves can use commodity switches, aggregation up the hierarchy requires higher capacity and specialized switches (with more ports and more backplane bandwidth). Existing datacenter topologies might increase cost and limit performance by a factor of 10 or more while limiting end-host connectivity to 1 gigabit per second (Gbps).[4]

The resulting performance limitations in the datacenter network limit the data center's flexibility and scalability as a whole. For instance, the number of network ports that can economically deliver sufficient aggregate bandwidth among all hosts limits the size of the largest services that can run in the data center. Oversubscribed networks are a reality in data centers of any scale, meaning that application developers must explicitly understand and account for widely variable cluster bandwidth. For example, 1 Gbps of bandwidth can be available to hosts in the same rack, 200 megabits per second (Mbps) to hosts in the same row, and 40 Mbps to hosts in a different row. Limiting available bandwidth in this manner limits various communication-intensive applications' scalability. At the same time, moving existing applications to a new cluster with a different network interconnect can violate hidden assumptions in application code.

Numerous recent efforts are collectively investigating scale-out networking. The goal is to make expanding the number of network ports or the amount of aggregate bandwidth as simple as adding processing power or storage. Operators should be able to manage the datacenter network as a single plug-and-play network, approaching zero manual configuration and transparently bringing online additional connectivity or bandwidth for servers. We'll take a look at the many challenges to realizing scale-out networking, with a particular focus on the University of California, San Diego's Triton network architecture.

## Requirements

An idealized network fabric should be scalable, manageable, flexible, and cost effective.

### Scalable

The network should scale to support the largest data centers: 100,000 or more ports and approaching 1 petabit per second of aggregate bandwidth. Furthermore, the scale should be incrementally realizable—that is, the number of ports should be expandable at the granularity of hundreds of ports and aggregate bandwidth scalable at the granularity of terabits per second.

### Manageable

Developers should be able to manage the entire network as a single logical layer 2 (L2) domain, realizing all of plug-and-play deployment's benefits of additional ports or bandwidth. Expanding the network shouldn't require configuring Dynamic Host Configuration Protocol (DHCP) servers, subnet numbers, subnet masks, or network maps. At the same time, one of the perhaps surprising management challenges with large-scale networks is cabling complexity. Avoiding the cost and complexity of interconnecting tens of thousands of long cables crisscrossing the data center is often reason enough to oversubscribe a network.

### Flexible

Any service should be flexible enough to run anywhere in the data center with full support for both end-host and network virtualization. Virtual machines should be able to migrate to any physical machine while maintaining their IP addresses and reachability. Datacenter architects should be able to allocate larger computations on the basis of resource availability rather than interrack bandwidth limitations.

### Cost effective

The price per port should be cost effective—that is, small relative to the cost of compute and storage, in terms of both capital and operational expenditures. The latter requires reducing not only the human configuration and management burden but also the per-port power consumption.[5]

### Triton

Addressing these challenges forms the basis of our Triton network architecture and implementation, which comprises

- a fundamentally scalable network topology as the basis for cost-effective bandwidth scalability,
- a hardware organization that enables incremental deployment of discrete, identical network elements with minimal cabling complexity,
- a set of protocols that network switches can use to self-discover their location in the network topology, as well as the available routes between all end hosts, and
- dynamic forwarding techniques that load-balance communication among all available paths, delivering the underlying fabric's full bandwidth for a range of communication patterns.

Taken together, our work aims to demonstrate one instance of a hardware, software, and protocol organization to deliver scale-out networking.

## Topology

Traditional best practices in building datacenter networks such as the Cisco Data Center Infrastructure 2.5 Design Guide describe a tree topology with aggregation to denser, higher-speed switches moving up the hierarchy. The highest density switches available for the topology's root limits the network performance in these environments. Until recently, nonblocking 10-Gbps Ethernet switches were limited to 128 ports. Numerous product announcements claim increased density to 200 to 300 ports. Assuming servers with Gbps NICs, this would limit a single network's size to a few thousand hosts. Expanding further is possible by building a tree with multiple roots and then using techniques such as equal-cost multipath forwarding (ECMP) to balance load among these roots.

Overall, however, this scale-up approach to datacenter networking has many drawbacks. First, it requires aggregation to higher-speed, denser switches as its fundamental scaling mechanism. For example, in transitioning to 10-Gbps NICs, this approach requires 40-Gbps or 100-Gbps Ethernet switches. Unfortunately, the time to transition from one generation of Ethernet technology to the next has been increasing. In 1998, 1-Gbps Ethernet debuted; in 2002, 10-Gbps Ethernet. The widespread transition to 10 Gbps to end hosts, and 40-Gbps Ethernet technology will see commercial deployment in 2010. During the transition period, the highest-end switches command a significant price premium primarily because of limited volumes, substantially increasing the datacenter network's cost.[4] Second, scale-up networking such as this limits the overall bisection bandwidth, in turn impacting the maximum degree of parallelism or overall system performance for large-scale distributed computations.[2]

Issues of cost lead datacenter architects to oversubscribe their networks. For instance, in a typical data center organized around racks and rows, intrarack communication through a top-of-rack switch might deliver nonblocking performance. Communication to a server in a remote rack but the same row might be oversubscribed by a factor of 5. Finally, interrow communication might be oversubscribed by some larger factor. Oversubscription ratios of up to 240 to 1 have occurred in commercial deployments.[6]

Our approach to the challenges with current datacenter networking topologies has been to adopt ideas from the area of high-performance computing.[7] In particular, we have explored the benefits of topologies built around fat trees,[4] a special buffered instance of a folded-Clos topology using identical switching elements and adapted for packet-switched networks. Of course, this general approach to network scaling is not

