

VL2: A Scalable and Flexible Data Center Network

By Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta

Abstract

To be agile and cost effective, data centers must allow dynamic resource allocation across large server pools. In particular, the data center network should provide a simple flat abstraction: it should be able to take any set of servers anywhere in the data center and give them the illusion that they are plugged into a physically separate, noninterfering Ethernet switch with as many ports as the service needs. To meet this goal, we present VL2, a practical network architecture that scales to support huge data centers with uniform high capacity between servers, performance isolation between services, and Ethernet layer-2 semantics. VL2 uses (1) flat addressing to allow service instances to be placed anywhere in the network, (2) Valiant Load Balancing to spread traffic uniformly across network paths, and (3) end system-based address resolution to scale to large server pools without introducing complexity to the network control plane. VL2's design is driven by detailed measurements of traffic and fault data from a large operational cloud service provider. VL2's implementation leverages proven network technologies, already available at low cost in high-speed hardware implementations, to build a scalable and reliable network architecture. As a result, VL2 networks can be deployed today, and we have built a working prototype. We evaluate the merits of the VL2 design using measurement, analysis, and experiments. Our VL2 prototype shuffles 2.7 TB of data among 75 servers in 395 s—sustaining a rate that is 94% of the maximum possible.

1. INTRODUCTION

Cloud services are driving the creation of huge data centers, holding tens to hundreds of thousands of servers, that concurrently support a large and dynamic number of distinct services (web apps, e-mail, map-reduce clusters, etc.). The case for cloud service data centers depends on a scale-out design: reliability and performance achieved through large pools of resources that can be rapidly reassigned between services as needed. With data centers being built with over 100,000 servers, at an amortized cost approaching \$12 million per month,¹⁴ the most desirable property for a data center is *agility*—the ability to assign any server to any service. Anything less inevitably results in stranded resources and wasted money.

Unfortunately, the data center network is not up to the task, falling short in several ways. First, existing architectures do not provide enough capacity between the servers they interconnect. Conventional architectures rely on treelike network configurations built from expensive

hardware. Due to the high equipment cost, the capacity between different levels of the tree is typically oversubscribed by factors of 1:5 or more, with paths near the root oversubscribed by factors of 1:80 to 1:240. This oversubscription limits communication between servers to the point that it fragments the server pool—congestion and computation hot spots are prevalent even when spare capacity is available elsewhere. Second, while data centers host multiple services, the network does little to prevent a traffic flood in one service from affecting other services around it—when one service experiences a traffic flood, it is common for all those sharing the same network subtree to suffer collateral damage. Third, the routing design in conventional networks achieves scale by assigning servers topologically significant IP addresses and dividing servers up among VLANs. However, this creates an enormous configuration burden when servers must be reassigned among services, further fragmenting the resources of the data center. The human involvement typically required in these reconfigurations undermines speed of deployment.

To overcome these limitations in the current network and achieve agility, we arrange for the network to implement a familiar and concrete model: give each service the illusion that all the servers assigned to it, and only those servers, are connected by a single noninterfering Ethernet switch—a *Virtual Layer 2*—and maintain this illusion even as the size of each service varies from 1 server to 100,000. Realizing this vision for the data center network concretely translates into building a network that meets the following three objectives:

- **Uniform high capacity:** The maximum rate of a server-to-server traffic flow should be limited only by the available capacity on the network-interface cards of the sending and receiving servers, and it should be possible to assign servers to a service without having to consider network topology.
- **Performance isolation:** Traffic of one service should not be affected by the traffic of any other service, just as if each service was connected by a separate physical switch.
- **Layer-2 semantics:** The servers in each service should experience the network as if it were an Ethernet Local Area Network (LAN). Data center management soft-

A previous version of this paper was published in the *Proceedings of SIGCOMM '09* (Barcelona, Spain, Aug. 17–21, 2009). ACM, New York.

ware should be able to assign any IP address the service requests to any server, and virtual machines should be able to migrate to any server while keeping the same IP address. Finally, features like link-local broadcast, on which many legacy applications depend, should work.

We design, implement, and evaluate VL2, a network architecture for data centers that meets these three objectives and thereby achieves agility.

Design philosophy: In designing VL2, a primary goal was to create a network architecture that could be deployed today, so we limit ourselves from making any changes to the hardware of the switches or servers, and we require that legacy applications work unmodified. Our approach is to build a network that operates like a very large switch—choosing simplicity and high performance over other features when needed. We sought to use robust and time-tested control plane protocols, and we avoid adaptive routing schemes that might theoretically offer more bandwidth but open thorny problems that might not need to be solved and would take us away from vanilla, commodity, high-capacity switches.

We observe, however, that the software and operating systems on data center servers are already extensively modified (e.g., to create hypervisors for virtualization or blob file systems to store data across servers). Therefore, VL2's design explores a new split in the responsibilities between host and network—using a layer 2.5 shim in servers' network stack to work around limitations of the network devices. No new switch software or switch APIs are needed.

Topology: VL2 consists of a network built from low-cost switch ASICs arranged into a Clos topology that provides extensive path diversity between servers. This design replaces today's mainframe-like large, expensive switches with broad layers of low-cost switches that can be scaled out to add more capacity and resilience to failure. In essence, VL2 applies the principles of RAID (redundant arrays of inexpensive disks) to the network.

Traffic engineering: Our measurements show data centers have tremendous volatility in their workload, their traffic, and their failure patterns. To cope with this volatility in the simplest manner, we adopt Valiant Load Balancing (VLB) to spread traffic across all available paths without any centralized coordination or traffic engineering. Using VLB, each server independently picks a path at random through the network for each of the flows it sends to other servers in the data center. Our experiments verify that using this design achieves both uniform high capacity and performance isolation.

Control plane: The switches that make up the network operate as layer-3 routers with routing tables calculated by OSPF, thereby enabling the use of multiple paths while using a time-tested protocol. However, the IP addresses used by services running in the data center must not be tied to particular switches in the network, or the ability for agile reassignment of servers between services would be lost. Leveraging a trick used in many systems,⁹ VL2 assigns servers IP addresses that act as names alone, with no topological significance. When a server sends a packet, the shim layer on the server invokes a directory system to learn

the actual location of the destination and then tunnels the original packet there. The shim layer also helps eliminate the scalability problems created by ARP in layer-2 networks, and the tunneling improves our ability to implement VLB. These aspects of the design enable VL2 to provide layer-2 semantics—eliminating the fragmentation and waste of server pool capacity that the binding between addresses and locations causes in the existing architecture.

