Towards a Next Generation Data Center Architecture: Scalability and Commoditization

Albert Greenberg, Parantap Lahiri, David A. Maltz, Parveen Patel, Sudipta Sengupta
Microsoft Research, Redmond, WA, USA

Abstract

Applications hosted in today's data centers suffer from internal fragmentation of resources, rigidity, and bandwidth constraints imposed by the architecture of the network connecting the data center's servers. Conventional architectures statically map web services to Ethernet VLANs, each constrained in size to a few hundred servers owing to control plane overheads. The IP routers used to span traffic across VLANs and the load balancers used to spray requests within a VLAN across servers are realized via expensive customized hardware and proprietary software. Bisection bandwidth is low, severly constraining distributed computation. Further, the conventional architecture concentrates traffic in a few pieces of hardware that must be frequently upgraded and replaced to keep pace with demand - an approach that directly contradicts the prevailing philosophy in the rest of the data center, which is to *scale out* (adding more cheap components) rather than *scale up* (adding more power and complexity to a small number of expensive components).

Commodity switching hardware is now becoming available with programmable control interfaces and with very high port speeds at very low port cost, making this the right time to redesign the data center networking infrastructure. In this paper, we describe Monsoon, a new network architecture, which scales and commoditizes data center networking. Monsoon realizes a simple mesh-like architecture using programmable commodity layer-2 switches and servers. In order to scale to 100,000 servers or more, Monsoon makes modifications to the control plane (e.g., source routing) and to the data plane (e.g., hot-spot free multipath routing via Valiant Load Balancing). It disaggregates the function of load balancing into a group of regular servers, with the result that load balancing server hardware can be distributed amongst racks in the data center leading to greater agility and less fragmentation. The architecture creates a huge, flexible switching domain, supporting any server/any service and unfragmented server capacity at low cost.

Categories and Subject Descriptors: C.2.1 [Computer-Communication Network]: Network Architecture and Design

General Terms: Design, Performance, Reliability

Keywords: Data center network, commoditization

1. INTRODUCTION

Data centers are comprised of both server and networking infrastructure. The server portion of the infrastructure is now far down the road of commoditization - high-end enterprise-class servers have been replaced by large numbers of low-cost servers. Innovation in distributed computing and systems management software have enabled the unreliability of individual servers to be masked by the aggregated reliability of the system as a whole. The running theme is "scaling out instead of scaling up," driven by the economics of PC commoditization.

The network portion of the data center infrastructure presents the next frontier for commoditization. Data centers now being built contain upwards of 100,000 servers, and the increase in the number of servers that need to be interconnected has stretched the limits of enterprise networking solutions so much that current architectures resemble a spectrum of patches and workarounds for protocols that were originally intended for use in enterprise networks orders of magnitude smaller. In our analysis of budget requests from multiple data centers owned by a large corporation, the networking gear was consuming 10–20% of the data center's equipment budget.

We believe that the confluence of two trends means that the time is right to redesign the data center networking infrastructure. The first trend is the availability of commodity switching hardware with 1 Gbps ports below $100 and 10 Gbps ports falling below $1,000 — these switches sometimes lack the sophisticated packet processing and large buffers and TCAMs found in their more expensive couterparts, but they offer basic data-plane operations at high speeds. The second trend is the ability to replace the control plane protocols used by these switches — this new programmability enables the switches' data-plane features to be applied in novel ways to the problems of flexibly interconnecting data center servers.

This paper presents *Monsoon*, a blueprint for commoditizing the networks of data centers used for "cloud" services where large numbers of servers cooperatively handle huge workloads (e.g., indexing the web). Monsoon design drives down the cost of the network infrastructure while simultaneously increasing its ability to support the bandwidth and functionality requirements of data centers. Our insights and contributions can be summarized as:

Traffic Engineering on Layer 2 Mesh: Data center networks today suffer from sporadic congestion as workload inside the data center changes frequently. The aggregate computational power of the data center is significantly reduced due to network bottlenecks. Monsoon uses Valiant Load Balancing (VLB) on a mesh of commodity Ethernet switches to realize a hot-spot free core fabric that supports arbitrary traffic patterns in an *oblivious manner* (i.e., it accommodates any traffic matrix permitted by the server interface cards). We leverage the programmability of the switches and the

PRESTO'08, August 22, 2008, Seattle, Washington, USA.
Copyright 2008 ACM 978-1-60558-181-1/08/08 ...$5.00.

servers so that VLB can be implemented using only the data-plane features available on commodity switches.