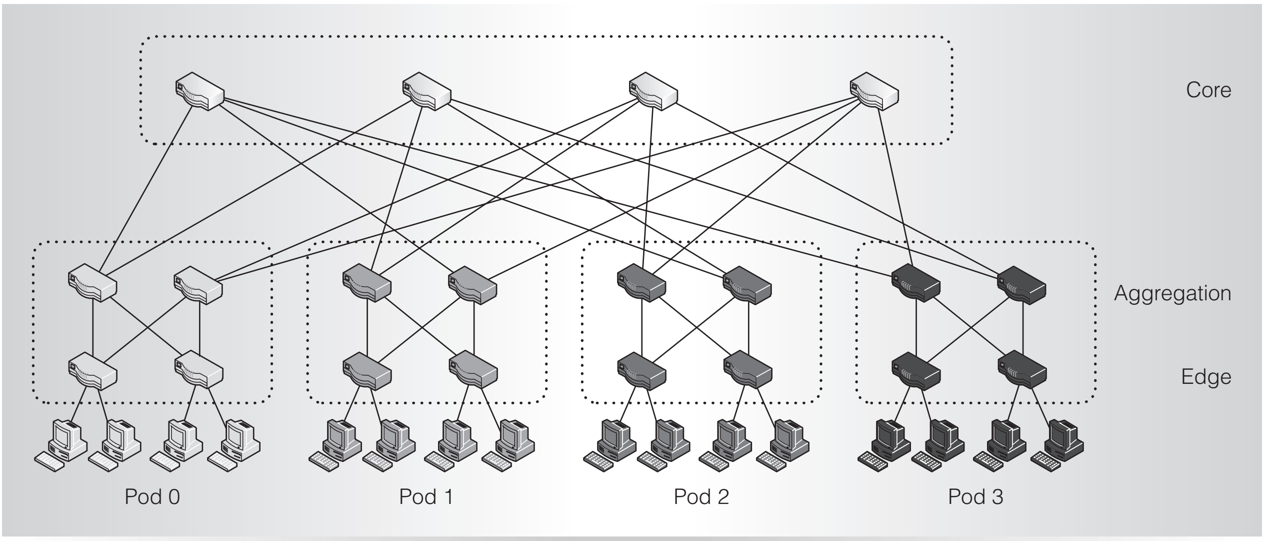


Figure 1. Sample fat tree topology. With appropriate scheduling, this topology can deliver the same throughput as a single-stage crossbar switch. Edge switches in each pod provide connectivity to end hosts and connect to aggregation switches in the hierarchy's second level. Core switches form the root of the fat tree, facilitating interpod communication.

new. However, the need to apply them to large-scale Ethernet and TCP/IP networks running standard Internet services has been limited, until the recent advent of planetary scale services and exploding data sets routinely requiring thousands of servers performing all-to-all communication.

Figure 1 shows a simple example of a fat tree topology. The topology effectively forms a 16-port switch built internally from 20 identical 4-port switches. With appropriate scheduling, this topology can deliver the same throughput as a single-stage crossbar switch. Put another way, it can achieve aggregate bisection bandwidth limited only by the speed of each server's network interface. (The analogous circuit-switched network would be rearrangeably nonblocking.) Achieving this functionality in practice might be computationally challenging, but we have achieved good results in practice, as we'll discuss in the "Multipath Forwarding" section.

The fat tree's scaling properties make it particularly appealing for the data center. In general, a fat tree built from identical $k$-port switches can support 100 percent throughput among $k^3/4$ servers using $k^2/4$ switching elements. We organize the topology into $k$ pods, as Figure 1 shows, each connecting $k^2/4$ end hosts. Edge switches in each pod provide connectivity to end hosts and connect to aggregation switches in the hierarchy's second level. Core switches form the root of the fat tree, facilitating interpod communication.

Although a fat tree can scale out to a nonblocking topology, it can also be oversubscribed depending on the underlying applications' communication requirements. For example, if the fat tree in Figure 1 had only two core switches, the network as a whole would be oversubscribed by a factor of two. Similarly, limiting the number of aggregation switches could oversubscribe each pod.

Critically, the fat tree is built from identical switching elements, so relying on aggregation to higher-speed, more expensive switching elements moving up the hierarchy is unnecessary. This approach demonstrates how to build a cost-effective network fabric from prevailing commodity network components. As higher-speed network elements transition to becoming commodity, fat trees can still support incremental network evolution. For instance, inserting higher-speed switches into the core switching layer can reduce oversubscription. Next, such elements might migrate into pods at the aggregation layer delivering more bandwidth into each pod. Finally, incorporating these switches into the edge can deliver more bandwidth to individual servers. This type of scale-out networking would require carefully designing the modular switching infrastructure, which we'll discuss in the next section.

Our evaluation of a fat tree-based topology's cost structure relative to existing best practices indicates a potential reduction in costs by a factor of 10 or more for the same network bandwidth.[4] Of course, the fat tree topology introduces its own challenges in cabling, management, and routing and forwarding. See the "Related work in scale-out networking" sidebar for more on topologies in this area.

## Merchant silicon

Two challenges of deploying large-scale network fabrics are managing a large network of switching elements as a single, logical switch, and the sheer number of long cables required to interconnect such topologies. For example, many so-called merchant silicon vendors, such as Fulcrum Microsystems, Marvell, and Broadcom, are currently shipping 24-, 48-, or 64-port 10 Gigabit Ethernet (GigE) switches on a chip that could serve as the building block for the fat-tree topologies we've described. Considering the $k = 64$ case, it becomes possible to build a 65,536-port 10 GigE switch partitioned into 64 pods of 1,024 ports each. Unfortunately, the topology would also require 5,120 individual 64-port switching elements and 131,072 cables running across the data center to interconnect edge switches to aggregation switches and aggregation switches to core switches (see Figure 1). These cables are in addition to the short cables required to connect end hosts to the edge switches.

Managing 5,120 individual switches and 131,072 individual cables is a tremendous logistical challenge. So, although a fat-tree topology might reduce a large-scale network's capital expenses, this advantage might be lost because of operational expenses in building and managing the infrastructure.

