Killer Fabrics for Scalable Datacenters

Michael Schlansker, Jean Tourrilhes, Jose Renato Santos, Yoshio Turner

HP Laboratories
HPL-2009-26

Keyword(s):
Networks, Ethernet, datacenter, routing, multipath, fat tree, bandwidth, hash, TCA

Abstract:

Large-scale datacenters are rapidly increasing in number and size to satisfy computing and storage needs for globally connected businesses and the World Wide Web. 10Gb/s Ethernet is now being adopted to meet increasing bandwidth needs for in-datacenter communications. However, most datacenter network architectures are still based on specialized hierarchical edge-core topologies which are costly to build, difficult to maintain and consume large amounts of power. This paper describes enhancements to Layer two Ethernet switches that support multipath L2 routing for scalable datacenters. This enables a cost-effective scalable network architecture based on enhanced layer two Ethernet switches. The architecture provides multipath routing for Ethernet while preserving important Ethernet features and interoperability with traditional Ethernet gear. While this work is currently limited to scalable fat tree networks, these topologies provide an attractive approach for scaling large datacenter networks.

External Posting Date: February 6, 2009 [Fulltext] Approved for External Publication
Internal Posting Date: February 6, 2009 [Fulltext]

Submitted to SIGCOMM, 2009

Killer Fabrics for Scalable Datacenters

Michael Schlansker, Jean Tourrilhes, Jose Renato Santos, Yoshio Turner
Hewlett-Packard Laboratories
mike_schlansker@hp.com, jean.tourrilhes@hp.com, joserenato.santos@hp.com, yoshio.turner@hp.com

ABSTRACT

Large-scale datacenters are rapidly increasing in number and size to satisfy computing and storage needs for globally connected businesses and the World Wide Web. 10Gb/s Ethernet is now being adopted to meet increasing bandwidth needs for in-datacenter communications. However, most datacenter network architectures are still based on specialized hierarchical edge-core topologies which are costly to build, difficult to maintain and consume large amounts of power. This paper describes enhancements to Layer two Ethernet switches that support multipath L2 routing for scalable datacenters. This enables a cost-effective scalable network architecture based on enhanced layer two Ethernet switches. The architecture provides multipath routing for Ethernet while preserving important Ethernet features and interoperability with traditional Ethernet gear. While this work is currently limited to scalable fat tree networks, these topologies provide an attractive approach for scaling large datacenter networks.

Keywords

Networks, Ethernet, datacenter, routing, multipath, fat tree, bandwidth, hash, TCAM.

1. INTRODUCTION

This paper addresses two key problems that limit the use of Ethernet in datacenters. First, we need network architectures that scale with increasing datacenter needs. We need flat layer two networks that seamlessly scale without complex administration. Second, we need network architectures that are constructed using the same devices that are used in more general commodity network settings. We need to exploit inexpensive components used in non datacenter settings to drive down the cost for scaling in-datacenter communications.

Datacenters support today's information infrastructure providing scalable application, database, and file services needed to power the internet and complex enterprise applications. In addition, scalable cluster computers have evolved to solve the technical problems needed for large-scale scientific and engineering applications. Scalable performance relies on pools of communicating processor, storage, and network devices. Fault tolerance is required for always on infrastructure and server virtualization for management flexibility. These requirements lead to a common need for scalable datacenter networks that provide flexible communications that support complex interactions between datacenter components.

Due to Ethernet's broad market penetration, large investments are made in Ethernet hardware that exploit the capabilities of highly-integrated VLSI components. This provides steady price decreases and performance increases so that current Ethernet speeds approach that of specialized high-end datacenter fabrics. Ethernet is now positioned to satisfy the raw low-level communication needs of most commercial datacenter applications.

These changes provide increasing motivation to exploit Ethernet in very large datacenters and mirror the prior replacement of specialized and expensive high-performance computers with inexpensive microprocessors called "killer micros". At that time, specialized processors dominated high-end computing, and microprocessors emerged as the commodity compute solution. The combined power of silicon chips and large engineering efforts funded by general purpose computer revenues drove microprocessor performance. Eventually, economies of scale won, and supercomputers are now constructed using many commodity microprocessors. The moral of the story is: good enough wins, volume matters, and the low-end eats the high-end.

This work rests on the assumption that history will repeat itself for networking. High-end datacenter networks will be implemented using ensembles of evolved Ethernet components that share implementation architectures with broadly deployed Ethernet. We call these "killer fabrics". We are designing killer fabrics that use enhanced commodity network components to support the needs of tomorrow's scalable datacenters.

Ethernet faces significant obstacles for large-scale datacenters. A key limitation is Ethernet's inability to efficiently scale [1]. Therefore, datacenter architects cannot currently exploit the aggregate communication capability of many network components. Key limitations arise from the use of the spanning tree protocol which is at the heart of transparent bridging [2]. Transparent bridging allows efficient communication to newly added end-nodes without administrative action. Spanning tree eliminates cycles and facilitates broadcasts needed to locate missing end-nodes. However, the advantages of spanning tree are offset by disadvantages that limit Ethernet's ability to exploit redundant paths. This leads to edge-core topologies that connect inexpensive edge switches with expensive monolithic core switches that are needed to provide adequate bisection bandwidth.

The use of monolithic core switches is impractical for large scale datacenters. An attractive alternative replaces the monolithic core switch with multiple lower cost core switches. However, to preserve high bisection bandwidth, multiple core switches must exploit multiple network paths. Existing approaches can exploit multiple paths on Ethernet. Network management can statically partition networks using layer-three IP subnets and/or layer-two VLANs. Both approaches limit the scope of flat layer two networks and assist in exploiting multiple paths. However, these approaches require complex and costly manual administration. Static network partitioning does not work when performing datacenter-wide any-to-any communications for data intensive operations. Static network partitioning also restricts virtualization which dynamically moves virtual machines and requires dynamic changes in network connectivity. Datacenters need flat scalable layer-two networks with dynamic multipath L2 routing to eliminate costly core switches and complex management.

Storage attachment also offers challenges not easily solved by today's Ethernet. This is primarily due to Ethernet's dropped packets and unpredictable performance under congested load. Hence, storage is often attached using Fibre Channel or InfiniBand networks. This leads to datacenters that require separate networks for compute-to-compute and compute-to-storage communications and inhibits the use of shared infrastructure for multiple communication needs.

1.1 Contributions

In order to satisfy future datacenter needs, fabric architectures must be developed to overcome a number of key technical obstacles while retaining key Ethernet compatibilities.

In this paper, we introduce a new hash-based routing architecture that enhances Ethernet to support multipath routing within scalable datacenter networks. This architecture enables the use of scalable networks of commodity Ethernet switches for large and communication-intensive datacenters. Our proposed improvements provide an evolutionary path for Ethernet switch implementations to incorporate both static and dynamic multipath routing. The architecture preserves important Ethernet properties including plug-and-play operation, support for end station mobility, and seamless interoperability with conventional Ethernet networks that are attached at the edge of the datacenter.

We identify dynamic path selection algorithms to handle time varying traffic within a datacenter. These algorithms have been implemented and provide substantial improvement in network performance compared to previous random static scheduling techniques. We explore the scalability of these algorithms and show that they are adequate for very large datacenters.

To reduce the high cost of multiple datacenter networks, our architecture supports converged-fabric datacenters that transport LAN and storage traffic on a common network. Our architecture provides a foundation for the active management of diverse traffic types to balance the needs of high throughput, predictable quality of service, fault tolerance, and power management. This includes support for the layer-two functionality needed for both IP and non-IP (e.g. Fibre Channel over Ethernet) traffic.

Our contributions can be divided into low-level hardware mechanisms and high-level software architectures. The low-level hardware mechanisms preserve and enhance switch micro-architectures that will often be used for non-datacenter needs. This allows the broadest deployment of networks based on commodity chips and amortizes costly chip design efforts over a broad base that includes both commodity and more specialized datacenter networks. Switch enhancements provide powerful low-level multipath routing support and a strong hardware foundation, for future high-level software architectures.