Scale to Huge Layer 2 Domain: Data centers today suffer from internal fragmentation of resources, where network infrastructure and IP addressing hierarchies limit the ability to dynamically reassign servers among the applications running in the data center. Using a new combination of existing techniques, Monsoon creates a single layer 2 domain that forwards frames using flat addresses but still scales up to 100,000 servers.

Ubiquitous Load Spreading: The ability to spread requests for a service across a pool of servers is a critical building block in any scale-out data center. Today, this spreading is implemented by a small number of specialized hardware load balancers that are expensive, difficult to configure, and frequently responsible for outages. Monsoon load balances traffic to any service across a pool of servers of arbitrary size. We introduce *MAC Rotation*, the underlying mechanism used for load spreading in Monsoon. This mechanism also enables disaggregation of application-aware load balancing to regular servers placed anywhere in the data center.

2. CHALLENGES AND REQUIREMENTS

Multiple applications run inside a single data center, typically with each application is hosted on its own set of (potentially virtual) server machines. Each application is associated with one or more publicly visible and routable IP addresses to which clients in the Internet send their requests and from which they receive replies. Inside the data center, requests are spread among a pool of front-end servers that process the requests. This spreading is typically performed by a specialized load balancer [9]. Using conventional load-balancer terminology, the IP address to which requests are sent is called a virtual IP address (VIP) and the IP addresses of the servers over which the requests are spread are known as direct IP addresses (DIPs).

Figure 1 shows the conventional architecture for a data center, taken from a recommended source [2]. Requests arriving from the Internet are IP (layer 3) routed through border and access routers to a layer 2 domain based on the destination VIP address. The VIP is configured onto the two load balancers connected to the top switches, and complex mechanisms are used to ensure that if one load balancer fails, the other picks up the traffic [13]. For each VIP, the load balancers are configured with a list of DIPs, which are the private and internal addresses of physical servers in the racks below the load balancers. This list of DIPs defines the pool of servers that can handle requests to that VIP, and the load balancer spreads requests across the DIPs in the pool.

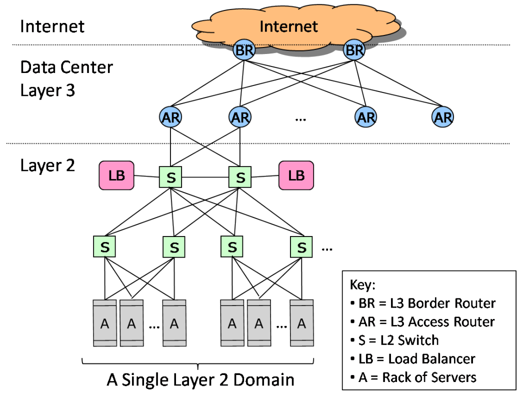

Figure 1: The conventional network architecture for data centers (adapted from figure by Cisco [2]).

As shown in the figure, all the servers that connect into a pair of access routers comprise a single layer 2 domain. With conventional network architectures and protocols, a single layer 2 domain is limited in size to about 4,000 servers in practice, driven by the need for rapid reconvergence upon failure. Since the overhead of broadcast traffic (e.g., ARP) limits the size of an IP subnet to a few hundred servers, the layer 2 domain is divided up into subnets using VLANs configured on the Layer 2 switches, one subnet per VLAN.

The conventional approach has the following problems:

Fragmentation of resources: Popular load balancing techniques, such as *destination NAT* (or *half-NAT*) and *direct server return*, require that all DIPs in a VIP's pool be in the same layer 2 domain [9]. This constraint means that if an application grows and requires more servers, it cannot use available servers in other layer 2 domains — ultimately resulting in fragmentation and underutilization of resources. Load balancing via *Source NAT* (or *full-NAT*) does allow servers to be spread across layer 2 domains, but then the servers never see the client IP, which is often unacceptable because servers use the client IP for everything from data mining and response customization to regulatory compliance.