Alternatively, we believe that a next-generation scale-out network infrastructure will involve constructing larger-scale modular components using available merchant silicon as an internal building block. We developed a modular switch infrastructure along this model.[8] (See the "Related work in scale-out networking" sidebar for additional commercial switch vendors employing merchant silicon.) Our earlier work showed how to construct a 3,456-port 10 GigE switch using then-commodity 24-port 10 GigE chips as the basic switching element. With 64-port switches as the fundamental building block, our exemplar deployment would comprise 64 modular pod switches and one modular core switch array as the basis for scale-out networking (Figure 2). A single cable comprising multiple optical fiber strands would connect each pod switch to the core switch array. Thus, we reduce the previously required 5,120 individual switches to 64 pod switches and one core switch array, and the 131,072 required cables to 64.

In one possible hardware configuration, each pod switch consists of up to 16 line cards, each delivering 64 ports of 10 GigE host connectivity. The pod switch also supports up to eight fabric/uplink cards. Each uplink card delivers 1.28 terabits per second (Tbps) of bisection bandwidth to the pod switch. Installing fewer uplink cards creates a less costly but internally oversubscribed pod switch. We use identical line cards and identical uplink cards to support our vision for scale-out networking: installing additional commodity components delivers additional capacity, whether ports or aggregate bisection bandwidth.

Because copper cables are power-limited to approximately 10 meters in length for 10 GigE, the uplink cards use optical transceivers. This removes the 10-meter limitation, but, just as importantly, provides opportunities for reducing the numbers of cables. Each optical transceiver module takes eight independent 10 GigE channels, encodes them onto different optical wavelengths, and transmits them over a single strand of fiber. Up to 128 such fibers aggregate into a single cable, which then runs to the core switch array. An identical optical module on the core switch array then demultiplexes the original 10 GigE channels and routes them to the appropriate electrical switch in the core array.

In our design, the modular core switch array supports up to 64 line cards, each providing 10.24 Tbps of bandwidth for inter-pod communication. Internally, each line

## Related work in scale-out networking

In developing our approach to scale-out networking, we examined four areas: topology, merchant silicon, layer 2 versus layer 3 routing, and multipath forwarding. Related work in these areas includes the following.

### Topology

VL2 employs a folded-Clos topology to overcome the scaling limitations of traditional tree-based hierarchies.[1] DCell[2] and BCube[3] let end hosts also potentially act as switches, allowing the datacenter interconnect to scale using a modest number of low-radix switching elements.

### Merchant silicon

Several commercial switches are logically organized internally as a three-stage folded-Clos topology built from merchant silicon. As two examples, Arista has built a 48-port 10 Gigabit Ethernet (GigE) switch (www.aristanetworks.com/en/7100 Series SFPSwitches) and Voltaire a 288-port 10 GigE switch (www.voltaire.com/Products/Ethernet/voltaire_vantage_8500), both employing 24-port 10 GigE switch silicon as the fundamental building block.

### Layer 2 versus layer 3 fabrics

Transparent Interconnection of Lots of Links (TRILL) is an IEEE standard for routing at layer 2 (L2).[4] It requires global knowledge of flat memory access controller (MAC) addresses and introduces a separate TRILL header to protect against L2 loops and to limit the amount of per-switch forwarding state. Scalable Ethernet Architecture for Large Enterprises (SEATTLE) employs a distributed hash table among switches to ease determining the mapping between end host and egress switch.[5] Multilevel Origin-Organized Scalable Ethernet (MOOSE) introduces hierarchical MAC addresses similar to our approach with PortLand.[6] Finally, VL2 performs IP-in-IP encapsulation to translate an application IP address (AA) into a host's actual IP address as its local switch (LA) determines.

### Multipath forwarding

VL2 and Monsoon[7] propose using valiant load balancing on a per-flow basis, which suffers from the same hash collision problem as static equal-cost multipath forwarding. Per-packet load-balancing should overcome the hash collision problem, however, variable path latencies can result in out-of-order packet delivery, which can substantially reduce TCP performance.

Traffic Engineering Explicit Control Protocol (TeXCP)[8] and MPLS Adaptive Traffic Engineering (MATE)[9] propose dynamic and distributed traffic engineering techniques to route around congestion in the wide area. FLARE, a flowlet aware routing engine, proposes the idea of grouping back-to-back TCP segments into flowlets that can independently follow different wide-area paths.[10] Several proposals of TCP-variants[11,12] aim to overcome current TCP reordering limitations.