We have developed initial software architectures for real datacenter networks. These architectures provide both a practical demonstrator for the power of low-level hardware mechanisms as well as a prototype for first generation datacenter network management architectures.

1.2 Outline of paper

In section 2 we introduce a hash based multipath L2 routing architecture that upgrades Ethernet to support routing in fat tree networks for scalable datacenters. Section 3 describes algorithms that provide multipath load balancing. Section 4 presents results that demonstrate the performance of multipath load balancing. Section 5 explores the scalability of centrally managed networks using our algorithms. Section 6 describes how our network architecture preserves important capabilities of traditional Ethernet. Section 7 describes features of the proposed architecture for converged-fabric networks that combine LAN, storage, and other traffic types. Section 8 describes related work, and section 9 provides conclusions.

2. Hash-based routing for scalable networks

This section introduces architectural principles for hash-based L2 routing in scalable fat tree networks. Hash-based routing is designed to be compatible with existing Ethernet mechanisms for learning and plug-and-play operation, and enables the utilization of redundant paths in the fabric. The approach is applicable to multi-level fat-tree topologies with the flexibility to independently scale the bisection bandwidth and edge port count. The required enhancements to the switch micro-architecture greatly leverage existing hardware structures and require only modest extensions.

2.1 Existing Ethernet

Ethernet uses the spanning tree protocol to identify an active acyclic network that transports all packets [2]. Spanning tree establishes a cycle-free backbone for broadcast information and identifies a unique path between every pair of end stations, thus making all forwarding decisions unambiguous. After a spanning tree is identified, switches use a simple initial forwarding procedure to send a message to a destination at an unknown location. When a packet is sent from any end stations A to B, each switch forwards an incoming message on all active (spanning tree) ports except the port on which the packet arrived. This process, called flooding, is performed with no forwarding information except spanning tree information that defines active ports. However, this broadcast-based procedure is inefficient.

Adaptive forwarding enhances performance by learning a path to each destination. A forwarding cache supports layer two learning. Each packet contains source and destination MAC addresses. While the destination address is used for forwarding, the source address is used for learning. When a message is received on a link L, with source address X, then a forwarding cache entry is created or updated so that subsequent messages destined for X are forwarded on L. For each packet, the cache is searched using its destination MAC address to identify a forwarding port that may have been recorded when a prior message was received from the destination. If present, the forwarding port is used to send data to a next hop switch. If the cache entry is lost due to replacement or timeout, then the switch reverts to flooding.

Traditional Ethernet restricts the use of redundant paths leading to great difficulty in large-scale datacenters. As a network scales to support additional hosts, bisection bandwidth is added to support increasing communication across the entire network. When conventional Ethernet is used to implement the datacenter fabric, larger and larger core switches are required at the root of a spanning tree to provide adequate bandwidth. Each core switch is traditionally implemented as a complex and costly system with a proprietary internal architecture and cannot be implemented by assembling many inexpensive and off-the-shelf commodity Ethernet switches.

2.2 Fat tree network topologies

Network topologies such as Clos networks or fat trees can scale to arbitrarily large size using fixed port-count switches avoiding costly core switches. These networks independently scale in both the number of ports and bisection bandwidth. Similar networks have been used for many years first in telephone switching [3], then for computer interconnection networks [4], and more recently for scalable Ethernet networks [5][6]. We present a family of scalable network architectures, based on these topologies, for datacenter networking.

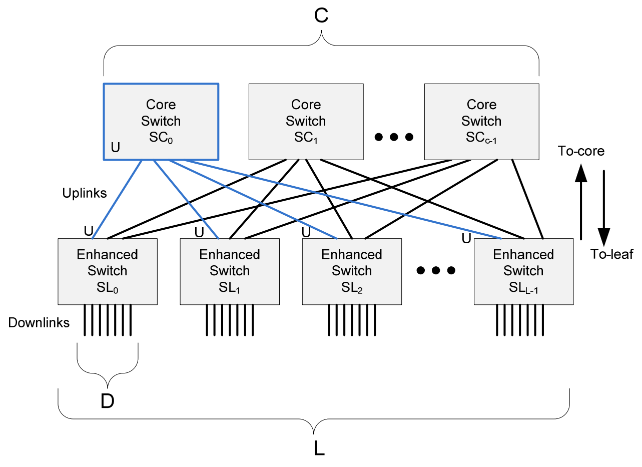

Figure 1 - Two-Tiered Fat Tree

Figure 1 illustrates that two-tiered fat trees can offer independent control over both the number of edge ports and the bisection bandwidth. This allows datacenter architects to tailor networks for varying communication needs. Exposed edge ports appear at the bottom where D downlinks are provided by each of L leaf switches. As leaf switches are added, the number of exposed edge ports expands as D×L. Adding core switches (C) enhances the bisection bandwidth which grows as C×L.

A key feature and a key difficulty presented by these networks is that multiple paths exist between every pair of edge ports. When a packet is injected into a leaf switch, an uplink choice is confronted that is not typically seen with traditional L2 Ethernet. However, to avoid network cycles conventional switches run Ethernet's spanning tree algorithm which disables redundant links and thus eliminates the bandwidth benefits of all but one of the core switches. Multipath L2 routing is needed to exploit the bandwidth that is provided by scalable and redundant fat tree networks.

A two tiered network has a limited maximal size as dictated by the fan-out (radix) of switches that are available at any time. In particular, core switches must provide at least L ports while edge switches must provide C+D ports. With next generation 48 port 10Gb Ethernet switches, very large two tiered networks can be constructed, as we describe in Section 5.

2.3 Routing approach

Multipath routing can be divided into static and dynamic approaches. Static routing describes how packets are routed without measuring load to rebalance traffic. Dynamic routing provides a control loop: data is collected to measure points of congestion within the network; new routing information is calculated; and then new routing tables are set to improve throughput by rerouting traffic around congestion. Dynamic routing should be made on global measurements, as a local decision can impact a remote bottleneck.

A central issue with dynamic routing is the overhead of management traffic and routing computation. A good way to reduce both the management traffic and routing computation is to not individuate every flow but to assign flows to a limited number of hash classes. Hash-based routing provides a powerful tool for managing multipath traffic within fat-tree-style Ethernet networks. It provides an effective approach for both static and dynamic routing. Packet source and destination addresses are processed using a hash to pseudo randomly assign flows to hash classes. Each packet's forwarding route is specified based on the resulting hash class. Hash functions may use higher layer (e.g. IP) information in addition to MAC addresses to facilitate randomization across multiple paths. Hash functions are selected so that all packets of a network flow map to one hash class. This ensures that packets of the same flow traverse a single network path, and preserves packet order within the flow.

Hash-based routing can be divided into asymmetric and symmetric counterparts. For each source address S, and each destination address D, H(S, D) represents the hash class for every message sent from S to D. Symmetric routing uses symmetric hashes that have H(S, D) = H(D, S), and a symmetric placement of routes. The symmetry property is used to ensure that after a message flows through a path from a source to any destination, that the same path is retraced by a reply. Symmetric hash routing facilitates the learning process that preserves the plug and play Ethernet. Experiments below rely on a hash that adds source and destination MAC addresses, multiplies by a large prime, and masks bits to yield a result.

Hash-based routing actively manages the flow of packets through a fat tree as shown in Figure 1. Routing is not limited to full bisection bandwidth networks – low cost networks with reduced bisection bandwidth are also supported. Core switches are assumed to be unmodified Ethernet switches. Leaf switches are enhanced to support multipath routing but preserve the feature set and micro architecture of a conventional Ethernet switch.

The fat tree topology greatly simplifies the routing problem. When a message flows through a fat tree from any source to any destination, a series of one or more uplink choices are confronted until reaching a switch S that can route downward to the destination. At each uplink choice, any selected uplink identifies a next hop on one of the shortest paths to the desired destination. A hash function assigns all uplink messages to randomly selected hash classes. Active load balancing routes hash classes on carefully selected uplinks. Due to the nature of fat trees, uplink selection does not dictate whether any specific destination is reached and can be freely used for load balancing or traffic segregation independent of any concern for correct routing.