Poor server to server connectivity: The hierarchical nature of the network means that for servers in different layer 2 domains to communicate, traffic must go through the layer 3 portion of the network. Layer 3 ports are significantly more expensive than layer 2 ports, owing in part to the cost of supporting large buffers, and in part to market place factors. As a result, these links are typically oversubscribed (i.e., the capacity of the links between access routers and border routers is significantly less than the sum of the output capacity of the servers connected to the access routers). The result is that the bandwidth available between servers in different parts of the data center can be quite limited. This creates a serious global optimization problem as all servers belonging to all applications must be placed with great care to ensure the sum of their traffic does not saturate any of the network links. Achieving this level of coordination between applications is untenable in practice.

Proprietary hardware that scales up, not out: The load balancers in the conventional architecture are used in pairs in a 1+1 resiliency configuration. When the load becomes too great for the load balancers, operators replace the existing load balancers with a new pair having more capacity, which is an unscalable and expensive strategy.

In contrast with the conventional architecture, we want an architecture that meets the following challenges:

Placement anywhere: The architecture should allow any server anywhere in the data center to be a part of the pool of servers behind any VIP, so that server pools can be dynamically shrunk or expanded.

Server to server bandwidth: In many data center applications, the amount of traffic between servers inside the data center dwarfs the amount of traffic exchanged with clients outside the data center (e.g,. many applications require extensive computation spread over many servers in order to generate a response in a short period of time). This implies the architecture should provide as much bandwidth as possible between every pair of servers in the data center, regardless of where they are located.

Commodity hardware that scales out: As more capacity is needed, it should be easier to add more individual components than replace existing components with newer, higher capacity models. This requires that the architecture supports a n+1 resiliency model.

Support 100,000 servers: Large data centers today house 100,000 or more servers, though the network stands in the way of harnessing these servers in arbitrary pools. Large data centers of the future should support intensive internal communications between all servers.

As explained in the next section, the Monsoon design meets these challenges by leveraging the programmability of servers, switches with programmable control planes, and certain useful data-plane primitives implemented in the switch hardware.

3. ARCHITECTURE

Figure 2 provides an overview of the Monsoon architecture. The two key features are the large layer 2 network that connects together all 100,000 servers inside the data center and the flexible ways in which requests can be distributed over pools of servers. We first summarize the architecture, and then the subsections that follow go into more detail on each point.

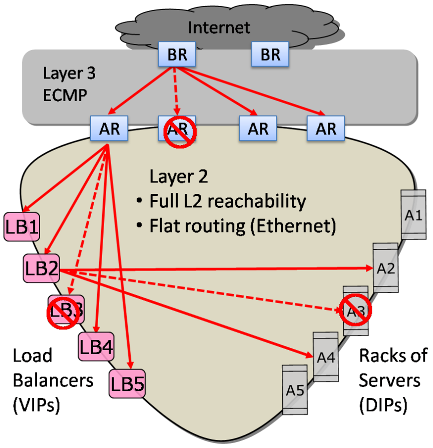

Figure 2: Overview of the Monsoon architecture. All servers connected by a layer 2 network with no oversubscribed links.

From a high-level architectural perspective, the differences between layer 2 (Ethernet) and layer 3 (IP) are shrinking, especially for a network contained inside a single building, like a data center. However, there are a number of practical factors that drove Monsoon to connect all the servers via a single large domain at layer 2. First is cost: we want to reduce network cost as much as possible. Second is the need to eliminate fragmentation of servers: with hierarchical addresses, the servers used to expand an application need to be located in the same network hierarchy as the application. With a huge flat address space, one pool of free servers can be shared by all applications. Third is the principle of least disturbance: existing applications, management systems, and security policies make extensive use of IP addresses (e.g., VIP/DIP mappings on load balancers and security policies on application servers). By implementing Monsoon's features underneath IP, these systems can continue to operate largely unmodified. Combining these three factors makes Ethernet the clear winner: it runs under IP, and it is already cost and performance optimized for forwarding based on flat addresses. An Ethernet port is roughly 10% to 50% the cost of a equivalent speed layer-3 port, and data centers contain hundreds of thousands of ports.