### References

1. A. Greenberg et al., ''VL2: A Scalable and Flexible Data Center Network,'' *Proc. ACM Special Interest Group on Data Communication Conf. Data Comm.* (SIGCOMM 09), ACM Press, 2009, pp. 51-62.
2. C. Guo et al., ''DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers,'' *Proc. ACM Special Interest Group on Data Communication (SIGCOMM) Computer Comm. Rev.,* vol. 38, no. 4, 2008, pp. 75-86.
3. C. Guo et al., ''BCube: A High Performance, Server-Centric Network Architecture for Modular Data Centers,'' *Proc. ACM Special Interest Group on Data Communication Conf. Data Comm.* (SIGCOMM 09), ACM Press, 2009, pp. 63-74.
4. J. Touch and R. Perlman, *Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement,* IETF RFC 5556, May 2009; www.rfc-editor.org/rfc/rfc5556.txt.
5. C. Kim, M. Caesar, and J. Rexford, ''Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises,'' *Proc. ACM Special Interest Group on Data Communication Conf. Data Comm.* (SIGCOMM 08), ACM Press, 2008, pp. 3-14.
6. M. Scott and J. Crowcroft, ''MOOSE: Addressing the Scalability of Ethernet,'' poster session, ACM Special Interest Group on Operating Systems (SIGOPS) European Conf. Computer Systems, 2008; www.cl.cam.ac.uk/~mas90/MOOSE/EuroSys2008-Poster+abstract.pdf.
7. A. Greenberg et al., ''Towards a Next Generation Data Center Architecture: Scalability and Commoditization,'' *Proc. ACM Workshop Programmable Routers for Extensible Services of Tomorrow* (PRESTO 08), ACM Press, 2008, pp. 57-62.
8. S. Kandula et al., ''Walking the Tightrope: Responsive Yet Stable Traffic Engineering,'' *ACM Special Interest Group on Data Communication (SIGCOMM) Computer Comm. Rev.,* vol. 35, no. 4, 2005, pp. 253-264.
9. A. Elwalid et al., ''MATE: MPLS Adaptive Traffic Engineering,'' *Proc. IEEE Int'l Conf. Computer Comm.* (INFOCOM 01), IEEE Press, vol. 3, 2001, pp. 1300-1309.
10. S. Sinha, S. Kandula, and D. Katabi, ''Harnessing TCPs Burstiness Using Flowlet Switching,'' *Proc. ACM Special Interest Group on Data Communication (SIGCOMM) Workshop Hot Topics in Networks* (HotNets 04), ACM Press, 2004; http://nms.csail.mit.edu/papers/flare-hotnet04.ps.
11. M. Zhang et al., ''RR-TCP: A Reordering-Robust TCP with DSACK,'' *Proc. IEEE Int'l Conf. Network Protocols* (ICNP 03), IEEE CS Press, 2003, p. 95.
12. S. Bohacek et al., ''A New TCP for Persistent Packet Reordering,'' *IEEE/ACM Trans. Networking (TON),* vol. 14, no. 2, 2006, pp. 369-382.

card comprises 16 identical 64-port switches. Datacenter operators could deploy as many core line cards as required to support the interpod communication requirements for their particular application mix. So, the network as a whole could be oversubscribed by up to a factor of 64 with one core switch array card or deliver full bisection bandwidth with 64 cards. For fault tolerance, operators can physically distribute the core switch array in different portions of the data center and on different electrical circuits.

Our architecture involving pod switches interconnected by a core switch array aligns well with recent trends toward modular data centers. Currently, many large-scale data centers are being built from individual containers with 100 to 1,000 servers.[9] Our modular pod switch maps well to the networking requirements of individual containers. The core switch array then provides an incrementally bandwidth-scalable interconnect for a variable number of containers. As we add additional containers to a constantly evolving data center, additional core switch array cards can support the expanding communication requirements.

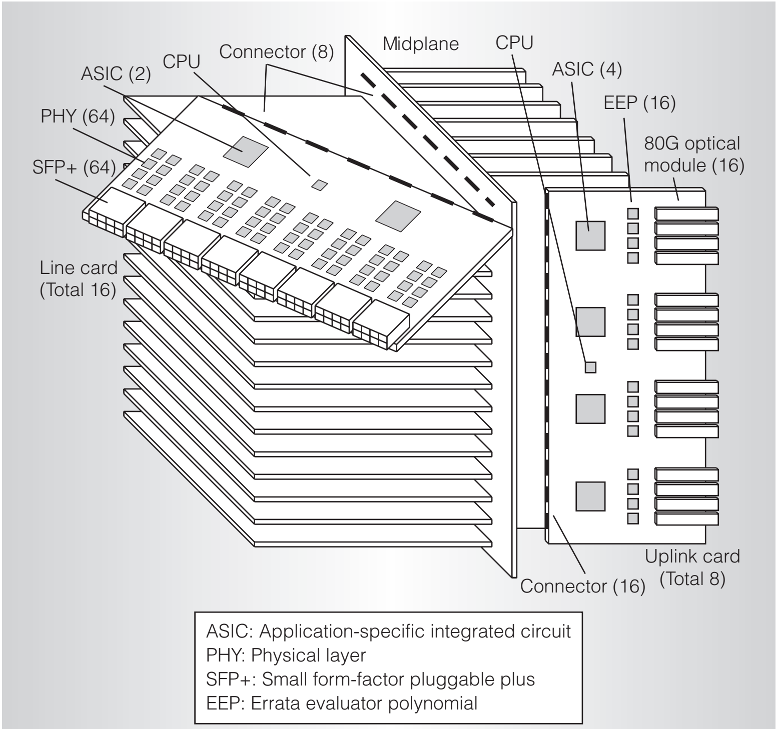


Figure 2. Modular switch infrastructure. A 1,024-port 10 Gigabit Ethernet pod switch with 10.24 terabits per second (Tbps) of uplink capacity to a core switch array. We can combine up to 64 such switches to form a fabric with 655 Tbps of bisection bandwidth.

## Layer 2 versus layer 3 fabrics

Once any datacenter network is constructed, independent of its underlying topology or modularity, the next question is whether the network should be managed as an L2 or L3 addressing domain. Each approach has an associated set of trade-offs for manageability, scalability, switch state, and support for end host virtualization (see Table 1). An L2 network is essentially plug-and-play: the datacenter operator deploys individual switches, and the switches learn their position in the extended local area network through a protocol such as Rapid Spanning Tree (www.ieee802. org/1/pages/802.1w.html). With L3, the operator must configure each switch with a subnet mask and appropriately synchronize any DHCP servers such that they can assign hosts suitable IP addresses matching the subnet of the connected switch. A minor configuration error can render portions of the data center inaccessible.

Network scalability is related to the required control protocols' overhead, primarily the routing protocols. Popular routing protocols, such as Intermediate System-to-Intermediate System (IS-IS) routing for L2 and Open Shortest Path First (OSPF) routing for L3, require broadcast among all network switches to discover the network topology and end host locations, limiting system scalability and stability. L2 forwarding protocols face the additional challenge of distributing information for all end hosts because naming is based on flat memory access controller (MAC) addresses rather than hierarchical IP addresses.