After reaching switch S, a sequence of downlink choices are confronted. At each tier, a single correct downlink must be selected to reach the target. The correct identification of downlinks uses traditional layer-two forwarding. Therefore unmodified core switches can be used and the architecture preserves the simple administration of plug & play networks which results from learning.

2.4 Plug-and-play Ethernet

Ethernet's spanning tree and learning functions provide plug-and-play operation, allowing dynamic addition and removal of switches and end stations at the edge of a fabric. Enhancements to leaf switches are needed to combine Ethernet's existing plug-and-play with enhanced multi-path routing.

To prevent the spanning tree protocol from disabling any uplinks at each leaf switch, we carefully control which uplinks participate in the spanning tree protocol. The uplinks are chosen such that all participating uplinks join the spanning tree. Non-participating uplinks remain enabled. In particular, we designate one core switch as the *Ucore*. In each leaf, the uplink *Uport* is connected to the Ucore (see fig 1). The leaf is modified to identify the Uport as well as all downlinks as participating links in the spanning tree protocol, and to disable spanning tree on the remaining uplinks. The rest of the cores are effectively excluded from the spanning tree protocol, rendering spanning tree formation identical to a conventional network with a single core. The Ucore is configured to be the root for spanning tree formation. The spanning tree protocol systematically enables links that are on least cost paths to the root. The use of the Ucore as the root guarantees that the Uport links are spanning tree members even when additional switches are added that interconnect multiple leaf switches.

To make Ethernet learning compatible with multi-path routing, all the uplinks of a leaf switch are treated as a single virtual uplink. The Uport provides a single virtual port name for all uplink ports. All downward traversing packets that are received by a leaf switch on an uplink look as if they were received on the Uport. Packets that are destined to the Uport are processed by the hash routing mechanism, which implements a multipath routing choice to determine the chosen physical uplink on which to forward.

Leaf switch downlinks, and all core links, provide a single path to any given destination. Therefore, for downward routing, the forwarding path can be learned by the adaptive forwarding. By building downward routing on traditional adaptive forwarding, Ethernet plug and play and device mobility are preserved.

2.5 The Enhanced Ethernet switch

Ethernet switches commonly use a processing pipeline that includes the packet header processing, forwarding cache, and TCAM lookup stages. Figure 2 shows these functions in white boxes illustrating the conventional portion of an Ethernet switch. The enhanced Ethernet switch preserves these features but is enhanced by adding blocks (shown in gray) that support multipath routing and active load balancing. Those simple additions include the uplink identifier, hash, and gather hash statistics blocks.

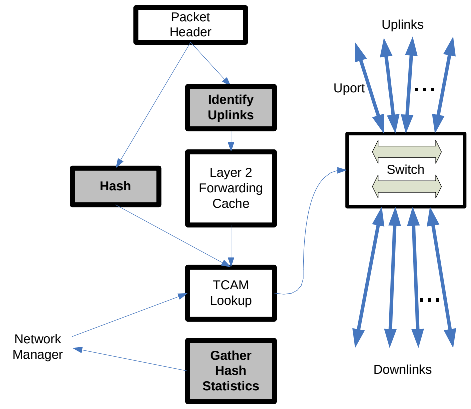

Figure 2: Enhanced Switch Microarchitecture

The role of the identify uplink block is to substitute the Uport as the arrival port for any packet that arrives on an uplink. As a result, the forwarding cache treats all uplink arrivals as arrivals on Uport. In traditional switches, link aggregation groups (LAGs) are groups of links that are treated as a single link to aggregate link bandwidth between pairs of switches. Our switch treats uplinks in a manner similar to conventional LAGs, and existing LAG support may be reused for this block.

Arriving packets are then processed by the learning function in the forwarding cache. If the packet was received from a downlink, it enters the source MAC address and the receiving port into the forwarding table. If the packet was received on an uplink port, the “identify uplink” block has substituted the Uport name as the name of the port, and an entry is created in the forwarding table that associates the packet’s source address with the Uport.

The forwarding cache then processes the packet’s destination to determine a forwarding port. Destination address processing specifies one of three results: the forwarding port is either: unknown, a downlink, or the Uport. When the port is unknown, the packet is flooded on the Uport and all downlinks (except the arrival port). When the forwarding port specifies a downlink, the packet is forwarded on that link. When the forwarding port specifies the Uport, the actual port will be one of the uplinks and is selected in the subsequent TCAM lookup.

The hash block computes the hash field from the packet header. This hash provides routing diversity for small lookup tables and may include a variety of header fields.

The actual multipath routing and load balancing can be done using a TCAM or using a RAM-based lookup table. The TCAM within existing Ethernet switches enhances flexibility and supports higher layer functions. A frequent traditional use for TCAMs is to implement access control lists and QoS. We prefer TCAM-based implementations for a variety of reasons. This allows the reuse of an existing function rather than adding a single purpose block. TCAMs allow joint processing of the hash value with other header fields, such as VLAN, Ethertype, and type or class of service, for differentiating flows. The use of TCAMs also enables rules that override hash based routing for specific hosts or flows, for example to route control traffic to a specific core switch itself.

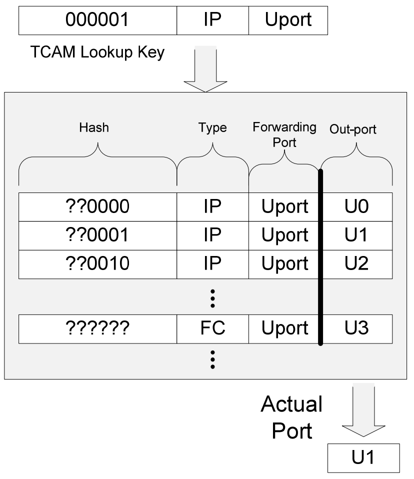

Figure 3: Load Balancing with TCAM Lookup Table

A TCAM lookup key, shown in Figure 3, is formed by combining the hash field and the destination port field resulting from the forwarding cache lookup. Additional TCAM fields distinguish packet types that receive specialized treatment. Our example uses traffic types IP (IP) and Fibre Channel over Ethernet (FC). The key is used as an input for TCAM lookup that queries the entire contents of an associative table for a match. A match identifies a result out-port that directs traffic to a carefully selected uplink.

The TCAM lookup table entries are set by the network manager to distribute load across multiple uplinks. Each entry is matched against the lookup key for a zero, one, or don’t care in each field position. Each entry contains a match field (including hash, type, and forwarding port) as well as the resulting out-port field that identifies the actual port for a matching entry. If the hash value and forwarding port TCAM fields match corresponding key fields, then the out-port field is substituted as the actual uplink. Our example TCAM lookup matches the second TCAM entry and selects uplink U1 for the actual port.

Consider an example where IP traffic is balanced over uplinks. Traffic is divided into 16 groups where each group is identified by a hash class. The example ignores two of the six hash bits and matches the remaining four bits to select a corresponding hash class. Load balancing is accomplished as each hash class receives a carefully selected assignment of traffic to an uplink. For each possible hash (e.g. 16 combinations for a 4-bit hash field), one TCAM entry is created that contains the hash constant that identifies the chosen traffic group. Additionally, the entry specifies the forwarding port as "Uport" and the type as "IP". Finally, the entry specifies an out port field that selects the actual uplink choice.

Non-matching packets are unprocessed and the forwarding port is used as the actual output port. Packets destined for downlinks do not match the Uport in the forwarding port field and no action results. After successful processing in the forwarding table, a packet that is destined for any uplink matches the Uport field. However, each uplink packet matches the hash field for a single TCAM entry. This entry determines the actual port substituted by the TCAM and specifies the chosen uplink.

Types IP and FC are matched and treated separately. This allows for the type-specific treatment of traffic and will be discussed in further detail in section 7.