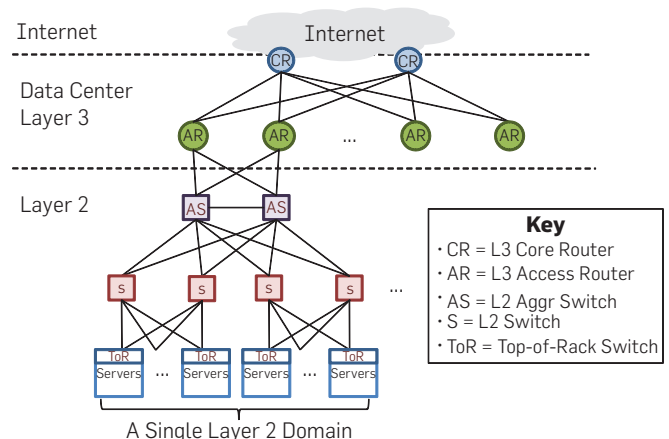
Contributions: In the course of this paper, we describe the current state of data center networks and the traffic across them, explaining why these are important to designing a new architecture. We present VL2's design, which we have built and deployed into an 80-server cluster. Using the cluster, we experimentally validate that VL2 has the properties set out as objectives, such as uniform capacity and performance isolation. We also demonstrate the speed of the network, such as its ability to shuffle 2.7TB of data among 75 servers in 395s (averaging 58.8Gbps). Finally, we describe our experience applying VLB in a new context, the inter-switch fabric of a data center, and show that VLB smooths utilization while eliminating persistent congestion.

2. BACKGROUND

In this section, we first explain the dominant design pattern for data center architecture today.⁵ We then discuss why this architecture is insufficient to serve large cloud-service data centers.

As shown in Figure 1, the network is a hierarchy reaching from a layer of servers in racks at the bottom to a layer of core routers at the top. There are typically 20–40 servers per rack, each singly connected to a Top of Rack (ToR) switch with a 1Gbps link. ToRs connect to two aggregation switches for redundancy, and these switches aggregate further connecting to access routers. At the top of the hierarchy, core routers carry traffic between access routers and manage traffic into and out of the data center. All links use Ethernet as a physical-layer protocol, with a mix of copper and fiber cabling. All switches below each pair of access routers form a single layer-2 domain.

Figure 1. A conventional network architecture for data centers (adapted from figure by Cisco⁵).



layer-2 domain is typically limited to a few hundred due to Ethernet scaling overheads (packet flooding and ARP broadcasts). To limit these overheads and to isolate different services or logical server groups (e.g., e-mail, search, web front ends, web back ends), servers are partitioned into virtual LANs (VLANs) placed into distinct layer-2 domains. Unfortunately, this conventional design suffers from three fundamental limitations:

Limited server-to-server capacity: As we go up the hierarchy, we are confronted with steep technical and financial barriers in sustaining high bandwidth. Thus, as traffic moves up through the layers of switches and routers, the oversubscription ratio increases rapidly. For example, servers typically have 1:1 oversubscription to other servers in the same rack—that is, they can communicate at the full rate of their interfaces (e.g., 1 Gbps). We found that uplinks from ToRs are typically 1:2 to 1:20 oversubscribed (i.e., 1–10 Gbps of uplink for 20 servers), and paths through the highest layer of the tree can be 1:240 oversubscribed. This large oversubscription factor fragments the server pool by preventing idle servers from being assigned to overloaded services, and it severely limits the entire data center’s performance.

Fragmentation of resources: As the cost and performance of communication depends on distance in the hierarchy, the conventional design encourages service planners to cluster servers nearby in the hierarchy. Moreover, spreading a service outside a single layer-2 domain frequently requires the onerous task of reconfiguring IP addresses and VLAN trunks, since the IP addresses used by servers are topologically determined by the access routers above them. Collectively, this contributes to the squandering of computing resources across the data center. The consequences are egregious. Even if there is plentiful spare capacity throughout the data center, it is often effectively reserved by a single service (and not shared), so that this service can scale out to nearby servers to respond rapidly to demand spikes or to failures. In fact, the growing resource needs of one service have forced data center operations to evict other services in the same layer-2 domain, incurring significant cost and disruption.

Poor reliability and utilization: Above the ToR, the basic resilience model is 1:1. For example, if an aggregation switch or access router fails, there must be sufficient remaining idle capacity on the counterpart device to carry the load. This forces each device and link to be run up to at most 50% of its maximum utilization. Inside a layer-2 domain, use of the Spanning Tree Protocol means that even when multiple paths between switches exist, only a single one is used. In the layer-3 portion, Equal Cost Multipath (ECMP) is typically used: when multiple paths of the same length are available to a destination, each router uses a hash function to spread flows evenly across the available next hops. However, the conventional topology offers at most two paths.

3. MEASUREMENTS AND IMPLICATIONS

Developing a new network architecture requires a quantitative understanding of the traffic matrix (who sends how

much data to whom and when?) and churn (how often does the state of the network change due to switch/link failures and recoveries, etc.?). We studied the production data centers of a large cloud service provider, and we use the results to drive our choices in designing VL2. Details of the methodology and results can be found in other papers.^{10,16} Here we present the key findings that directly impact the design of VL2.

Most traffic is internal to the data center: The ratio of traffic volume between servers in our data centers to traffic entering/leaving our data centers is currently around 4:1 (excluding CDN applications). An increasing fraction of the computation in data centers involves back-end computations, and these are driving the demands for network bandwidth.

The network bottlenecks computation: Data center computation is focused where high-speed access to data on memory or disk is fast and cheap. Even inside a single data center, the network is a bottleneck to computation—we frequently see switches whose uplinks are above 80% utilization. Intense computation and communication on data does not straddle data centers due to the cost of long-haul links.

Structured flow sizes: Figure 2 illustrates the nature of flows within the monitored data center. The flow size statistics (marked as “+”s) show that the majority of flows are small (a few KB); most of these small flows are hellos and meta-data requests to the distributed file system. To examine longer flows, we compute a statistic termed *total bytes* (marked as “o”s) by weighting each flow size by its number of bytes. Total bytes tells us, for a random byte, the distribution of the flow size it belongs to. Almost all the bytes in the data center are transported in flows whose lengths vary from about 100MB to about 1GB. The mode at around 100MB springs from the fact that the distributed file system breaks long files into 100-MB-long chunks. Importantly, there are almost no flows over a few GB.