As shown in Figure 2, all servers connect to the layer 2 network that is designed to have full reachability with no oversubscribed links, meaning that any server can communicate with any other server at the full 1 Gbps rate of the servers' network interfaces. The layer 3 portion of the Monsoon network connects the data center to the Internet, and it uses Equal Cost MultiPath (ECMP) to spread the requests received from the Internet equally over all access routers. As the requests enter the layer 2 domain, the access routers use consistent hashing to spread the requests going to each application's public VIP equally over the set of servers acting as load balancers for that application. Finally, the load balancers spread the requests using an application-specific load distribution function over the pool of servers, identified by their DIPs, that implement the application functionality.

Monsoon's ability to spread the packets destined to an IP address over a set of servers means that load balancers can be disaggregated — that is, built from commodity servers instead of specialized high-throughput hardware. This both saves cost and adds flexibility. Today, load balancers need hardware-assisted forwarding to keep up with the high-data rate. With the network providing the ability to spread requests across servers, the load on each server can be kept at the level where forwarding can be done in software. As the offered load begins to overwhelm the existing load balancers, additional servers can be provisioned as load balancers to dilute the load. This costs substantially less than the equivalent hardware load balancer. Additionally, using commodity servers as load balancers enables them to be fully programmable, with their algorithms tunable to particular data center applications rather than making do with the algorithms vendors provide in firmware.

As indicated in the figure, the architecture provides recovery against failure of any access router, load balancer, or server in the data center. A health service continually monitors the liveness of each server, and when a problem is detected, that server is taken out of the rotation pool and new requests are no longer sent to it. (Both control and management plane realizations are possible; see, for example, [6]).

3.1 Available Components

Layer 2 switches lack many of the packet handling features of layer 3 routers. However, Monsoon shows how devices having a layer 2 forwarding plane implemented in hardware, but programmable control plane software, are more than sufficient to implement an elegant and performant design. We assume Ethernet switches with the following hardware capabilities:

MAC-in-MAC tunneling: IEEE 802.1ah [5] defines a layer 2 analogue to IP-in-IP encapsulation. When a switch receives a frame sent to its own MAC address, it removes the outermost MAC header, and, if there is another MAC header inside, forwards the frame towards the MAC destination address in the inner header. Today 802.1ah is implemented in "carrier Ethernet" switches, but we expect it to become widely available. As shown in the sections that follow, having switches that implement this primitive is tremendously valuable, as it allows software on the end-servers to orchestrate sophisticated network behavior.

16K MAC entries: While a few switches support over 100K MAC entries in their forwarding tables, the vast majority of switches support only 16,000. Since we target a data center with 100K servers, it is clear that not every switch can hold a route to each server.

Node degree: We make use of two types of switches. First, a top-of-rack (TOR) switch that aggregates the 20 1-Gbps links coming from the 20 servers in each rack onto 2 10-Gbps uplinks. Second, a "core" switch with 144 ports of 10-Gbps. Switches in these classes are available from vendors such as Woven Systems, Fulcrum Microsystems, Broadcom, and Nortel.

3.2 Server-to-Server Forwarding

The ability to forward frames between servers in the data center is the most basic aspect of Monsoon, and other functions are built on top of it. Forwarding in a scale-out data center must meet three main requirements.

Forwarding scalability: Monsoon must connect 100,000 servers using a single layer 2 network built of switches that can store only 16,000 MAC forwarding entries each. Clearly, Monsoon cannot allow the switches to see the destination address of every server.

Monsoon's solution is to have the sending server encapsulate its frames in a MAC header addressed to the destination's top-of-rack switch, so that switches need only store forwarding entries for other switches and their own directly connected servers.

Traffic engineering: Without knowing the traffic patterns of the applications that will run over it, Monsoon must support any traffic matrix in which no server is asked to send or receive more traffic than the 1-Gbps its network interface allows (known as the *hose traffic model* [3]).

Monsoon's solution is to use *Valiant Load Balancing (VLB)*, an oblivious routing strategy known to handle arbitrary traffic variations that obey the hose model [8, 18]. VLB requires that every frame sent across the network first "bounce" off a randomly chosen intermediate switch before being forwarded to its destination.[1] VLB has been shown to be capacity efficient for handling traffic variation under the hose model [8]. Monsoon implements VLB by adding an additional encapsulation header to frames that directs them to a randomly chosen switch.