2.6 Recursive network construction

Hash-based routing is scalable to networks of arbitrary size. In particular, the two-tiered fat tree shown in Figure 1 can be implemented hierarchically, expanded to arbitrary size, and controlled using hash-based routing.

If a two-tiered network were expanded to a very large network, core switches with a very large number of ports would be needed. Due to chip pin-out and area limitations, core switches with arbitrary port count cannot be constructed using a single-chip. However, large port count core switches can be constructed by noting that the fat tree of Figure 1 itself has all of the properties needed to serve as a core switch for a larger fat tree. This allows the recursive construction of large networks. The substitution of a compound fat tree network to provide a core switch within a larger fat-tree can be repeated to form networks of arbitrary scale. Our load balancing approach, presented below, can be hierarchically applied within each compound core switch and among core switches to control networks of arbitrary scale.

3. Active load balancing

This section demonstrates hash-based routing using a centralized control architecture for active management. While decentralized approaches are of great interest, this paper describes a centralized management approach which is appealing due to the inherent simplicity. Centralized managers quickly propagate information throughout a fabric but suffer inherent risks in limiting scalability. We demonstrate surprisingly good scalability for centrally managed fabrics below.

Active load balancing can be separated into symmetric and asymmetric approaches. For symmetric load balancing, all round-trip communications traverse a single bidirectional path. While the asymmetric approach provides higher performance, the symmetric approach simplifies data collection, route computation and Ethernet learning.

3.1 Data collection

Active load balancing requires measurements that identify network bottlenecks. Each leaf switch samples packets at a rate that balances processing costs with sampling accuracy. Measurements are gathered in separate hash classes for hash-based multipath routing. Packets from one downlink to another are ignored. For packets that traverse an uplink, a histogram entry is augmented for that packet's hash class. Separate data is collected for the to-core (upward) and to-leaf (downward) directions on each uplink. For the to-core direction, each switch accumulates a single histogram value for each hash class. For the to-leaf direction, each switch accumulates a single histogram value for symmetric routing and a vector for asymmetric routing. Many TCAM implementations include a hit count for each TCAM entry. In most cases, these hit counts are sufficient to gather measurements for symmetric routing.

3.2 Load balancing for symmetric routing

Symmetric routing ensures that a common path is used for round trip communications. Symmetric routing relies on a symmetric hash with H(S, D) = H(D, S). To improve randomization, layer-two and higher-layer Ethernet information may be incorporated into source and destination specifications as long as the same hash value results for outbound and returning messages for round-trip communications. To preserve route symmetry, while each hash class can be placed independently on any uplink, each hash class must be placed on corresponding uplinks leading to the same core switch for all leaf switches. This ensures that round trip communications traverse a common path between any source and destination pair. For the symmetric case, each leaf switch collects link use statistics for the to-core (upward) and to-leaf (downward) directions. For each switch, two values are needed for each hash class that indicate the up-bound and down-bound traffic resulting from the network-wide placement of a given hash class to a selected uplink.

A key benefit for the symmetric approach is that, due to the round-trip nature of most communications, core switches can use traditional learning to determine destination paths. When a core switch needs to forward a packet it is almost certain that the same switch has, or will see, a reply packet that learns the destination port and avoids flooding. When symmetry is relaxed, replies to a message that traverses one core switch may traverse an alternate core switch, and learning may fail to eliminate core switch flooding.

We have developed heuristics to optimize multipath traffic based on measured traffic flow. Our greedy algorithm iterates across hash classes and optimizes uplink selection for each class after prior classes have been placed. In each step, an uplink is selected that minimizes the sum of squared traffic across all links. Minimizing square traffic places high penalties on heavily loaded links.

After all hash classes are assigned to uplinks, the central controller downloads TCAM entries for all switches. For slow time varying traffic, we show that network performance is greatly improved using this process.

3.3 Load balancing for asymmetric routing

Asymmetric load balancing follows the approach of symmetric load balancing with a few key differences. Asymmetric routing relaxes the need for a symmetric hash function and a distinct uplink is independently selected for each hash class on each leaf switches.

Additional sample data is needed for asymmetric routing. For the to-core direction, the uplink choice made for each hash class on each leaf switch has a local effect that is measured with a scalar value at each source leaf switch. For the to-leaf direction, the uplink choice made for each hash class on each switch has a remote and distributed effect that is measured as a vector of values at each destination leaf switch. Each value measures the effect of a routing choice in the i^{th} source switch on the to-leaf link resource usage at the given destination switch.

For packets that arrive across leaf switch uplinks in the to-leaf direction, asymmetric routing requires sampling logic that identifies the originating source leaf switch and accumulates measured load in a vector. While this can be accomplished, this represents one of the difficulties for asymmetric routing. One approach maintains a table associating each source MAC address with a hosting leaf switch. This table is replicated in each switch and referenced for each sampled to-leaf packet. However, maintaining this table interferes with plug-and-play support as, for example, when an end station is moved from one leaf switch to another, tables must be updated.

Another issue for asymmetric routing is that adaptive forwarding in the core switch may not properly learn MAC addresses. The core switch may see only packets destined to a MAC address and never see packet coming from that address, those packets being routed through another core switch. For asymmetric routing to work, either core switches need modification or the edge switches needs to send gratuitous packets to cause the core switch to learn needed MAC addresses.

3.4 Load balancing comparison

The symmetric and asymmetric load balancing approaches are summarized in Table 1. Common to both approaches are a set of architectural features described above that are summarized in the common responsibilities cell. Common properties are features that are provided by the architecture for both symmetric and asymmetric routing.

Table 1: Routing Comparison

Common Responsibilities	**Common Properties**
Randomize flows over multiple uplinks Leaf switches tolerate multipath Active management optimizes uplink Uplink choice preserved for each flow Single core switch for spanning tree	Supports scalable construction Correct multipath routing Can dynamically balance load Per-flow packet ordering preserved Spanning tree for attached switches
Symmetric Responsibilities	**Symmetric Properties**
Consistent path for round trips No need to track message origin	L2 learning works seamlessly Reduced sample data size Reduced optimization complexity Achieves very good performance
Asymmetric Responsibilities	**Asymmetric Properties**
Learning requires special support Must track message origin across core Need vector stats for to-leaf packets	Larger sample size Larger optimization complexity Achieves near optimal performance

4. Performance Results

A software-based simulation for the proposed architecture was developed to evaluate the architecture and heuristics. Current software models the two-tiered fat trees of Figure 1. Parameters include the number of core switches, leaf switches, and downlinks. The number of TCAM entries is parameterized and can be increased to enhance fine-grained routing decisions needed for larger networks. Each link is bidirectional, and inbound and outbound resources are independently modeled. For each resource, a flow that requests f flow units is completed on a capacity r link in time t=f/r.

Currently, a simple performance model demonstrates multipath routing benefits. Transmission begins when a number of flows begin crossing the network using a chosen policy. Each flow competes for bandwidth on each link on its routed path. Delivery is proportionally delayed when flows share a common link. Transmission ends when a final flow completes. The flow rate is calculated by summing the load over all flows and dividing by the transmission time through a worst case bottleneck link. This provides an aggregate measure of parallelism in the network's ability to process all flows. While this does not model real system dynamics, it does demonstrate the benefits of on-line optimization.

The model generates a number of flows that present a parameterized load using a uniform random selection of flow sources and flow destinations. Flows are routed using four policies. Spanning tree ("span") always chooses the Uport for uplink traffic and thus uses only a single core switch. Random ("rand") uses a round-robin mapping of TCAM entries to uplinks. When the number of TCAM entries is a multiple of the number of core switches, this corresponds to assigning an equal number of hash classes that contain approximately equal amounts of randomly selected traffic to each uplink. The "sym" and "asym" methods apply symmetric and asymmetric load balancing.