Figure 3 shows the probability density function (as a fraction of time) for the number of concurrent flows going in and out of a machine. There are two modes. More

Figure 2. Mice are numerous; 99% of flows are smaller than 100MB. However, more than 90% of bytes are in flows between 100MB and 1GB.

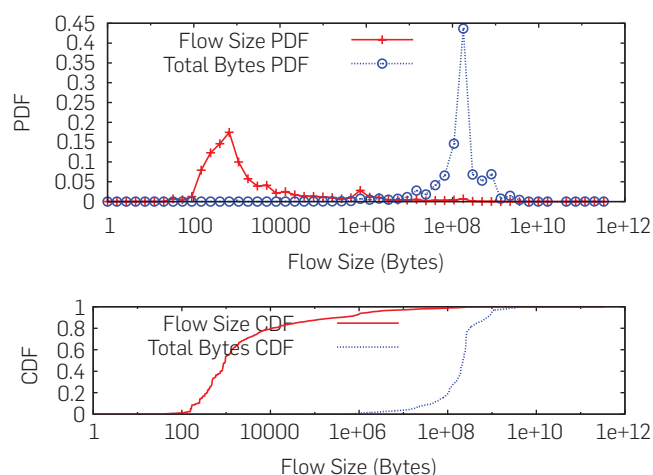
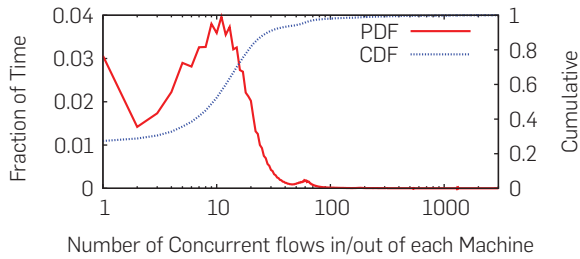


Figure 3. Number of concurrent connections has two modes: (1) 10 flows per node more than 50% of the time and (2) 80 flows per node for at least 5% of the time.



than 50% of the time, an average machine has about ten concurrent flows, but at least 5% of the time it has greater than 80 concurrent flows. We almost never see more than 100 concurrent flows.

The distributions of flow size and number of concurrent flows both imply that flow-based VLB will perform well on this traffic. Since even big flows are only 100MB (1s of transmit time at 1Gbps), randomizing at flow granularity (rather than packet) will not cause perpetual congestion if there is unlucky placement of too many flows in the same link.

Volatile traffic patterns: While the sizes of flows show a strong pattern, the traffic patterns inside a data center are highly divergent. When we cluster the traffic patterns, we find that more than 50 representative patterns are required to describe the traffic in the data center. Further, the traffic pattern varies frequently—60% of the time the network spends only 100s in one pattern before switching to another. **Frequent failures:** As discussed in Section 2, conventional data center networks apply 1 + 1 redundancy to improve reliability at higher layers of the hierarchical tree. This hierarchical topology is intrinsically unreliable—even with huge effort and expense to increase the reliability of the network devices close to the top of the hierarchy, we still see failures on those devices resulting in significant downtime. In 0.3% of failures, all redundant components in a network device group became unavailable (e.g., the pair of switches that comprise each node in the conventional network (Figure 1) or both the uplinks from a switch). The main causes of failures are network misconfigurations, firmware bugs, and faulty components.

With no obvious way to eliminate failures from the top of the hierarchy, VL2’s approach is to broaden the top levels of the network so that the impact of failures is muted and performance degrades gracefully, moving from 1 + 1 redundancy to $n + m$ redundancy.

4. VIRTUAL LAYER 2 NETWORKING

Before detailing our solution, we briefly discuss our design principles and preview how they will be used in our design.

Randomizing to cope with volatility: The high divergence and unpredictability of data center traffic matrices suggest that optimization-based approaches to traffic engineering risk congestion and complexity to little benefit. Instead, VL2 uses VLB: destination-independent (e.g., random) traffic spreading across the paths in the network. VLB, in

theory, ensures a *noninterfering* packet-switched network⁶ (the counterpart of a non-blocking circuit-switched network) as long as (a) traffic spreading ratios are uniform, and (b) the offered traffic patterns do not violate edge constraints (i.e., line card speeds). We use ECMP to pursue the former and TCP’s end-to-end congestion control to pursue the latter. While these design choices do not perfectly ensure the two assumptions (a and b), we show in Section 5.1 that our scheme’s performance is close to the optimum in practice.

Building on proven networking technology: VL2 is based on IP routing and forwarding technologies already available in commodity switches: link-state routing, ECMP forwarding, and IP any-casting. VL2 uses a link-state routing protocol to maintain the switch-level topology, but not to disseminate end hosts’ information. This strategy protects switches from needing to learn voluminous, frequently changing host information. Furthermore, the routing design uses ECMP forwarding along with anycast addresses to enable VLB while minimizing control plane messages and churn.

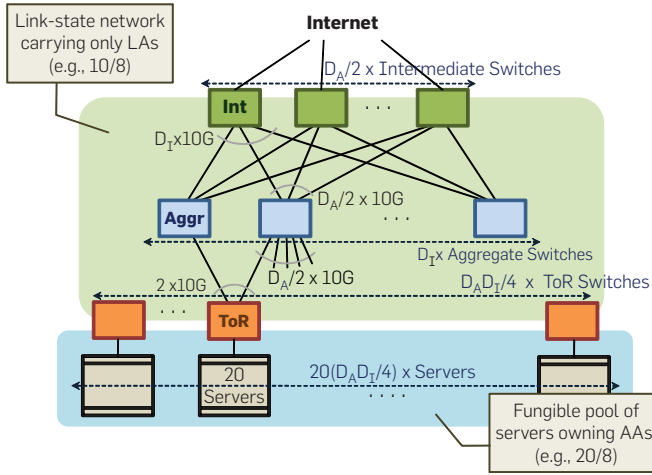
Separating names from locators: To be able to rapidly grow or shrink server allocations and rapidly migrate VMs, the data center network must support agility, which means support hosting any service on any server. This, in turn, calls for separating names from locations. VL2’s addressing scheme separates servers’ names, termed application-specific addresses (AAs), from their locations, termed location-specific addresses (LAs). VL2 uses a scalable, reliable directory system to maintain the mappings between names and locators. A shim layer running in the networking stack on every host, called the VL2 agent, invokes the directory system’s resolution service.