At L2, forwarding based on flat MAC addresses imposes significant scalability challenges on commodity switch hardware. Modern data centers might host more than 100,000 servers with millions of virtual end hosts. Switch forwarding tables must include an entry for every flat MAC address in the data center. Unfortunately, one million forwarding table entries would require 10 Mbytes of on-chip memory (assuming

Table 1. Comparing traditional approaches.

| Approach | Required configuration | Routing | Switch state | Seamless virtual machine migration |
|---|---|---|---|---|
| Layer 2 | Plug and play | Flooding | Large | Supported |
| Layer 3 | Subnet configuration | Broadcast-based | Small | Not supported |

10 bytes per entry), which would consume nearly the entire transistor count of a modern commodity switch ASIC. In fact, current network switches, such as Cisco Nexus 5000 Series (www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9670/white_paper_c11-462176.html) and Fulcrum Microsystems' FM4000 Series (www.fulcrummicro.com/products/focalpoint/fm4000. htm), are limited to between 16,000 and 64,000 forwarding table entries. An L3 network employs topologically meaningful hierarchical IP addresses, so forwarding tables are compactable to have single entries for multiple hosts.

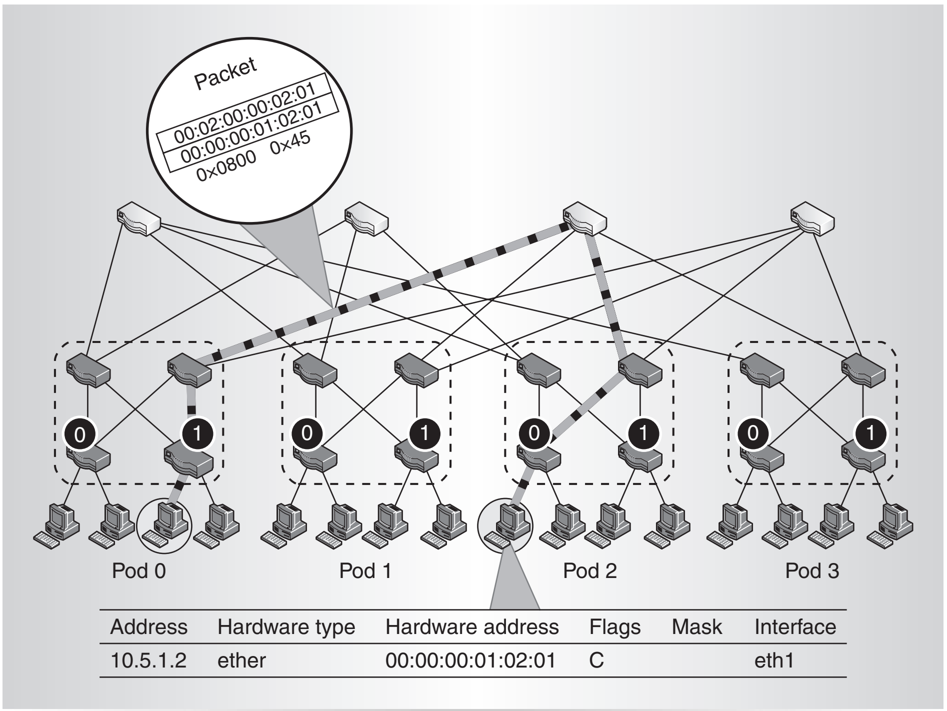
Finally, prevalent use of virtual machines in the data center imposes novel requirements on the networking infrastructure. An IP address uniquely identifies an end host, virtual or otherwise. An end host can cache another end host's IP address (delivering some service) in application-level state and name TCP connections in part by the IP address of the connection's source and destination. At the same time, in an L3 network, the subnet number of a host's first-hop switch partly determines a host's IP address. A virtual machine migrating from one physical machine to another across the data center must then potentially change its IP address to match the appropriate subnet number. This remapping is a key requirement for the hierarchical forwarding we discussed earlier. Unfortunately, adopting a new IP address then can invalidate significant network-wide state and all existing TCP connections. An L2 network won't face this limitation because any physical machine can host any IP address.

In summary, neither L2 nor L3 network protocols are currently a good match to datacenter requirements. However, we believe the general spirit of L2 networks, ease of configuration and management, are in principal better aligned with datacenter network requirements. So, we designed PortLand to combine the benefits of both approaches without the need to modify end hosts and switch hardware.[10] (See the "Related work in scale-out networking" sidebar for additional work in this area.)

We leverage the known technique of separating host identity from host location. In PortLand, the network automatically self-organizes to assign topologically meaningful location labels to every end host. These labels efficiently encode location, with hosts closer to one another in the topology sharing more specific location prefixes. As Figure 3 shows, all packet forwarding proceeds on the basis of these location labels, with hierarchical location labels enabling compact switch forwarding tables.

To maintain transparent compatibility with unmodified end hosts, we insert these labels as L2 pseudo MAC (PMAC) addresses. PortLand switches employ a decentralized Location Discovery Protocol (LDP) to learn their pod membership. They use this information to assign themselves a unique pod number (for aggregation and edge switches) and a unique position number within each pod (for edge switches). Edge switches then assign to each end host 48-bit PMACs of the form pod.position.port.vmid. They learn pod and position values through LDP: the port field is based on the port to which the end host is connected. The edge switches also assign vmids to each virtual machine resident on the same end host based on the virtual machine's unique MAC addresses.

Edge switches transmit IP-address-to-PMAC mappings to a centralized fabric manager on seeing the first packet from any end host. Our insertion point for backward compatibility is the ubiquitous Ethernet Address Resolution Protocol (ARP). Hosts in the same L2 domain will broadcast ARP requests to determine the MAC address associated with a particular host's IP address.



| Address | Hardware type | Hardware address | Flags | Mask | Interface |
|---|---|---|---|---|---|
| 10.5.1.2 | ether | 00:00:00:01:02:01 | C | | eth1 |

Figure 3. Hierarchical forwarding in PortLand. PortLand separates host identity and host location.