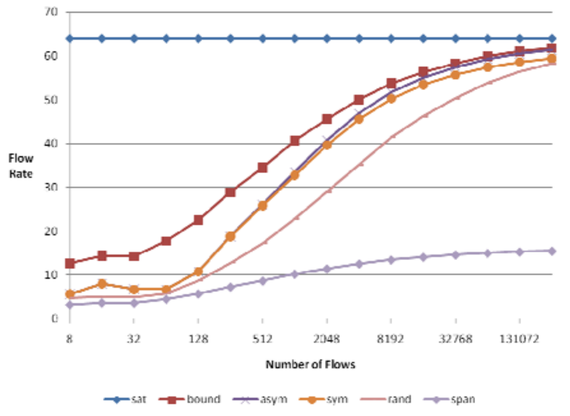

Figure 4: Non-uniform Routing

Figure 4 shows the performance for a network with: 4 core switches, 16 leaf switches, 16 downlinks per leaf, and 256 TCAM entries. All links have unit capacity. Random flows are added on a logarithmic scale as the number of flows is repeatedly doubled from 8 on the left to 262,144 on the right. Non-uniform flows are generated. Each flow is generated as an independent trial with probability .95 of acquiring a unit load and probability .05 of acquiring a load of 20. The flow rate is plotted as unit load equivalent flows completed per unit time. Data has been smoothed by averaging over 100 independent trial experiments.

Two performance bounds are shown. The bound "sat" is based only on the bisection bandwidth of network hardware and is independent of flows. Sat is achieved when all uplinks are simultaneously saturated in both directions. The example cannot transport more than C×L=64 unit flows per unit time. A tighter "bound" is calculated given a specific set of flows that present known loads for each switch. The sum of loads that traverse each switch's uplinks divided by the aggregate uplink capacity on that switch provides a time bound. A time bound is calculated separately for the to-core and to-leaf directions on each switch. The worst case time establishes an overall time bound for the entire set of flows.

For a modest number of flows, active management consistently produces large performance gains. For example, in Figure 4, for 4096 flows, symmetric load balancing ("sym") yields a 30% performance increase over random. Asymmetric load balancing outperforms symmetric load balancing at almost every point but typically by a small margin. For a large number of flows, (e.g. 8192) both asymmetric and symmetric load balancing approach the bound and are near optimal. For a very large number of flows (e.g. 262,144), "asym", "sym", and "rand" all produce near optimal ("sat") results.

For a small number of flows, all load balancing approaches remain substantially below the bound. This overly optimistic bound allows each individual flow to spread across multiple uplinks. For small numbers of flows, achieved results look poor when compared to this unrealistic bound. A tighter bound ("TB") is calculated when all flows have unit load. In the case, when N unit flows are transported across M unit capacity uplinks, then ceiling (M/N) units of time are required to complete all flows.

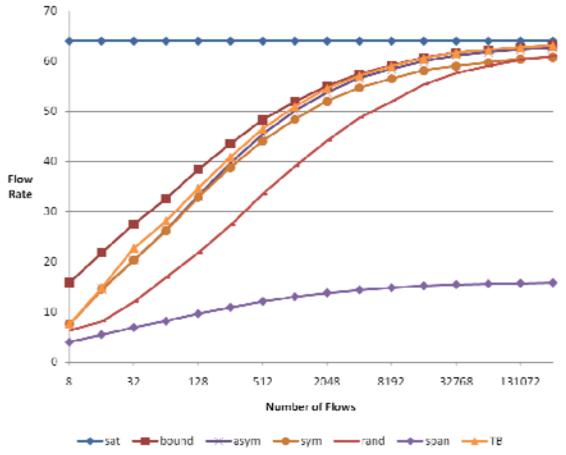

Figure 5: Uniform Routing

Figure 5 repeats the experiment of Figure 4 using uniform unit loads and plotting both bounds. Now, both symmetric and asymmetric management closely follow the TB bound. The remaining experiments use uniform loads and plot the TB bound.

Figure 6 explores symmetric routing performance while varying the number of hash bins. Experiments are performed for 16, 64, 256, and 1024 hash bins. While the performance for symmetric management with 16 hash bins mirrors random, performance for 1024 hash bins closely follows the TB bound. Results for asymmetric routing (not shown) are similar but, systematically outperform symmetric with the same number of hash bins.

Figure 7 explores network scalability for a varying number of core switches. This experiment uses 16 leaf switches and 512 hash bins. The number of flows is held constant at 100, while the bisection bandwidth is increased by adding core switches. Again, active management outperforms random for a modest number of flows, however, the growth in flow rate is limited due to the small number of flows.

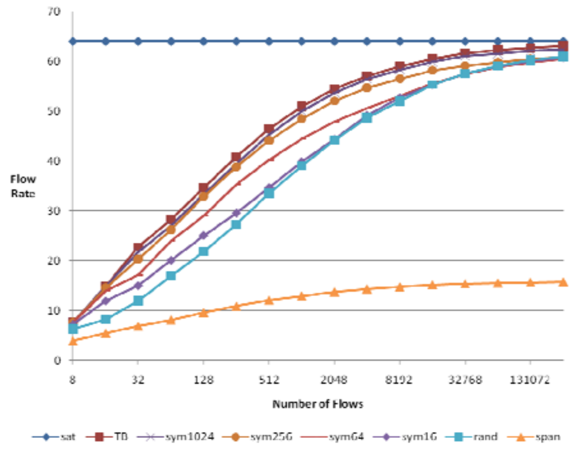

Figure 6: Flow Rate for Varying Hash Bins

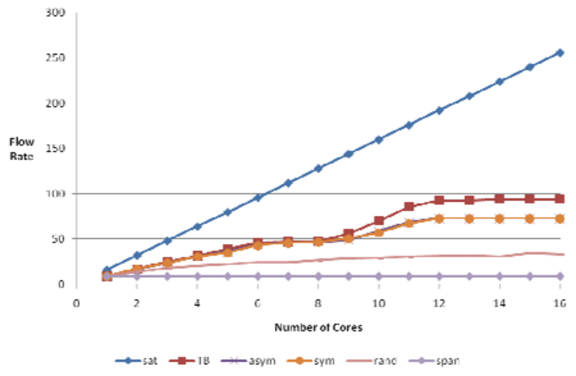

Figure 7: Flow Rate vs Core scaling for 100 Flows

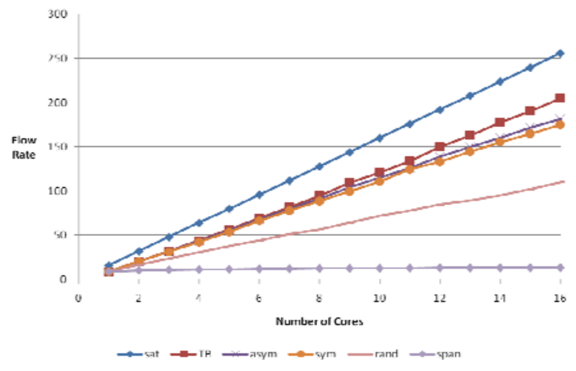

Figure 8: Flow Rate vs Core Scaling for 100×C Flows

Figure 8 repeats Figure 7 but scales the number of flows with the number of core switches by applying 100 flows per core switch. Flow rate now scales almost linearly with core switches and systematically outperforms random.

In summary, both symmetric and asymmetric hash-based routing architectures can efficiently route traffic for scalable datacenter networks. Several issues should be addressed before a practical asymmetric architecture is presented. A detailed and fair comparison of these two approaches is beyond the scope of this paper.

5. Scalability for Centralized Routing

A potential liability for large-scale network routing is the scalability of dynamic network management. We preface this discussion by emphasizing that the goals of this work do not include support for the very largest networks that can be constructed at any given time. Instead, our goals are to support the vast majority of real datacenters and to do so using a network constructed with commodity Ethernet switches with needed enhancements. Currently, our experiments rely on a central controller to actively manage a large network of switches. There are significant risks that either the communication costs between the central controller and managed switches, or central controller compute costs may preclude responsive management for networks of adequate scale. We hope to identify approaches for parallelizing control functions, but such work is beyond the scope of this paper.