Embracing end systems: The rich and homogeneous programmability available at data center hosts provides a mechanism to rapidly realize new functionality. For example, the VL2 agent enables fine-grained path control by adjusting the randomization used in VLB. The agent also replaces Ethernet’s ARP functionality with queries to the VL2 directory system. The directory system itself is also realized on regular servers, rather than switches, and thus offers flexibility, such as fine-grained access control between application servers.

4.1. Scale-out topologies

As described in Sections 2 and 3, conventional hierarchical data center topologies have poor bisection bandwidth and are susceptible to major disruptions due to device failures. Rather than *scale up* individual network devices with more capacity and features, we *scale out* the devices—building a broad network offering huge *aggregate* capacity using a large number of simple, inexpensive devices, as shown in Figure 4. This is an example of a folded Clos network⁶ where the links between the intermediate switches and the aggregation switches form a complete bipartite graph. As in the conventional topology, ToRs connect to two aggregation switches, but the large number of paths between any two aggregation switches means that if there are n intermediate switches, the failure of any one of them reduces the bisection bandwidth by only $1/n$ —a desirable property we call *graceful degradation*

Figure 4. An example Clos network between aggregation and intermediate switches provides a richly connected backbone well suited for VLB. The network is built with two separate address families—topologically significant locator-specific addresses (LAs) and flat application-specific addresses (AAs).



of bandwidth. Further, it is easy and inexpensive to build a Clos network for which there is no oversubscription (further discussion on cost is given in Section 6). For example, in Figure 4, we use D_A -port Aggregation and D_I -port intermediate switches, and connect these switches such that the capacity between each layer is $D_I D_A / 2$ times the link capacity.

The Clos topology is exceptionally well suited for VLB in that by forwarding traffic through an intermediate switch that is chosen in a destination-independent fashion (e.g., randomly chosen), the network can provide bandwidth guarantees for any traffic matrices that obey the hose model.⁸ Meanwhile, routing remains simple and resilient on this topology—take a random path up to a random intermediate switch and a random path down to a destination ToR switch.

4.2. VL2 addressing and routing

This section explains the motion of packets in a VL2 network, and how the topology, routing design, VL2 agent, and directory system combine to virtualize the underlying network fabric and create the illusion that hosts are connected to a big, noninterfering data center-wide layer-2 switch.

Address Resolution and Packet Forwarding: VL2 uses two separate classes of IP-address illustrated in Figure 4. The network infrastructure operates using LAs; all switches and interfaces are assigned LAs, and switches run an IP-based (layer-3) link-state routing protocol that disseminates only these LAs. This allows switches to obtain complete knowledge about the switch-level topology, as well as forward any packets encapsulated with LAs along the shortest paths. On the other hand, applications use permanent AAs, which remain unaltered no matter how servers' locations change due to VM migration or reprovisioning. Each AA (server) is associated with an LA, the IP address of the ToR switch to which the application server is connected. The ToR switch need not be physical hardware—it could be a virtual switch or hypervisor implemented in software on the server itself!

The VL2 directory system stores the mapping of AAs to LAs, and this mapping is created when application servers are provisioned to a service and assigned AA addresses. Resolving these mappings through a unicast-based custom protocol eliminates the ARP and DHCP scaling bottlenecks that plague large Ethernets.

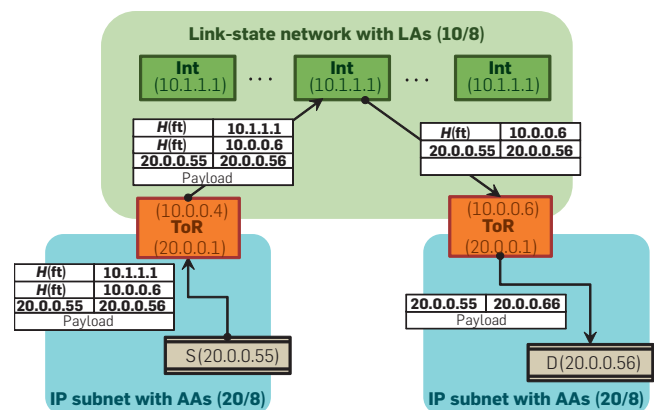
Packet forwarding: Since AA addresses are not announced into the routing protocols of the network, for a server to receive a packet the sending server must first encapsulate the packet (Figure 5), setting the destination of the outer header to the LA of the destination AA. Once the packet arrives at the LA (the destination ToR or hypervisor), the switch (the ToR or the VM switch in the destination hypervisor) decapsulates the packet and delivers it to the destination AA given in the inner header.

Address resolution and access control: Servers in each service are configured to believe that they all belong to the same IP subnet, so when an application sends a packet to an AA for the first time, the networking stack on the host generates a broadcast ARP request for the destination AA. The VL2 agent running in the source's networking stack intercepts the ARP request and converts it to a unicast query to the VL2 directory system. The directory system answers the query with the LA of the ToR to which packets should be tunneled. During this resolution process, the directory server can additionally evaluate the access-control policy between the source and destination and selectively reply to the resolution query, enforcing necessary isolation policies between applications.

These addressing and forwarding mechanisms were chosen for two main reasons. First, they make it possible to use low-cost switches, which often have small routing tables (typically just 16K entries) that can hold only LA routes, without concern for the huge number of AAs. Second, they allow the control plane to support agility with very little overhead; the design obviates frequent link-state advertisements to disseminate host-state changes and host/switch reconfiguration.

Random Traffic Spreading over Multiple Paths: To offer hot

Figure 5: VLB in an example VL2 network. Sender s sends packets to destination D via a randomly chosen intermediate switch using IP-in-IP encapsulation. AAs are from 20/8, and LAs are from 10/8. $H(\epsilon t)$ denotes a hash of the five tuple.



spot-free performance for arbitrary traffic matrices, VL2 uses VLB as its traffic engineering philosophy. As illustrated in Figure 5, VL2 achieves VLB using a combination of ECMP routing implemented by the switches and packet encapsulation implemented by the shim on each server. ECMP, a mechanism already implemented in the hardware of most switches, will distribute flows across the available paths in the network, with the packets with the same source and destination address taking the same path to avoid packet reordering. To leverage all the available paths in the network and overcome some limitations in ECMP, the VL2 agent on each sender encapsulates each packet to an intermediate switch. Hence, the packet is first delivered to one of the intermediate switches, decapsulated by the switch, delivered to the ToR's LA, decapsulated again, and finally sent to the destination server. The source address in the outer headers of the encapsulated packet is set to a hash of the inner packet's addresses and ports—this provides additional entropy to better distribute flows between the same servers across the available paths.