PortLand edge switches intercept these ARP requests and proxy them to the fabric manager, which returns the appropriate PMAC (rather than actual MAC address) for the IP address. The edge switch returns this PMAC to the querying host. From this point, all packets include the correct destination PMAC in the Ethernet header. Egress edge switches rewrite the PMAC to the actual MAC address on the basis of locally available soft state, maintaining transparency.

Our implementation requires modifications only to switch software and leverages already available hardware functionality to forward based on PMACs, to intercept ARPs, to set up soft-state mappings on seeing the first packet from end hosts, and so on. We employ OpenFlow to interface with switch software, with the hope of enabling compatibility with various hardware switch platforms.[11] Numerous additional protocol details support virtual machine migration, fault tolerance, and multicast.[10]

## Multipath forwarding

The final component we will consider in scale-out networking is packet forwarding in networks with multiple paths. Modern datacenter topologies must contain multiple paths between hosts to deliver the necessary aggregate bandwidth and to tolerate failures. Unfortunately, traditional L3 Internet routing protocols are optimized for delivering baseline connectivity rather than for load balancing among a set of available paths.

ECMP forwarding balances flows among multiple network paths. An ECMP-capable switch matches a packet's destination address against the forwarding table. To choose among possible next hops, the switch computes a hash of a predefined set of fields in the packet header modulo the number of possible paths to choose the particular next hop. By hashing on fields such as source and destination address, source and destination port number, and time to live, ECMP ensures that packets belonging to the same flow follow the same path through the network, mitigating concerns over packet reordering endemic to multipath forwarding protocols.

Although a step in the right direction, ECMP is oblivious to dynamically changing communication patterns. For any static hash algorithm, multiple flows can collide on a congested link, as Figure 4 depicts. For example, in our earlier work, analysis shows that

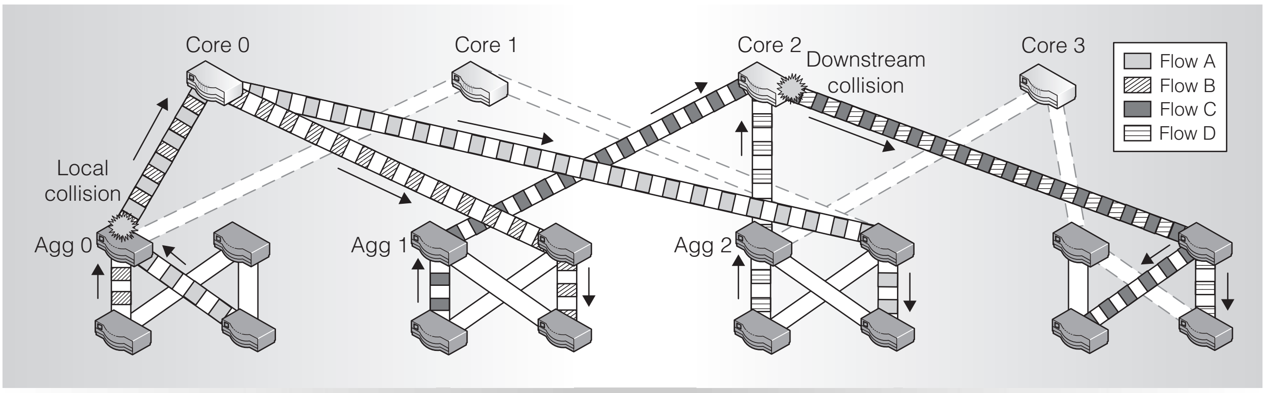


Figure 4. Equal-cost multipath (ECMP) forwarding collisions. As in this example, multiple flows can collide on a congested link, resulting in reduced throughput. We omitted unused links for clarity.

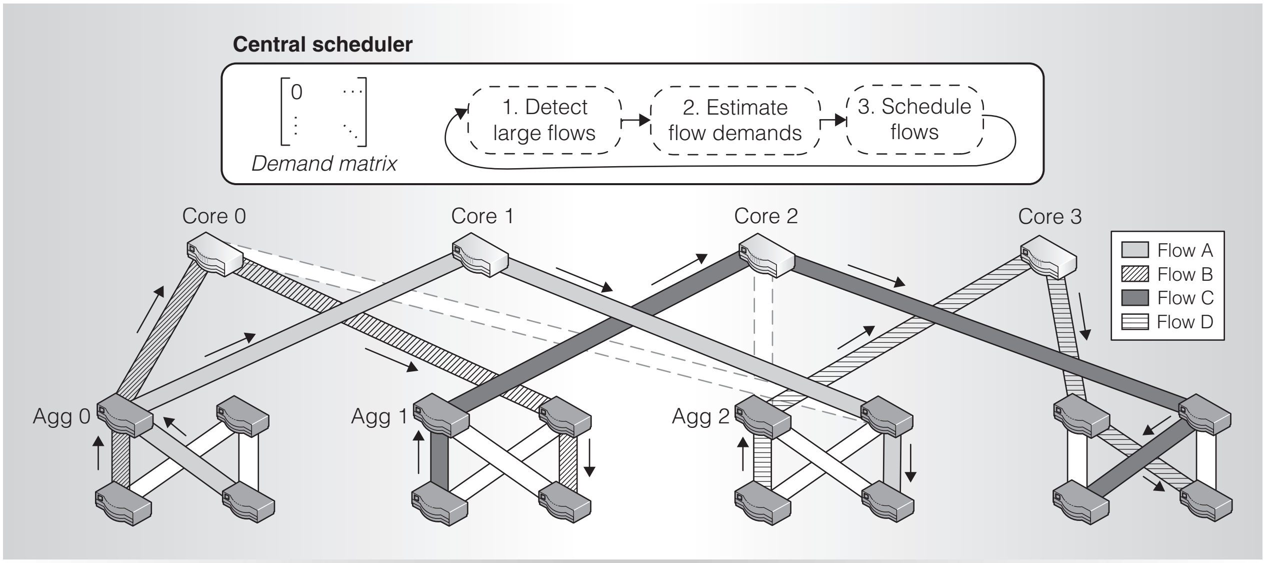


Figure 5. Dynamic flow placement with Hedera avoids long-term flow collisions.