A key benefit for hash-based routing architectures is that small amounts of summary data can be used to manage very large fabrics. We conducted experiments to evaluate the scalability capabilities of a centralized network controller. For these experiments, we assume a stressful situation where full bisection bandwidth is required for networks of varying scale. In contrast, networks with less than full bisection bandwidth have strictly lower hardware costs and management difficulty.

Table 2: Scalable Server Configurations

C - Core Switches	1	2	3	4	6	8	12	24
A – Ports per LAG	24	12	8	6	4	3	2	1
L - Leaf Switches	2	4	6	8	12	16	24	48
E - Exposed Ports	48	96	144	192	288	384	576	1152
S - Servers	480	960	1440	1920	2880	3840	5760	11520

To generate a scalable family of networks, 48-port switches are used to assemble two-tiered fat trees to achieve a desired number of exposed (available for host attachment) ports. The following detailed discussion, along with Table 2, describes the network construction approach used in these experiments.

In each leaf switch, the 48 ports are segregated into 24 downlinks and 24 uplinks. The 24 uplinks are aggregated into equal sized Link Aggregation Groups (LAGs). For each leaf switch, one LAG attaches to each core switch, so there are **C** LAGs. Consequently, the number of uplink ports per LAG is **A = 24/C**. Core switches have 48 ports, so they can support **L = 48/A** LAGs, and therefore can connect to **L = 48/A** leaf switches. Each leaf switch provides 24 downlinks, and thus we have **E = L*24** exposed 10Gb/s ports. We further assume 10 servers per exposed port, each sustaining 1Gb/s of data, for a total number of server **S = L*24*10**. The smallest network describes a useless two-tiered architecture for building a 48 port network out of three 48 port switches.

5.1 Scalability for Symmetric Control

As described in Sections 3.1 and 3.2, for symmetric hash-based routing each leaf switch samples packets, and maintains separate histogram entries for each hash bin and for both the to-core and to-leaf directions. A central controller collects histogram data from each switch and processes the data to determine routes within a scalable fabric.

For these experiments, we assume that each histogram entry is represented by a 32 bit sample value. The size of summary information for symmetric routing grows as H×L where H represents the number of hash bins. The load balancing computation yields a result that determines an uplink choice for each hash class. Output control results must be transmitted to each switch. While the same table could be broadcast to each switch, the transmission cost is currently evaluated as a separate unicast transaction per leaf switch. Thus, output control data also grows as H×L.

For these experiments, we also assume that all input to and output from a controller passes through a dedicated 1 Gb/s control network that connects the switches and controller. The amount of communication data for control is measured as the total number of input data bits plus the number of output control bits summed over all leaf switches. Communication time is computed by dividing the number of bits by the control network bandwidth. We assume 100% utilization of the control channel, so real communication costs will be higher. Communication times are not the dominant time for large systems so more careful analysis has not yet been performed.

Compute costs to determine routes also scale nonlinearly and represent a substantial bottleneck for large networks. While communication costs for symmetric routing grow as H×L, compute costs grow as $H \times L \times C^2$. Thus, compute costs can be subject to significant slowdowns for networks of very large scale.

In Figure 9 we plot the symmetric control communication time (“sym comm.”) and compute time (“sym comp”) on a logarithmic scale assuming that 1024 hash bins are used to manage a datacenter. Compute times are measured using a working implementation of our routing algorithms. Compute costs are measured on an Intel Core2© @2.13Ghz running on a single thread of execution. Communication times are computed by tabulating needed sample and control data and calculating time needed to pass all data across the 1Gb/s control channel.

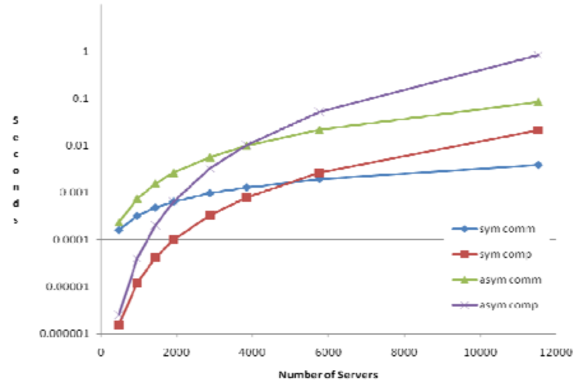

Figure 9: Performance for Symmetric and Asymmetric Control Traffic and Route Calculation

The update rate is the reciprocal of the larger of the communication and compute times. Initial objectives were to support a global reaction to changing traffic at rates of between 100 times and 10 times a second. This objective is easily met throughout most of the range. For large datacenters with 11520 hosts, compute costs dominate communication costs and limit the update rate to about 50 per second. For most datacenters of interest, we believe that this experiment demonstrates a surprisingly high responsiveness for centralized management using summary hash data. Efforts to improve the performance of sequential algorithms and to develop distributed parallel algorithms may provide additional room for growth.

5.2 Scalability for Asymmetric control

Asymmetric load balancing suffers scalability limitations due to the need to collect and distribute more data, and the need for more complex optimization algorithms. The size of summary information for the asymmetric routing case grows as H×L for uplink data, but downlink data requires $H \times L^2$ distinct samples as a per-source-switch vector is maintained for each hash class and in every destination switch. The cost for distributing TCAM control is H×L as a size H lookup table is distributed to each leaf switch. While the optimization algorithm is very similar to symmetric routing, a separate routing decision is made for each leaf switch thus increasing the computational requirements. While communications costs for asymmetric routing grow as H×L^2, compute costs for asymmetric routing scale as H×L^2×C^2.

Asymmetric load balancing compute costs can lead to a control bottleneck for large scale networks. As seen in Figure 9, compute costs limit the asymmetric update rate for a centralized network manager for 11520 servers to about 1 update per second. Again, it is hoped that the update rate can be improved with further effort in parallelizing or improving algorithms.

5.3 Performance Scalability

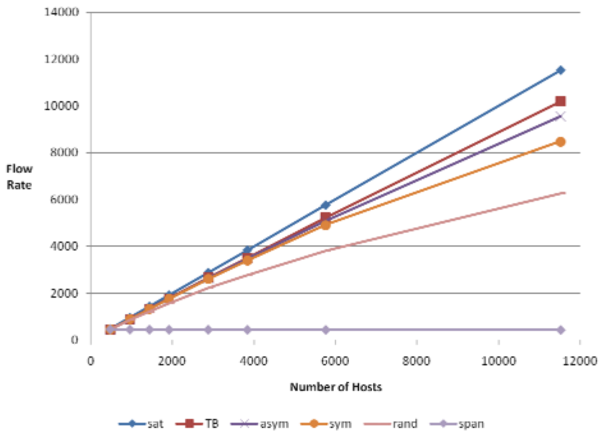

Figure10: Datacenter Performance Scalability

The performance scalability for the datacenters of Table 2 is demonstrated in Figure 10. In this experiment, datacenters are managed using 1024 hash entries, and the number of flows is two times the number of hosts. Additional experiments (not shown) demonstrate that while increasing the number of hash entries moves performance up toward TB, decreasing hash entries moves performance down toward random. While our demonstrated approach scales well, improved management responsiveness is of great interest.

6. Preserving Ethernet Features

Our architecture supports traditional Ethernet features that are important for future network architectures including VLANs and robust fault tolerant behavior.

Support for Virtual LANs (VLANs) is a traditional Ethernet feature that logically partitions large physical networks into smaller virtual networks. While VLANs share hardware as necessary, layer-two communications is limited to the scope of a single VLAN. This feature improves performance by limiting broadcasts and improves security by limiting the scope of traffic to appropriate links and end stations. The proposed multipath architecture inherits VLAN support when switches that support traditional VLANs are used to construct the network. Figure 11 illustrates a fat-tree that has been logically partitioned into red and green VLANs. Some switches and some links carry traffic that is exclusive to only one VLAN (red or green). Other switches and links carry traffic for both VLANs (gray). The figure illustrates that hash-based multipath L2 routing can preserve traditional VLAN benefits.