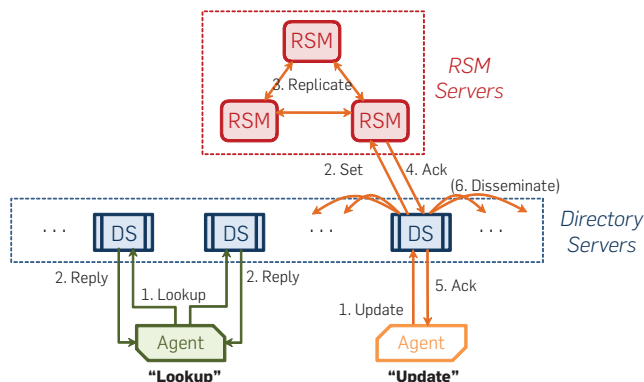
One potential issue for both ECMP and VLB is the chance that uneven flow sizes and random spreading decisions will cause transient congestion on some links. Our evaluation did not find this to be a problem on data center workloads (Section 5), but should it occur, the VL2 agent on the sender can detect and deal with it via simple mechanisms. For example, it can change the hash used to create the source address periodically or whenever TCP detects a severe congestion event (e.g., a full window loss) or an Explicit Congestion Notification.

4.3. Maintaining host information using the VL2 directory system

The VL2 directory system provides two key functions: (1) *lookups* and *updates* for AA-to-LA mappings and (2) a reactive cache update mechanism that ensures eventual consistency of the mappings with very little update overhead (Figure 6).

We expect the lookup workload for the directory system to be frequent and bursty because servers can communicate with up to hundreds of other servers in a short time period, with each new flow generating a lookup for an AA-to-LA mapping. The bursty nature of workload implies that lookups

Figure 6. VL2 Directory System Architecture.



require high throughput and low response time to quickly establish a large number of connections. Since lookups replace ARP, their response time should match that of ARP, that is, tens of milliseconds. For updates, however, the workload is driven by server-deployment events, most of which are planned ahead by the data center management system and hence can be batched. The key requirement for updates is reliability, and response time is less critical.

Our directory service replaces ARP in a conventional L2 network, and ARP ensures eventual consistency via timeout and broadcasting. This implies that eventual consistency of AA-to-LA mappings is acceptable as long as we provide a reliable update mechanism. Nonetheless, we intend to support live VM migration in a VL2 network; our directory system should be able to correct all the stale entries without breaking any ongoing communications.

The differing performance requirements and workload patterns of lookups and updates lead us to a two-tiered directory system architecture consisting of (1) a modest number (50–100 servers for 100K servers) of read-optimized, replicated lookup servers that cache AA-to-LA mappings and that communicate with VL2 agents, and (2) a small number (5–10 servers) of write-optimized, asynchronous replicated state-machine (RSM) servers offering a strongly consistent, reliable store of AA-to-LA mappings. The lookup servers ensure low latency, high throughput, and high availability for a high lookup rate. Meanwhile, the RSM servers ensure strong consistency and durability for a modest rate of updates using the Paxos¹⁹ consensus algorithm.

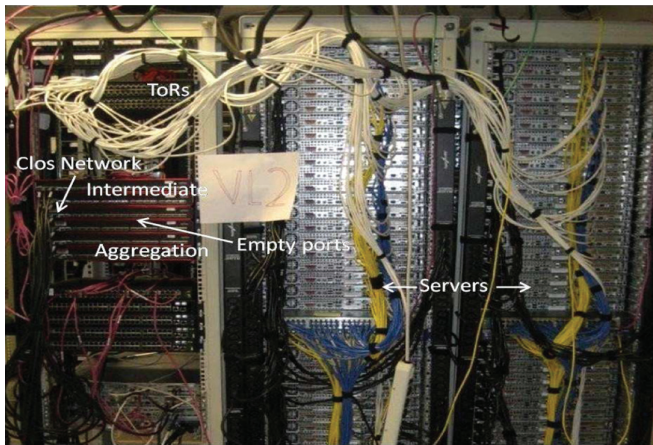
Each lookup server caches all the AA-to-LA mappings stored at the RSM servers and independently replies to lookup queries from agents using the cached state. Since strong consistency is not required, a lookup server lazily synchronizes its local mappings with the RSM every 30s. To achieve high availability and low latency, an agent sends a query to k (two in our prototype) randomly chosen lookup servers and simply chooses the fastest reply. Since AA-to-LA mappings are cached at lookup servers and at VL2 agents' cache, an update can lead to inconsistency. To resolve inconsistency, the cache-update protocol leverages a key observation: a stale host mapping needs to be corrected only when that mapping is used to deliver traffic. Specifically, when a stale mapping is used, some packets arrive at a stale LA—a ToR which does not host the destination server anymore. The ToR forwards such non-deliverable packets to a lookup server, triggering the lookup server to correct the stale mapping in the source's cache via unicast.

5. EVALUATION

In this section, we evaluate VL2 using a prototype running on an 80-server testbed and 10 commodity switches (Figure 7). Our goals are first to show that VL2 can be built from components available today, and second, that our implementation meets the objectives described in Section 1.

The testbed is built using the Clos network topology of Figure 4, consisting of three intermediate switches, three aggregation switches, and 4 ToRs. The aggregation and intermediate switches have 24 10Gbps Ethernet ports, of which 6 ports are used on each aggregation switch and 3

Figure 7. VL2 testbed comprising 80 servers and 10 switches.



ports on each intermediate switch. The ToR switches have 2 10Gbps ports and 24 1Gbps ports. Each ToR is connected to 2 aggregation switches via 10Gbps links, and to 20 servers via 1Gbps links. Internally, the switches use commodity ASICs: Broadcom ASICs 56820 and 56514, although any switch that supports line rate L3 forwarding, OSPF, ECMP, and IPinIP decapsulation will work.