Load spreading: In building data center applications, it is frequently useful to spread requests across a set of servers. Monsoon's solution is to support load spreading as a part of basic forwarding, using a mechanism we call *MAC Rotation*.

In Monsoon, any IP address can be mapped to a pool of servers, each server identified by its individual MAC address. A health service keeps track of active servers in every pool with a common IP address. When a server has a frame to send to an IP address, the frame is sent to an active server in the pool. To avoid packet reordering, the MAC address is chosen on a per-flow basis, e.g., using consistent hashing on IP 5–tuple.

Taken together, these solutions enable us to build a large layer 2 mesh with the traffic oblivious properties of VLB while using switches with small, and hence cheap, MAC forwarding tables.

3.2.1 *Obtaining Path Information From Directory*

When the application running on server presents its network stack with a packet to send to an IP address, the server needs two pieces of information before it can create and send a layer 2 frame. As explained above, it must have the list of MAC addresses for the servers responsible for handling that IP address, and the MAC address of the top-of-rack switch where each of those servers is connected. It also needs a list of switch MAC addresses from which it will randomly pick a switch to "bounce" the frame off of.

Servers obtain these pieces of information from a *Directory Service* maintained by Monsoon. The means by which the directory service is populated with data is explained in Section 3.6.

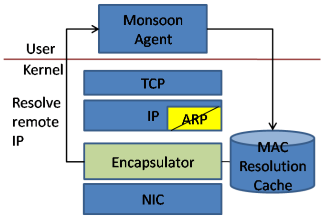

Figure 3: The networking stack of a host. The Monsoon Agent looks up remote IPs in the central directory.

Figure 3 shows the networking stack of a server in Monsoon. The traditional ARP functionality has been disabled and replaced with a user-mode process (the Monsoon Agent), and a new virtual MAC interface, called the encapsulator, has been added to encapsulate outgoing Ethernet frames. These changes are completely transparent to applications.

When the encapsulator receives a packet from the IP network stack, it computes a flow id for the packet and examines its cache of active flows for a matching entry. If there is no entry, it queues the packet and sends a request to the Monsoon Agent to look up the remote IP using the directory service.

Once the directory service returns the MAC addresses to which the IP address resolves and the set of VLB intermediate switches to use, the encapsulator chooses a destination MAC address (and its corresponding top-of-rack switch address) and a VLB intermediate node for the flow and caches this mapping. The server randomly chooses an intermediate node for every IP flow, thus spreading its load among all VLB intermediate nodes without causing TCP packet reordering.[2] If the directory service maps a remote IP address to a list of MAC addresses, servers will choose a different MAC address for each flow to the remote IP, thereby implementing load spreading.

3.2.2 *Encapsulation of Payload and Forwarding*

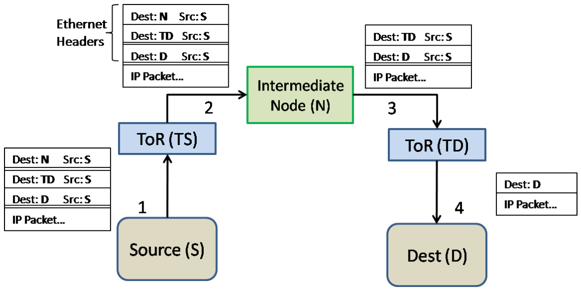

Figure 4: Frame processing when packets go from one server to another in the same data center.

With the information from the entry in the flow cache, encapsulating and forwarding a frame is straight forward. Figure 4 shows how IP packets are transmitted with three MAC headers. The outermost header has the selected intermediate node as the destination, the middle header has the target's top-of-rack switch as the destination, and the innermost header has the ultimate destination's MAC address. The sending server's top-of-rack switch forwards the frame towards the VLB intermediate node, which upon receiving the frame removes the outer header and forwards the frame to the destination's top-of-rack switch. The process repeats, with the top-of-rack switch forwarding a normal Ethernet frame with a single header towards the destination server.