we can restrict ECMP to less than 48 percent of the available bandwidth for some communication patterns, such as serial data shuffles (for example, common to MapReduce[2]).[12]

To address this limitation, we designed and implemented Hedera to perform dynamic flow scheduling in data centers with significant multipathing (see Figure 5).[12] Two observations motivated us in designing Hedera. First, when flows are uniform in size and their individual rates are small, ECMP works well in distributing flows given a reasonable hash function. Problems arise with ECMP's static hashing when flow length varies. In this case, a relatively small number of large flows can account for most bandwidth. So, good load balancing requires efficiently identifying and placing these large flows among the set of available paths between source and destination. Our second observation was that it's insufficient to perform load balancing on the basis of measured communication patterns, because poor scheduling can mask the actual demand of a congestion-responsive flow (such as a TCP flow). Good scheduling then requires a mechanism to estimate a flow's intrinsic bandwidth demand independent of any bottlenecks the network topology or past scheduling decisions might introduce.

Given these observations, Hedera operates as follows: A central fabric manager, akin to

NOX,[13] periodically queries switches for statistics for flows larger than a threshold size. We employ OpenFlow to perform such queries to leverage hardware flow counters available in a range of modern switch hardware. Hedera then uses information about the set of communicating source-destination pairs to build a traffic demand matrix. This matrix represents estimated bandwidth requirements of all large flows in the data center, absent any topological or scheduling bottlenecks—that is, the max-min fair share of each flow on an idealized crossbar switch.

This traffic demand matrix, along with the current view of actual network utilization, serves as input to a scheduling/placement algorithm in the fabric manager. Our goal is to maximize aggregate throughput given current communication patterns. In Hedera, new TCP flows default to ECMP forwarding to choose a path to the destination. However, if the fabric manager determines sufficient benefit from remapping the flow to an alternate path, it informs the necessary switches along the path to override the default forwarding decision with a new Hedera-chosen path.

Our evaluation indicates it's possible to dynamically shift traffic at the granularity of one second and possibly hundreds of milliseconds with additional optimizations. So, as long as the communication patterns at this granularity are stable, an approach such as Hedera can appropriately schedule flows. Overall, Hedera delivers performance within a few percentage points of optimal for a range of communication patterns. (For more on multipath forwarding, see the "Related work in scale-out networking" sidebar.)

## Future challenges

We have focused on some of the key requirements for scale-out networking in the data center, but many important challenges remain. As services become increasingly tightly coupled, many applications would benefit from low-latency communication. Given current network architectures, machine-to-machine latencies of under one microsecond should be achievable even in multihop topologies. However, achieving such latencies will require reexamining hardware components down to the physical layer as well as legacy software optimized for wide-area deployments where millisecond latencies are the norm.

Given the bursty nature of datacenter communication patterns, any static oversubscription for a network might be wasteful along one dimension or another. Although a nonblocking datacenter fabric will support arbitrary communication patterns, the cost is wasteful if most links are idle most of the time. More information is required about datacenter communication patterns to design appropriate topologies and technologies that support a range of communication patterns.[14] The ideal circumstance is to allocate bandwidth when and where required without having to preallocate for the worst case.

Bursty communication can take place at multiple time scales. Recent work on the incast problem, in which sudden bandwidth demand from multiple senders to a common destination can overwhelm shallow switch buffers, quantifies issues with fine-grained communication bursts to a single node.[15] The commonality of these microbursts requires enhancements either at the transport layer or at the link layer to avoid negative interactions between bursts of lost packets and higher-level transport behavior.

On the standardization front, to support migration toward a converged, lossless fabric, Ethernet is undergoing significant enhancements, such as IEEE 802.1Qau: Congestion Notification, IEEE 802.1Qaz: Enhanced Transmission Selection, and IEEE 802.1Qbb: Priority-Based Flow Control. The goal is to employ a single network topology to support application scenarios that traditionally aren't well matched to the Ethernet's best-effort nature, such as storage area networks and high-performance computing. Understanding the interactions between emerging L2 standards and higher-level protocols and applications will be important.

Once we achieve a scale-out network architecture, one of the next key requirements will be increased network virtualization. Although we can carve up processing and storage with current technology, the key missing piece for isolating portions of a larger-scale cluster is virtualizing the network. The goal would be to deliver application-level bandwidth and quality-of-service guarantees using virtual switches, with performance largely indistinguishable from the analogous physical switch.

Energy is increasingly important in the data center. The switching fabric might account for only a modest fraction of total datacenter power consumption. However, the absolute total power consumption can still be significant, making it an important target for future architectures aiming to reduce energy consumed per bit of transmitted data.[5,16] Perhaps more importantly, overall datacenter energy consumption might increase when system components such as CPU or DRAM idle unnecessarily waiting on a slow network.

We've discussed the importance of simplifying network fabric management through plug-and-play L2 network protocols. More aggressively, an entire datacenter network comprising thousands of switches should be manageable as a single unified network fabric. Just as datacenter operators can insert and remove individual disks transparently from a petabyte storage array, network switches should act as "bricks" that together make up a larger network infrastructure. Such unification will require both better hardware organization and improved software protocols.