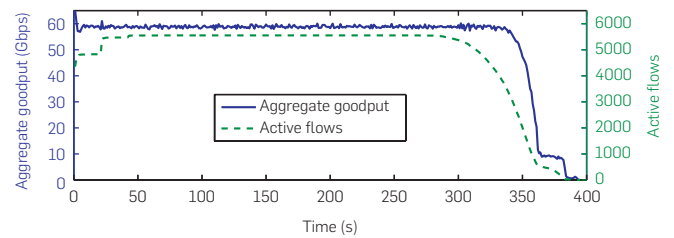
Overall, our evaluation shows that VL2 provides an effective substrate for a scalable data center network; VL2 achieves (1) 94% optimal network capacity, (2) a TCP fairness index of 0.995, (3) graceful degradation under failures with fast reconvergence, and (4) 50 K lookups/s under 10 ms for fast address resolution.

5.1. VL2 provides uniform high capacity

A central objective of VL2 is uniform high capacity between any two servers in the data center. How closely does the performance and efficiency of a VL2 network match that of a layer-2 switch with 1:1 oversubscription? To answer this question, we consider an all-to-all *data shuffle* stress test: all servers simultaneously initiate TCP transfers to all other servers. This data shuffle pattern arises in large-scale sorts, merges, and joint operations in the data center, for example, in Map/Reduce or DryadLINQ jobs.^{7, 22} Application developers use these operations with caution today, because they are so network resource expensive. If data shuffles can be supported efficiently, it would have large impact on the overall algorithmic and data storage strategy.

We create an all-to-all data shuffle traffic matrix involving 75 servers. Each of 75 servers must deliver 500MB of data to each of the 74 other servers—a shuffle of 2.7TB from memory to memory. Figure 8 shows how the sum of the goodput over all flows varies with time during a typical run of the 2.7TB data shuffle. During the run, the sustained utilization of the core links in the Clos network is about 86%, and VL2 achieves an aggregate goodput of 58.8Gbps. The goodput is very evenly divided among the flows for most of the run, with a fairness index between the flows of 0.995¹⁵ where 1.0 indicates perfect fairness (mean goodput per flow 11.4Mbps,

Figure 8. Aggregate goodput during a 2.7TB shuffle among 75 servers.



standard deviation 0.75Mbps). This goodput is more than 10x what the network in our current data centers can achieve with the same investment.

We measure how close VL2 gets to the maximum achievable throughput in this environment by computing the goodput efficiency for this data transfer. Goodput efficiency is defined as the ratio of the sent goodput summed over all interfaces divided by the sum of the interface capacities. An efficiency of 1.0 would mean that all the capacity on all the interfaces is entirely used carrying useful bytes from the time the first flow starts to when the last flow ends. The VL2 network achieves an efficiency of 94%, with the difference from perfect being due to the encapsulation headers (3.8%), TCP congestion control dynamics, and TCP retransmissions.

This 94% efficiency combined with the fairness index of 0.995 demonstrates that VL2 can achieve uniform high bandwidth across all servers in the data center.

5.2. VL2 provides performance isolation

One of the primary objectives of VL2 is *agility*, which we define as the ability to assign any server, anywhere in the data center to any service (Section 1). Achieving agility critically depends on providing sufficient performance isolation between services so that if one service comes under attack or a bug causes it to spray packets, it does not adversely impact the performance of other services.

Performance isolation in VL2 rests on the mathematics of VLB—that any traffic matrix that obeys the hose model is routed by splitting to intermediate nodes in equal ratios (through randomization) to prevent any persistent hot spots. Rather than have VL2 perform admission control or rate shaping to ensure the traffic offered to the network conforms to the hose model, we instead rely on TCP to ensure that each flow offered to the network is rate limited to its fair share of its bottleneck.

A key question we need to validate for performance isolation is whether TCP reacts sufficiently quickly to control the offered rate of flows within services. TCP works with packets and adjusts their sending rate at the time scale of RTTs. Conformance to the hose model, however, requires instantaneous feedback to avoid oversubscription of traffic ingress/egress bounds. Our next set of experiments shows that TCP is “fast enough” to enforce the hose model for traffic in each service so as to provide the desired performance isolation across services.

In this experiment, we add two services to the network. The first service has a steady network workload, while the workload of the second service ramps up and down. Both the services' servers are intermingled among the 4 ToRs, so their traffic mixes at every level of the network. Figure 9 shows the aggregate goodput of both services as a function of time. As seen in the figure, there is no perceptible change to the aggregate goodput of service one as the flows in service two start up or complete, demonstrating performance isolation when the traffic consists of large long-lived flows. In Figure 10, we perform a similar experiment, but service two sends bursts of small TCP connections, each burst containing progressively more connections. These two experiments demonstrate TCP's enforcement of the hose model sufficient to provide performance isolation across services at timescales greater than a few RTT (i.e., 1–10ms in data centers).

5.3. VL2 directory system performance

Finally, we evaluate the performance of the VL2 directory system which provides the equivalent semantics of ARP in layer 2. We perform this evaluation through macro- and micro-benchmark experiments on the directory system. We run our prototype on up to 50 machines: 3–5 RSM nodes, 3–7 directory server nodes, and the remaining nodes emulating multiple instances of VL2 agents generating lookups and updates.

Our evaluation supports four main conclusions. First, the directory system provides high throughput and fast response time for lookups: three directory servers can

Figure 9: Aggregate goodput of two services with servers intermingled on the ToRs. Service one's goodput is unaffected as service two ramps traffic up and down.

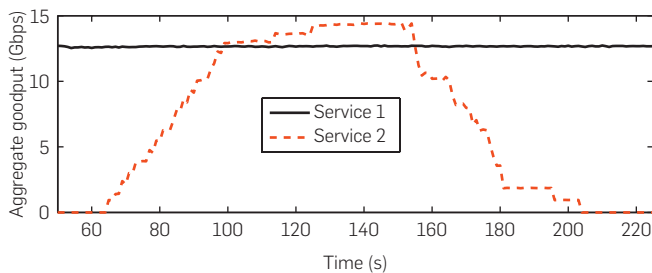
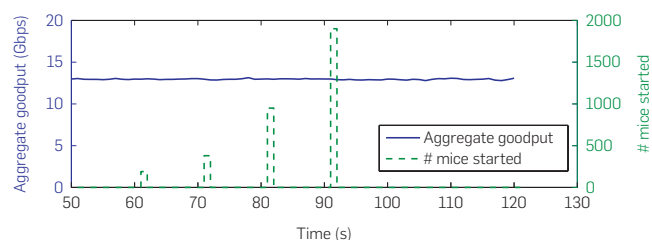


Figure 10. Aggregate goodput of service one as service two creates bursts containing successively more short TCP connections.



handle 50K lookup/second with latency under 10ms (99th percentile latency). Second, the directory system can handle updates at rates significantly higher than the expected churn rate in typical environments: three directory servers can handle 12K updates/s within 600ms (99th percentile latency). Third, our system is incrementally scalable: each directory server increases the processing rate by about 17K for lookups and 4K for updates. Finally, the directory system is robust to component (directory or RSM servers) failures and offers high availability under network churn.