A key benefit for hash-based L2 routing is that it behaves, from an external view, as a conventional Ethernet network. As shown in Figure 11, the architecture allows the arbitrary attachment of conventional Ethernet equipment at any of the exposed leaf ports. As with traditional Ethernet, networks can be attached to the hash-based fat tree at multiple points. While multipath routing is supported within the fat tree, redundant links within the conventional portion of the network are used only to support fault tolerance. An example, "Router" is also shown in Figure 11. This router has two points of attachment for each of the two VLANs to the central network. A storage subsystem and conventional switches are similarly attached. These devices all benefit from Ethernet's ability to use spanning tree to reconfigure a network after a hardware failure.

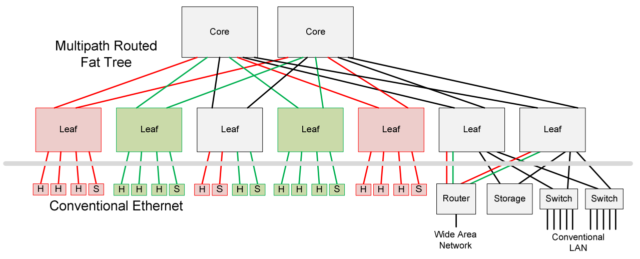

Figure 11: Traditional Ethernet Feature Support

7. Converged Fabric Networks

Important progress has been made in developing and standardizing Ethernet technologies to support the attachment of high-performance storage. Fibre Channel over Ethernet (FCoE) defines a new standard for transporting Fibre Channel traffic while preserving established Fibre Channel management functions. [8]. FCoE uses a collection of low level Ethernet features called CEE (Converged Enhanced Ethernet). CEE includes congestion management, priority scheduling, and per priority pause features needed to support lossless and prioritized storage transport.

Converged fabric networks offer the promise of using a single datacenter fabric for all in-datacenter communications. If CEE functionality is provided in both core and leaf switches, then end-to-end CEE functions can be provided through the network. This provides a powerful and uniform approach for managing LAN and storage traffic within a converged-fabric datacenter.

A key benefit of our use of TCAMs to manage flows is that *type-specific management* can be provided as a natural extension to the multipath load balancing discussed in previous sections. A flow classification module examines the headers of incoming packets and extracts information needed to make routing decisions. Important information includes the type of the packet (e.g. IP vs FCoE), the VLAN, and any other indication of flow type or priority. Flow classification information is used in a subsequent TCAM processing step which supports the non-uniform treatment of specialized traffic classes and assists in providing Quality of Service functions needed for many applications.

Table 3: Example of type-specific network management

Active Cores	Policy match keys		Hashbins allocated	Cores allocated
8	Type==FCoE,	VLAN==*	H⇒ 16	C⇒ 2
	Type==IP,	VLAN==V1	H⇒ 1	C⇒ 1
	Type==IP,	VLAN==*	H⇒ 32	C⇒ 5
7	Type==FCoE,	VLAN==*	H⇒ 16	C⇒ 2
	Type==IP,	VLAN==V1	H⇒ 1	C⇒ 1
	Type==IP,	VLAN==*	H⇒ 32	C⇒ 4
6	Type==FCoE,	VLAN==*	H⇒ 16	C⇒ 2
	Type==IP,	VLAN==V1	H⇒ 1	C⇒ 1
	Type==IP,	VLAN==*	H⇒ 32	C⇒ 3
5	Type==FCoE,	VLAN==*	H⇒ 1	C⇒ 1
	Type==IP,	VLAN==V1	H⇒ 1	C⇒ 1
	Type==IP,	VLAN==*	H⇒ 32	C⇒ 3
4	Type==FCoE,	VLAN==*	H⇒ 1	C⇒ 1
	Type==IP,	VLAN==V1	H⇒ 1	C⇒ 1
	Type==IP,	VLAN==*	H⇒ 16	C⇒ 2
3	Type==FCoE,	VLAN==*	H⇒ 1	C⇒ 1
	Type==IP,	VLAN==V1	H⇒ 1	C⇒ 1
	Type==IP,	VLAN==*	H⇒ 1	C⇒ 1

Table 3 presents an example high-level method for specifying traffic management policies. This method describes a partitioning of available uplinks and core switches among traffic types. This allows, for example, an administrator to specify that specialized traffic such as FCoE must be isolated onto dedicated uplinks. The table describes high-level actions that a central controller should take under a variety of network failure conditions. In the example of Table 3, we assume that a fully functional network provides 8 core switches. Core switches may fail or core switches can be disabled to save power under light load. For each number of active core switches, a set of load distribution rules is specified which a central controller uses to generate detailed TCAM entries for leaf switches. The number of active cores for each scenario is specified in the left-hand column of the Table. As the network status changes, the central controller consults the table to identify the appropriate rules that correspond to the current status.

In this example, we assume that routing decisions are sensitive to the Ethertype, the VLAN, and the hash class in which a packet is classified. Within each row, rules are processed in priority order from top to bottom. For each rule, the match keys match the Ethertype and VLAN tag, and the lookup returns the number of hashbins and number of cores allocated to that class of traffic. The first rule of the table specifies that when 8 cores are active, FCoE traffic should be load balanced in 16 distinct hash classes and sent to 2 dedicated core switches.

For each number of active cores, three rules are specified: one for all FCoE traffic, one for IP traffic on a special high priority VLAN (V1), and one for all remaining IP traffic. The number of allocated core switches within each row total to the number of active switches available for that row. Using this table, low-level software can automatically implement management procedures for a datacenter under a variety of failure conditions.

8. Related Work

Datacenter networks are currently constructed using multiple network technologies. Ethernet dominates low-end networks. The prevalence of Ethernet as the commodity networking solution, and Ethernet's familiarity to network administrators, place Ethernet at the top of the preferred list for most networking solutions.

For many large-scale, high-performance, or time-critical applications, Ethernet does not provide the required functionality. Many technical compute cluster designers were unable to rely on Ethernet due to inadequate scalability and poor performance under congested load. Where Ethernet was inadequate, high-end networks such as Quadrics [9], Myrinet [10], and InfiniBand [11] were employed. These networks have smaller customer bases which rely on less commonly used network architectures targeted for performance-critical applications.

Customized networks for supercomputers were early adopters of scalable networks based on fat trees or other network topologies. These networks provided low level messaging services needed for parallel programming but did not specifically address the needs of low cost and highly scalable Ethernet datacenter applications. [12] [13][14][15][16]

Non-Ethernet vendors including Myricom and Quadrics have proprietary technologies for multi-path L2 routing that are designed to accelerate unordered data transport. These technologies are especially useful for accelerating scalable messaging applications such as those based on MPI. These vendors are now developing and selling Ethernet based networks as Ethernet's base performance approaches that of specialized networks.

InfiniBand is used for many high-end applications and currently offers very cost-effective high bandwidth communications. InfiniBand is scalable to very large scale in a variety of network topologies [10]. However, when InfiniBand is used in conjunction with Ethernet, conversions are required between transport standards. Separate administrative expertise is required for both networks. We believe that Ethernet's marked domination at the low end will stimulate large engineering investments and substantial price decreases to position

Ethernet as a preferred solution even for high-end network solutions.

Hash-based routing can be compared to destination-based or flow-based routing. Techniques that route packets based on destination addresses alone (e.g., as in InfiniBand) limit a network's flexibility to distribute traffic across core switches. In contrast, approaches such as OpenFlow [7] route packets based on flow specifications that can include both source and destination information at MAC and IP layers. Selecting routes based on complete flow specifications which include source and destination pairs presents great difficulties especially for dynamic routing. For networks having a large number of end stations (E), providing routing resources for up to E^2 address pairs can be expensive. A dynamically routed network must provide resources to: measure flow information, calculate routes, and state to set routing choices for a very large number of pairs. Hash-based routing also uses both source and destination information but greatly reduces information needed to monitor and control network traffic for dynamic routing.