Computing technology periodically undergoes radical shifts in organization followed by relatively sustained periods of more steady improvement in established metrics such as performance. Networking is currently at one of those inflection points, driven by mega data centers and their associated application requirements. Significant work remains to be done to realize the scale-out network model, and we believe that this will be fertile space for both research and industry in the years ahead. MICRO

## References

1. L.A. Barroso, J. Dean, and U. Hoelzle, "Web Search for a Planet: The Google Cluster Architecture," *IEEE Micro,* vol. 23, no. 2, Mar.-Apr. 2003, pp. 22-28.
2. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Proc. Symp. Operating Systems Design & Implementation* (OSDI 04), Usenix Assoc., pp. 137-150.
3. M. Isard et al., "Dryad: Distributed Data-parallel Programs from Sequential Building Blocks," *ACM Special Interest Group on Operating Systems (SIGOPS) Operating Systems Rev.,* vol. 41, no. 3, 2007, pp. 59-72.
4. M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity, Data Center Network Architecture," *ACM Special Interest Group on Data Communication (SIGCOMM) Computer Comm. Rev.,* vol. 38, no. 4, Oct. 2008, pp. 63-74.
5. B. Heller et al., "ElasticTree: Saving Energy in Data Center Networks," *Proc. Usenix Symp. Networked Systems Design and Implementation* (NSDI 10), Usenix Assoc., 2010; www.usenix.org/events/nsdi10/tech/full_papers/heller.pdf.
6. A. Greenberg et al., "VL2: A Scalable and Flexible Data Center Network," *Proc. ACM Special Interest Group on Data Communication Conf. Data Comm.* (SIGCOMM 09), ACM Press, 2009, pp. 51-62.
7. W.D. Hillis and L.W. Tucker, "The CM-5 Connection Machine: A Scalable Supercomputer," *Comm. ACM,* vol. 36, no. 11, 1993, pp. 31-40.
8. N. Farrington, E. Rubow, and A. Vahdat, "Data Center Switch Architecture in the Age of Merchant Silicon," *Proc. IEEE Symp. High Performance Interconnects* (HOTI 09), IEEE CS Press, 2009, pp. 93-102.
9. C. Guo et al., "BCube: A High Performance, Server-Centric Network Architecture for Modular Data Centers," *Proc. ACM Special Interest Group on Data Communication Conf. Data Comm.* (SIGCOMM 09), ACM Press, 2009, pp. 63-74.
10. R.N. Mysore et al., "PortLand: A Scalable, Fault-Tolerant Layer 2 Data Center Network Fabric," *Proc. ACM Special Interest Group on Data Communication Conf. Data Comm.* (SIGCOMM 09), ACM Press, 2009, pp. 39-50.
11. N. McKeown et al., "OpenFlow: Enabling Innovation in Campus Networks," *ACM Special Interest Group on Data Communication (SIGCOMM) Computer Comm. Rev.,* vol. 38, no. 2, Apr. 2008, pp. 69-74.
12. M. Al-Fares et al., "Hedera: Dynamic Flow Scheduling for Data Center Networks,"

*Proc. Usenix Symp. Networked Systems Design and Implementation* (NSDI 10), Usenix Assoc., 2010; www.usenix.org/events/nsdi10/tech/slides/al-fares.pdf.
13. N. Gude et al., ''NOX: Towards an Operating System for Networks,'' *ACM Special Interest Group on Data Communication (SIGCOMM) Computer Comm. Rev.,* vol. 38, no. 3, 2008, pp. 105-110.
14. S. Kandula et al., ''The Nature of Data Center Traffic: Measurements & Analysis,'' *Proc. ACM Special Interest Group on Data Communication (SIGCOMM) Internet Measurement Conf.* (IMC 09), ACM Press, 2009, pp. 202-208.
15. A. Phanishayee et al., ''Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems,'' *Proc. Usenix Conf. File and Storage Technologies* (FAST 08), Usenix Assoc., 2008, pp. 175-188; www.usenix.org/events/fast08/tech/full_papers/phanishayee/phanishayee.pdf.
16. M. Gupta and S. Singh, ''Greening of the Internet,'' *Proc. Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm.,* ACM Press, 2003, pp. 19-26.

**Amin Vahdat** is a professor and holds the Science Applications International Corporation chair in the Department of Computer Science and Engineering at the University of California, San Diego. He's also director of UC San Diego's Center for Networked Systems. His research interests include distributed systems, networks, and operating systems. Vahdat has a PhD in computer science from University of California, Berkeley.

**Mohammad Al-Fares** is a PhD candidate in computer science at the University of California, San Diego, working in the Systems and Networking group. His research interests include building large datacenter network architectures, especially focusing on topology, routing, and forwarding efficiency. Al-Fares has an MS in computer science from UC San Diego. He's a member of ACM and Usenix.

**Nathan Farrington** is a PhD candidate in computer science and engineering at the University of California, San Diego. His research interests include improving datacenter networks' cost, performance, power efficiency, scalability, and manageability. Farrington has an MS in computer science from UC San Diego. He's a member of IEEE and ACM.

**Radhika Niranjan Mysore** is a PhD candidate in computer science at the University of California, San Diego. Her research interests include datacenter networks' scalability, storage, and protocol design and correctness proofs. Mysore has an MS in electrical and computer engineering from Georgia Institute of Technology. She's a member of ACM and Usenix.

**George Porter** is a postdoctoral researcher in the Center for Networked Systems at the University of California, San Diego. His research interests include improving the scale, efficiency, and reliability of large-scale distributed systems, with a current focus on the data center. Porter has a PhD in computer science from the University of California, Berkeley.

**Sivasankar Radhakrishnan** is a graduate student in computer science at the University of California, San Diego. His research interests include designing large-scale datacenter networks that deliver the high performance and efficiency required for planetary scale services. Radhakrishnan has a BTech in computer science and engineering from Indian Institute of Technology, Madras. He's a member of ACM and Usenix.

Direct questions and comments to Amin Vahdat, 9500 Gilman Drive, M/C 0404, University of California San Diego, La Jolla, CA 92093-0404; vahdat@cs.ucsd.edu.