To understand the incremental scalability of the directory system, we measured the maximum lookup rates (ensuring sub-10ms latency for 99% requests) with 3, 5, and 7 directory servers. The result confirmed that the maximum lookup rates increases linearly with the number of directory servers (with each server offering a capacity of 17K lookups/s). Based on this result, we estimate the worst case number of directory servers needed for a 100K server data center. Using the concurrent flow measurements (Figure 3), we use the median of 10 correspondents per server in a 100s window. In the worst case, all 100K servers may perform 10 simultaneous lookups at the same time resulting in a million simultaneous lookups per second. As noted above, each directory server can handle about 17K lookups/s under 10ms at the 99th percentile. Therefore, handling this worst case will require a directory system of about 60 servers (0.06% of the entire servers).

6. DISCUSSION

In this section, we address several remaining concerns about the VL2 architecture, including whether other traffic engineering mechanisms might be better suited to the DC than VLB, and the cost of a VL2 network.

Optimality of VLB: As noted in Section 4.2.2, VLB uses randomization to cope with volatility, potentially sacrificing some performance for a best-case traffic pattern by turning all traffic patterns (including both best-case and worst-case) into the average case. This performance loss will manifest itself as the utilization of some links being higher than they would under a more optimal traffic engineering system. To quantify the increase in link utilization VLB will suffer, we compare VLB's maximum link utilization with that achieved by other routing strategies on the VL2 topology for a full day's traffic matrices (TMs) (at 5 min intervals) from the data center traffic data reported in Section 3.

We first compare to *adaptive routing*, which routes each TM separately so as to minimize the maximum link utilization for that TM—essentially upper-bounding the best performance that real-time adaptive traffic engineering could achieve. Second, we compare to *best oblivious routing* over all TMs so as to minimize the maximum link utilization. (Note that VLB is just one among many oblivious routing strategies.) For adaptive and best oblivious routing, the routings are computed using respective linear programs in `cplex`. The overall utilization for a link in all schemes is computed as the maximum utilization over all routed TMs.

In Figure 11, we plot the CDF for link utilizations for the three schemes. We normalized the link utilization numbers so that the maximum utilization on any link for

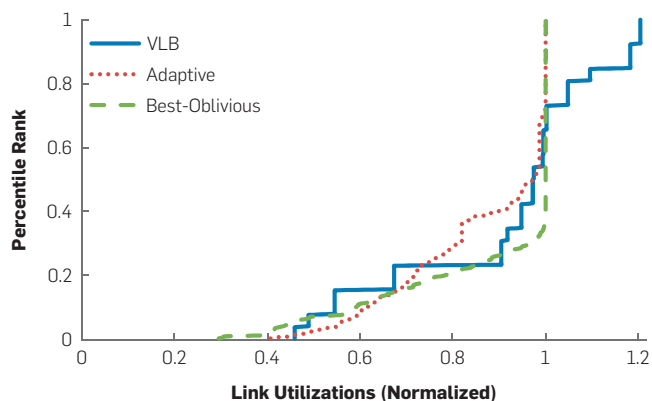
adaptive routing is 1.0. The results show that for most links, VLB performs about the same as the other two schemes. For the most heavily loaded link in each scheme, VLB's link capacity usage is at worst 20% higher than that of the other two schemes. Thus, evaluations on actual data center workloads show that the simplicity and universality of VLB costs relatively little capacity when compared to much more complex traffic engineering schemes. Moreover, adaptive routing schemes might be difficult to implement in the data center. Since even the “elephant” flows are about 100MB (see Figure 2), lasting about 1 s on a server with a 1Gbps NIC, any reactive traffic engineering scheme will need to run at least as frequently if it wants to react to individual flows.

Cost and Scale: With the range of low-cost commodity devices currently available, the VL2 topology can scale to create networks with no oversubscription between all the servers of even the largest data centers. For example, switches with 144 ports ($D = 144$) are available today for \$150K, enabling a network that connects 100K servers using the topology in Figure 4 and up to 200K servers using a slight variation. Using switches with $D = 24$ ports (which are available today for \$8K each), we can connect about 3K servers. Comparing the cost of a VL2 network for 35K servers with a conventional one found in one of our data centers shows that a VL2 network with no oversubscription can be built for the same cost as the current network that has 1:240 oversubscription. Building a conventional network with no oversubscription would cost roughly 14× the cost of a equivalent VL2 network with no oversubscription.

7. RELATED WORK

Data center network designs: There is great interest in building data center networks using commodity switches and a Clos topology.^{2, 11, 20, 21} The designs differ in whether they provide layer-2 semantics, their traffic engineering strategy, the maturity of their control planes, and their compatibility with existing switches. Other approaches use the servers themselves for switching data packets.^{1, 12,}

Figure 11. CDF of normalized link utilizations for VLB, adaptive, and best oblivious routing schemes, showing that VLB (and best oblivious routing) come close to matching the link utilization performance of adaptive routing.



¹³ VL2 also leverages the programmability of servers; however, it uses servers only to control the way traffic is routed as switch ASICs forward packets at less cost in power and dollars per Mbps.

Valiant load balancing: Valiant introduced VLB as a randomized scheme for communication among parallel processors interconnected in a hypercube topology.⁶ Among its recent applications, VLB has been used inside the switching fabric of a packet switch.³ VLB has also been proposed, with modifications and generalizations,^{18, 23} for oblivious routing of variable traffic on the Internet under the hose traffic model.⁸

Scalable routing: The Locator/ID Separation Protocol⁹ proposes “map-and-encap” as a key principle to achieve scalability and mobility in Internet routing. VL2's control plane takes a similar approach (i.e., demand-driven host-information resolution and caching) but adapted to the data center environment and implemented on end hosts. SEATTLE¹⁷ proposes a distributed host-information resolution system running on switches to enhance Ethernet's scalability.

Commercial networks: Data Center Ethernet (DCE)⁴ by Cisco and other switch manufacturers shares VL2's goal of increasing network capacity through multipath. These industry efforts are primarily focused on consolidation of IP and storage area network (SAN) traffic, and there are few SANs in cloud-service data centers. Due to the requirement to support loss-less traffic, their switches need much bigger buffers (tens of MBs) than commodity Ethernet switches do (tens of KBs), hence driving their cost higher.