3.3 External Connections

Figure 5 shows the network path for connections that originate or terminate outside the data center. External traffic enters and exits the data center through Border Routers. The Border Routers are connected to a set of Access Routers through a layer-3 Equal Cost Multi-Path (ECMP) routing configuration.

As described in Section 3.2, inside the data center traffic is routed by address resolution using the Monsoon directory service and the encapsulation of Ethernet frames at the source. It would be ideal if Access Routers had the ability to spread packets across a set

[1] The intuition behind VLB is that by randomly selecting a node in the network through which to forward traffic, the routing protocols do not need to adjust to changes in offered traffic load.

[2] Since the network provides bisection bandwidth thousands of times larger than the largest flow a server can source or sink, there should not be problem with large flows violating the VLB traffic split ratios. TCP flowlet switching [15] could be used if there is a problem.

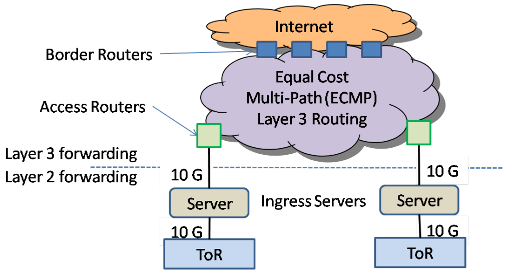

Figure 5: Network path for connections across the Internet. ECMP provides resiliency at Layer 3 for Access Router failures. Traffic is routed to nodes inside the data center with the help of Ingress Servers.

of MAC addresses. However, since routers today do not support the Monsoon load spreading primitive or the encapsulation Monsoon uses for VLB, we buddy a server, called an Ingress Server, with each Access Router. We configure the Access Router to send all the external traffic through the Ingress Server, which implements the Monsoon functionality and acts as a gateway to the data center.

Each Ingress Server has two network interfaces — one is directly connected to an Access Router and the other is connected to the data center network via a top-of-rack switch. For packets from the Internet, the Ingress Server takes packets from the Access Router, resolves internal IPs using the Monsoon directory service and forwards traffic inside the data center using encapsulated Ethernet frames like any other server. The directory service maps the IP address of the layer 2 domain's default gateway to the MAC address of the ingress servers, so packets headed to the Internet flow out through them to the access routers.

3.4 Load Balancing

Many data center applications require the ability to distribute work over a pool of servers. In some cases, the work originates from clients in the Internet, in others cases, from servers inside the data center. Monsoon provides mechanisms that support the most common types of work distribution.

Load spreading: When the objective is to have requests spread evenly over a pool of servers and the servers must see the IP address of the client as the source address of the request, the MAC rotation primitive offered by Monsoon's server-to-server forwarding is sufficient. All servers in the pool are configured with the VIP associated with the pool. The directory service will then answer a request to reach the VIP with a list of all MAC addresses in the pool, and senders use consistent hashing on flows to pick the server for each request.

Load balancing: When the objective is to place load balancers in front of the actual servers, Monsoon uses the approach shown in Figure 2. The VIP is configured onto all the load balancers, causing the ingress servers to use consistent hashing and the the load spreading method described above to spread requests evenly across the load balancers. By spreading the load across multiple load balancers in this manner, Monsoon supports a N+1 failover configuration, as opposed to the more expensive 1+1 failover configuration used by the conventional architecture.

The load balancers are free to implement any function the application desires. For example, the load balancer might implement a function that rewrites the source and destination IP addresses to spread requests over a second pool of servers using some workload- or request-sensitive logic, or it might do deep packet inspection to validate the request before sending it on to an application server.

As another example, the load balancer might terminate incoming TCP connections and decrypt SSL connections contained in them, thereby offloading work from the application servers. This example also illustrates why load spreading must use consistent hashing on flows — ECMP might direct packets that are part of the same TCP connection to different access routers and ingress servers. Yet, it is critical that all those packets are sent to the same load balancer, and consistent hashing ensures that happens.

3.5 Switch Topology

There are many physical topologies by which the switches making up the Monsoon layer 2 domain might be connected, but this section provides one concrete example of a topology that interconnects ≈100,000 servers and is particularly well suited for Valiant Load Balancing.