Scalable Ethernet networks, based on switch chips from Fulcrum Microsystems, have been deployed for real datacenters [4][5]. Fulcrum switches incorporate a symmetric address hash to allow the construction of scalable fat tree networks for large datacenters. Fulcrum's architecture does not support dynamic load balancing. We cannot determine whether Fulcrum's proprietary technology provides important flexibility in programming the deployment of uplinks. This flexibility is needed to configure a broad range of fat tree networks. The Woven Corporation adds "Vscale" ASICs at the edge of a Fulcrum-based network fabric for dynamic load balancing. This proprietary technology is expensive and requires separate Fulcrum switch ASICs and Woven Vscale ASICs.

Al-Fares *et al.* propose a network architecture that combines fat-tree network topologies with a structured addressing approach to simplify packet forwarding [1]. This approach supports multipath routing, but requires structured addresses that directly indicate a chosen path through the network. This limitation interferes with support for plug and play operation and end station mobility. The attachment of legacy network components at the perimeter of the datacenter is also not addressed. For dynamic routing, the approach relies on the identification of individual flows. A number of difficulties arise when the number of flows exceeds the ability of switches and the central router to maintain and process needed per-flow information.

Multipath routing and congestion avoidance has been explored for Clos networks [18]. Credit based management approaches have been proposed that propagate congestion measures backwards from bottleneck network outputs to earlier decision points within the network. These approaches require that switches generate and process control traffic to propagate needed credit information through a switched fabric. This requires complex per-packet processing that is not a simple upgrade to existing commodity Ethernet switches.

TCAM hardware structures are commonly used for flow classification in IP routers [19]. In a paper by Dong *et al.*, TCAM-like classification is performed using RAMs to process IP packets [20].

Multipath routing has been explored for IP routing, as described in Request for Comment documents from the Internet Engineering Task Force [21] [22]. The constraints of IP routing are different from the constraints of MAC routing, MAC routing needs to consider a larger set of routes and does not have a structured address space.

Prior research also pursues architectures for ultra-large-scale layer two networks. The use of flat layer two networks throughout large enterprises eliminates complex administrative requirements needed to partition large networks. However, for this to be successful, Ethernet enhancements are needed to limit broadcast traffic scope and to support multipath routing [23].

9. Conclusions

Hash-based L2 routing provides a combination of benefits not matched by prior datacenter fabric architectures. Hash-based static load balancing distributes traffic over multiple routes to exploit multipath routing within scalable Ethernet networks. Dynamic load balancing improves performance by identifying traffic bottlenecks and moving traffic to less highly loaded network components. Dynamic balancing also provides a tool to balance performance, reliability, and power for diverse traffic types as resources can be reallocated and core switches can be taken off-line and restored while a datacenter continues to operate. Fault tolerance is supported as fat-tree networks provide redundant connectivity needed for reliable communications.

Load balancing is performed with minimal new state that is independent of the number of end-nodes and the number of end-node connection pairs. This is accomplished while preserving plug and play network operation and the mobility of layer-2 and layer-3 devices. The architecture allows for the flexible attachment of conventional Ethernet hardware. Finally, the solution has been crafted as a modest enhancement to the current layer-2 switch architecture.

Management architectures have been developed and demonstrated that show substantial performance improvement and, in many situations, provide high network efficiency approaching known bounds. We have demonstrated that communication and compute capability needed for central management scales for very large networks commonly used in modern datacenters. This work validates the use of hash-based symmetric routing as a practical approach for managing large scale datacenter networks based on commodity Ethernet switches while preserving traditional Ethernet properties.

10. References

[1] Mohammad Al-Fares, Alexander Loukissas, Amin Vahdat: A Scalable, Commodity Data Center Network Architecture, SIGCOMM 2008.

[2] LAN/MAN Standards Committee of the IEEE Computer Society, ed. (2004), *ANSI/IEEE Std 802.1D - 2004: IEEE* Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges, *IEEE*

[3] C. Clos: A Study of Non-Blocking Switching Networks, Bell Systems Technical Journal, vol. 32, March 1953.

[4] Charles E. Leiserson: Fat-trees: universal networks for hardware-efficient supercomputing, IEEE Transactions on Computers, Volume 34 , Issue 10 (October 1985) , Pages: 892 - 901.

[5] Fulcrum white paper on scalable networks: http: //www.fulcrummicro.com/product_library/applications/clos.pdf.

[6] Fulcrum Microsystems: "FocalPoint Switches in the Datacenter" http://www.fulcrummicro.com /product_library/applications/datacenter.pdf.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. Proc. of SIGCOMM 2008.

[8] Fibre Channel over Ethernet within the T11 standards body: http://www.t11.org/index.htm.

[9] Fabrizio Petrini, Wu-chun Feng, Adolfy Hoisie, Salvador Coll, Eitan Frachtenberg : The Quadrics Network: High-Performance Clustering Technology, IEEE Micro, 22(1), Jan. 2002.

[10] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, and J. Seizovic: Myrinet: A Gigabit-per-second Local Area Network, IEEE Micro, 15(1), 1995.

[11] Introduction to InfiniBand, http://www.infinibandta.org/newsroom/whitepapers/intro_to_infiniband_1207.pdf.

[12] Charles Leiserson, Zahi Abuhamdeh, David Douglas, Carl Feynman, Mahesh Ganmukhi, Jeffrey Hill, Daniel Hillis, Bradley Kuszmau, Margaret St Pierre, David Wells, Monica Wong-Chang, Shan-Wen Yang, and Robert Zak: The Network Architecture of the Connection Machine CM-5, Journal of Parallel and Distributed Computing, 1992.

[13] John Kim, William J. Dally, Dennis Abts: Adaptive Routing in High-Radix Clos Network, SC2006, Tampa Florida, USA.

[14] John Kim, William J. Dally, Dennis Abts: Flattened butterfly: a cost-efficient topology for high-radix networks, Proceedings of the 34th Annual International Symposium on Computer Architecture, pp 126-137, 2007.

[15] John Kim, William J. Dally, Steve Scott, Dennis Abts: Technology-Driven, Highly-Scalable Dragonfly Topology, Proceedings of the 35th Annual International Symposium on Computer Architecture, 2008.

[16] Charles D. Norton and Thomas A. Cwik, Early Experiences with the Myricom 2000 Switch on an SMP Beowulf-Class Cluster for Unstructured Adaptive Meshing, IEEE International Conference on Cluster Computing, 2001.

[17] J. Duato, I. Johnson, J. Flich, F. Naven, P. García, and T. Nachiondo: A New Scalable and Cost-Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks, Proc. IEEE Symp. High-Perf. Computer Arch. (HPCA), San Francisco, USA, Feb. 2005.

[18] Nikolaos I. Chrysos: Congestion Management for Non-Blocking Clos Networks, ACM/IEEE Symp. on Architectures for Networking and Comm. Systems (ANCS'07), December 3–4, 2007, Orlando, Florida, USA.

[19] Karthik Lakshminarayanan, Anand Rangarajan, Srinivasan Venkatachary: Algorithms for Advanced Packet Classification with Ternary CAMs, SIGCOMM'05, August 21–26, 2005.

[20] Qunfeng Dong, Suman Banerjee, Jia Wang, Dheeraj Agrawal, Sshutosh Shukula: Wire speed packet classification without tcams: a few more registers (and a bit of logic) are enough, SIGMETRICS'07, June 12–16, 2007, San Diego, CA.

[21] RFC 2991: Multipath Issues in Unicast and Multicast Next-Hop Selection, http://www.apps.ietf.org/rfc/rfc2991.html.

[22] RFC 2992: Analysis of an Equal-Cost Multi-Path Algorithm, //www.apps.ietf.org/rfc/rfc2992.html

[23] Changhoon Kim, Matthew Caesar, Jennifer Rexford: Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises, SIGCOMM 2008.