8. SUMMARY

VL2 is a new network architecture that puts an end to the need for oversubscription in the data center network, a result that would be prohibitively expensive with the existing architecture.

VL2 benefits the cloud service programmer. Today, programmers have to be aware of network bandwidth constraints and constrain server-to-server communications accordingly. VL2 instead provides programmers the simpler abstraction that all servers assigned to them are plugged into a single layer-2 switch, with hot spot-free performance regardless of where the servers are actually connected in the topology. VL2 also benefits the data center operator as today's bandwidth and control plane constraints fragment the server pool, leaving servers (which account for the lion's share of data center cost) underutilized even while demand elsewhere in the data center is unmet. Instead, VL2 enables agility: any service can be assigned to any server, while the network maintains uniform high bandwidth and performance isolation between services.

VL2 is a simple design that can be realized today with available networking technologies, and without changes to switch control and data plane capabilities. The key enablers are an addition to the end-system networking stack, through well-established and public APIs, and a flat addressing scheme, supported by a directory service.

VL2 is efficient. Our working prototype, built using commodity switches, approaches in practice the high level

of performance that the theory predicts. Experiments with two data center services showed that churn (e.g., dynamic reprovisioning of servers, change of link capacity, and microbursts of flows) has little impact on TCP goodput. VL2's implementation of VLB splits flows evenly and VL2 achieves high TCP fairness. On all-to-all data shuffle communications, the prototype achieves an efficiency of 94% with a TCP fairness index of 0.995.

Acknowledgments

Insightful comments from David Andersen, Jon Crowcroft, and the anonymous reviewers greatly improved the final version of this paper. C

References

1. Abu-Libdeh, H., Costa, P., Rowstron, A., O'Shea, G., Donnelly, A. Symbiotic routing in future data centers. In *SIGCOMM* (2010).
2. Al-Fares, M., Loukissas, A., Vahdat, A. A scalable, commodity data center network architecture. In *SIGCOMM* (2008).
3. Chang, C., Lee, D., Jou, Y. Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering. *IEEE HPSR* (2001).
4. Cisco. Data center Ethernet. <http://www.cisco.com/go/dce>.
5. Cisco. Data center: Load balancing data center services, 2004.
6. Dally, W.J., Towles, B. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, 2004.
7. Dean, J., Ghemawat, S. MapReduce: simplified data processing on large clusters. In *OSDI* (2004).
8. Duffield, N.G., Goyal, P., Greenberg, A.G., Mishra, P.P., Ramakrishnan, K.K., van der Merwe, J.E. A flexible model for resource management in virtual private network. In *SIGCOMM* (1999).
9. Farinacci, D., Fuller, V., Oran, D., Meyer, D., Brimm, S. Locator/ID Separation Protocol (LISP). Internet-draft, Dec. 2008.
10. Greenberg, A., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D., Patel, P., Sengupta, S. VL2: A scalable and flexible data center network. In *SIGCOMM* (2009).
11. Greenberg, A., Lahiri, P., Maltz, D.A., Patel, P., Sengupta, S. Towards a next

generation data center architecture: Scalability and commoditization. In *PRESTO Workshop at SIGCOMM* (2008).

12. Guo, C., Wu, H., Tan, K., Shiy, L., Zhang, Y., Lu, S. DCell: a scalable and fault-tolerant network structure for data centers. In *SIGCOMM* (2008).
13. Guo, C., Wu, H., Tan, K., Shiy, L., Zhang, Y., Lu, S. BCube: a high performance, server-centric network architecture for modular data centers. In *SIGCOMM* (2009).
14. Hamilton, J. Cems: Low-cost, low-power servers for internet-scale services. In *Conference on Innovative Data Systems Research* (Jan 2009).
15. Jain, R. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, Inc., 1991.
16. Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R. The nature of datacenter traffic: Measurements and analysis. In *IMC* (2009).
17. Kim, C., Caesar, M., Rexford, J. Floodless in SEATTLE: a scalable ethernet architecture for large enterprises. In *SIGCOMM* (2008).
18. Kodialam, M., Lakshman, T.V., Sengupta, S. Efficient and robust routing of highly variable traffic. In *HotNets* (2004).
19. Lamport, L. The part-time parliament. *ACM Trans. Comput. Syst.* 16 (1998), 133-169.
20. Mudigonda, J., Yalagandula, P., Al-Fares, M., Mogul, J.C. Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In *NSDI* (2010).
21. Touch, J., Perlman, R. Transparent interconnection of lots of links (TRILL): Problem and applicability statement. IETF RFC 5556 (2009).
22. Yu, Y., Isard, M., Fetterly, D., Budiu, M., Erlingsson, U., Gunda, P.K., Currey, J. DryadLINQ: a system for general-purpose distributed data-parallel computing using a high-level language. In *OSDI* (2008).
23. Zhang-Shen, R., McKeown, N. Designing a Predictable Internet Backbone Network. In *HotNets* (2004).

Albert Greenberg, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, Sudipta Sengupta, Microsoft Research

James R. Hamilton, Amazon Web Services

© 2011 ACM 0001-0782/11/0300 \$10.00



Association for Computing Machinery

Advancing Computing as a Science & Profession



MentorNet®

You've come a long way.
Share what you've learned.



ACM has partnered with MentorNet, the award-winning nonprofit e-mentoring network in engineering, science and mathematics. MentorNet's award-winning **One-on-One Mentoring Programs** pair ACM student members with mentors from industry, government, higher education, and other sectors.

- Communicate by email about career goals, course work, and many other topics.
- Spend just **20 minutes a week** - and make a huge difference in a student's life.
- Take part in a lively online community of professionals and students all over the world.



Make a difference to a student in your field.
Sign up today at: www.mentornet.net
Find out more at: www.acm.org/mentornet

MentorNet's sponsors include 3M Foundation, ACM, Alcoa Foundation, Agilent Technologies, Amylin Pharmaceuticals, Bechtel Group Foundation, Cisco Systems, Hewlett-Packard Company, IBM Corporation, Intel Foundation, Lockheed Martin Space Systems, National Science Foundation, Naval Research Laboratory, NVIDIA, Sandia National Laboratories, Schlumberger, S.D. Bechtel, Jr. Foundation, Texas Instruments, and The Henry Luce Foundation.