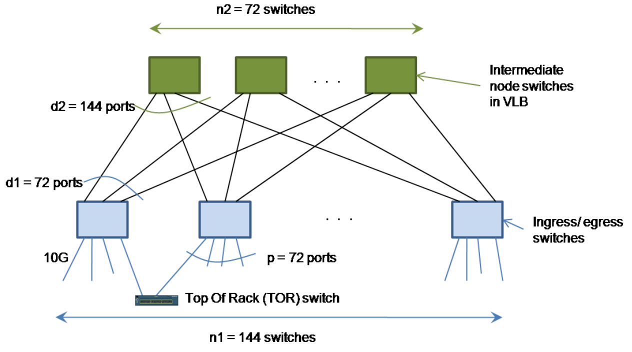

Figure 6: Example topology for layer 2 switches connecting 103,680 servers. Uses Valiant Load Balancing to support any feasible traffic matrix.

As shown in Figure 6, each top-of-rack switch has 2 10-Gbps ports on the network side that connect to two different core ingress-egress switches for fault-tolerance. There are $n_1 = 144$ such ingress-egress switches, shown in light gray. These ingress-egress switches have no links between them, but each of them connects, through a 10-Gbps port, to every intermediate switch, of which there are $n_2 = 72$, shown in dark gray. This topology is particularly well suited for use in VLB, as every flow can choose its intermediate switch to bounce off of from among the same set of switches – the dark gray intermediate switches in Figure 6.

3.6 Control Plane

The Monsoon control plane has two main responsibilities: (1), maintaining the forwarding tables in the switches; and (2), operating a directory service that tracks the port at which every server is connected to the network, as well as the server's IP and MAC addresses.

Maintaining Forwarding Tables: As outlined in Section 3.2, Monsoon requires that every switch have a forwarding table with an entry for every other switch. Any technique that computes routes among the ≈5K switches in the data center could be used.

Inspired by earlier proposals [11, 16, 12], we program the top-of-rack (TOR) switches to track the IP and MAC addresses of the servers directly connected to them, and announce this information in a Link-State Advertisement (LSA). Unlike earlier proposals, where the switches run a link-state routing protocol among themselves, we use logically-centralized routing based on the 4D architecture to compute the forwarding tables for the switches and recompute the tables as needed in response to failures. Tesseract [17] demonstrates that centralized control implemented using decision elements scales easily to manage 1000 switches, so computing routes for the roughly 5,000 switches in a 100,000 server data center does not appear difficult.

To eliminate the scaling problems caused by broadcast traffic, we program the switches to forward any packet for which they do not have a forwarding entry to the decision elements. This traps any broadcast packet a server might send (e.g., DHCP requests), and allows the decision element to decide how to handle it. To prevent the transparent learning algorithms from altering the forwarding tables created by the decision element, we disable.

Maintaining a Directory Service: There are many ways to implement the directory service required by Monsoon. A simple method is for the decision elements that run the control plane to also offer the directory service that maps the IP address of a server to a list of (server MAC address, top-of-rack switch MAC address) tuples and a list of intermediate node MAC addresses. This design is simple to implement as much of the information needed to populate the directory service comes from the LSAs obtained from the top-of-rack switches and is already available to the decision elements.

When a server crashes, however, it may take some time before the LSA updates and the server is removed from the directory service. Luckily, most data center applications already provide a health service that monitors the servers (e.g., AutoPilot [6]). Monsoon leverages this health service to remove failed servers from the pools across which requests are load-balanced.

We are currently designing and evaluating a directory service that is both scalable and quick to remove failed servers from the pools of which they were part.

4. RELATED WORK

SEATTLE [7] and Monsoon are similar in several respects. Both implement a large layer-2 network, and both bounce at least some packets off an intermediate switch. SEATTLE stores the location at which each server is connected to the network in a one-hop DHT distributed across the switches. The first few packets to a server are directed to the switch that knows where the server is attached, and subsequent packets short-cut directly to the server's top-of-rack switch. SEATTLE is also completely backwards compatible with existing network stacks (e.g., hosts use ARP). Monsoon deliberately bounces all packets off a randomly chosen intermediate switch in order to implement Valiant Load Balancing. Monsoon does make changes to the server network stack to turn off ARP and implement other primitives, such a load spreading, that are useful in data center applications. The Monsoon directory service is simpler than SEATTLE, using replication for reliability rather than a fully distributed DHT.

There is a significant body of work on implementing new network functionality by programming switches [14] or remotely controlling their forwarding tables [4, 10, 1]. Monsoon leverages these ideas and applies them to data center networks, showing how commodity switches can be used in a scale-out infrastructure. In particular, Ethane [1] is optimized for rigorous security, while Monsoon is optimized to create a network with high bandwidth between all pairs of servers. In data centers, rigorous security is implemented on the servers themselves via SSL or IPSEC.

5. SUMMARY

Given the scale, cost and importance of emerging cloud service data centers, it is incumbent on the networking community to rethink the components and overall architecture of their networks. Components of these data centers include: powerful multi-core servers with Gigabit speed network ports at remarkably low price points; distributed systems that automate the configuration and management of hundreds of thousands of servers and switches; and all of this under a single organization's control. Together, these factors open up the opportunity for fundamental change to servers and switches internal to the data center, while still maintaining an external IP interface. Unfortunately, the prevailing data center network architecture falls short of realizing the full benefits of these components. In particular, in today's cloud services data centers, network and server capacity is fragmented, and bisection bandwidth is one to two orders of magnitude below aggregate server bandwidth.

In Monsoon, we propose a network design that leverages the power of emerging data center components. We use programmable commodity switches to drive down the cost of the network while simulataneously increasing performance through the use of Valiant Load Balancing. We exploit the ability to put networking functionality into hosts to realize disaggregated scalable load balancing on the commodity servers. With functionality thus refactored, we scale the control plane and data plane to support a huge layer 2 switching domain providing full bandwidth between all servers in the data center.

Acknowledgements

The many insightful comments provided by the anonymous reviewers greatly improved the final version of this paper.

6. REFERENCES

[1] M. Casado, M. Freedman, J. Pettit, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. In *SIGCOMM*, 2007.

[2] Cisco systems: Data center: Load balancing data center services, 2004.

[3] N. G. Duffield, P. Goyal, A. G. Greenberg, P. P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe. A flexible model for resource management in virtual private network. In *SIGCOMM*, 1999.

[4] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe. The case for separating routing from routers. In *FDNA Workshop*, 2004.

[5] IEEE 802.1ah standard. http://www.ieee802.org/1/pages/802.1ah.html, 2008.

[6] M. Isard. Autopilot: Automatic data center management. *Operating Systems Review*, 41(2), 2007.

[7] C. Kim, M. Caesar, and J. Rexford. Floodless in SEATTLE: a scalable ethernet architecture for large enterprises. In *SIGCOMM*, 2008.

[8] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta. A Versatile Scheme for Routing Highly Variable Traffic in Service Overlays and IP Backbones. In *INFOCOM*, 2006.

[9] C. Kopparapu. *Load Balancing Servers, Firewalls, and Caches*. John Wisely & Sons Inc., 2002.

[10] T. V. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo. The SoftRouter architecture. In *HotNets*, Nov. 2004.

[11] A. Myers, T. S. E. Ng, and H. Zhang. Rethinking the service model: Scaling Ethernet to a million nodes. In *HotNets*, Nov. 2004.

[12] R. Perlman. RBridges: transparent routing. In *INFOCOM*, 2004.

[13] E. R. Hinden. Virtual router redundancy protocol (VRRP). RFC 3768, 2004.

[14] S. Rooney, J. van der Merwe, S. Crosby, and I. Leslie. The Tempest: a framework for safe, resource assured, programmable networks. *IEEE Trans on Comm*, 36(10):42–53, Oct 1998.

[15] S. Sinha, S. Kandula, and D. Katabi. Harnessing TCP's burstiness with flowlet switching. In *HotNets*, 2004.

[16] IETF TRILL Working Group. http://tools.ietf.org/wg/trill/, 2008.

[17] H. Yan, D. A. Maltz, T. S. E. Ng, H. Gogineni, H. Zhang, and Z. Cai. Tesseract: A 4D network control plane. In *NSDI*, Apr. 2007.

[18] R. Zhang-Shen and N. McKeown. Designing a Predictable Internet Backbone Network. In *HotNets*, 2004.