A Scalability Study of Enterprise Network Architectures

Brent Stephens, Alan L. Cox, Scott Rixner, T. S. Eugene Ng
Rice University
Houston, TX
{brents, alc, rixner, eugeneng}@rice.edu

ABSTRACT

The largest enterprise networks already contain hundreds of thousands of hosts. Enterprise networks are composed of Ethernet subnets interconnected by IP routers. These routers require expensive configuration and maintenance. If the Ethernet subnets are made more scalable, the high cost of the IP routers can be eliminated. Unfortunately, it has been widely acknowledged that Ethernet does not scale well because it relies on broadcast, which wastes bandwidth, and a cycle-free topology, which poorly distributes load and forwarding state.

There are many recent proposals to replace Ethernet, each with its own set of architectural mechanisms. These mechanisms include eliminating broadcasts, using source routing, and restricting routing paths. Although there are many different proposed designs, there is little data available that allows for comparisons between designs. This study performs simulations to evaluate all of the factors that affect the scalability of Ethernet together, which has not been done in any of the proposals.

The simulations demonstrate that, in a realistic environment, source routing reduces the maximum state requirements of the network by over an order of magnitude. About the same level of traffic engineering achieved by load-balancing all the flows at the TCP/UDP flow granularity is possible by routing only the heavy flows at the TCP/UDP granularity. Additionally, requiring routing restrictions, such as deadlock-freedom or minimum-hop routing, can significantly reduce the network's ability to perform traffic engineering across the links.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Packet Switching Networks*; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Ethernet*

General Terms

Experimentation, Management, Performance

Keywords

Network, Architecture, Enterprise, Scalable, Source Route, Flow

1. INTRODUCTION

IP routers are used in enterprise networks to interconnect Layer-2 subnets. IP routers require complex configuration and maintenance, which can account for as much as 70% of the network operating costs [13]. The high cost of network maintenance can be reduced if the subnets could be larger, removing the need for routers.

Ethernet is currently the most widely used technology in enterprise networking. Unfortunately, it is widely acknowledged that Ethernet does not have the scalability to meet the emerging networking needs of large enterprises [14,18,19,27,28]. Relying on broadcasts prevents scalability by wasting bandwidth and forcing the active forwarding topology to be cycle-free. This distributes forwarding table state poorly across the network and makes it impossible to properly load-balance the traffic on the network.

Nonetheless, because Ethernet is ubiquitous in the enterprise, any attempt at improving the scalability of the Layer-2 subnets will arguably need to remain compatible with Ethernet. In an effort to find a suitable replacement for Ethernet, a large number of alternate network designs have emerged [4,7–11,14,16–18,22,26–28], each having its own specialized strategy for improving the scalability of Ethernet. Some proposals eliminate broadcasts and flooding and reduce the control overhead [7,14,28]. Other proposals reduce forwarding table state [11,16,22,26–28]. Still other proposals increase the scalability of the network by improving the distribution of traffic on the network, either by using a more redundant topology or by changing routing [4,8,9,11,18,22]. Each of these proposals represents a different point in the design space, but there has been no evaluation of the design space as a whole.

Despite the plethora of proposed designs, there have been no concrete comparisons among the designs. The evaluations in these proposals typically only compare the scalability of the proposal directly against standard Ethernet and only for the aspects of scalability for which the proposal addresses, so it has become unclear which proposal is best. When two proposals do evaluate the same aspect of scalability, the evaluations are not directly comparable. Previous prototype evaluations and simulations use differing topologies and traffic patterns, whereas previous analytical evaluations use differing assumptions and estimates. This study evaluates all of the factors that affect the scalability of Ethernet in a unified way in order to better compare the competing proposals.

Each proposal improves scalability by using a unique combination of mechanisms. These mechanisms include routing at the source, routing implicitly, routing at the flow or aggregate granularity, and imposing routing restrictions to reduce state. This paper gives an overview of the proposals to replace Ethernet in a scalable way and provides quantitative comparisons of the behaviors of these techniques under trace-driven enterprise network traffic workloads on realistic topologies. It would not be feasible to evaluate a full implementation of every existing proposal, so this paper identifies orthogonal architectural mechanisms from the commonly proposed networking strategies and evaluates them independently and in conjunction.

The simulations demonstrate that, in a realistic environment, source routing reduces the maximum state requirements of the network by over an order of magnitude while still requiring less overhead to respond to failures. The simulations also demonstrate that load-balancing only heavy flows at the TCP/UDP granularity uses an order of magnitude less state than routing all flows at the TCP/UDP granularity. The distribution of traffic in the network, however, is almost unaffected. The last set of experiments demonstrates that imposing routing restrictions, such requiring deadlock-freedom or minimum length paths, significantly reduces the ability to perform traffic engineering across the links in the networks.

The remainder of this paper proceeds as follows. Section 2 gives a background on the scalability limitations of Ethernet, and Section 3 provides an overview of the proposed Ethernet replacements and their underlying architectural mechanisms. Section 4 discusses the experimental methodology, and Section 5 presents the evaluation for enterprise networks.Next, Section 6 discusses related work. Finally, Section 7 concludes the paper.

2. BACKGROUND

Ethernet is ubiquitous in enterprise networks, primarily because the operation of Ethernet switches is simple and flexible. Ethernet switches require little to no manual configuration. Forwarding is performed based on globally unique identifiers assigned by device manufacturers. Switches learn how to perform forwarding automatically, and failure recovery is fully automatic with the RSTP protocol. Ethernet switches can be wired into any arbitrary topology, and upgrades to the standard retain backward compatibility with existing versions.

However, Ethernet is not scalable. Currently, this problem is solved by building the network out of Ethernet subnets connected by IP routers. The scheme of bridging Ethernet over IP routers is not desirable, even for enterprise networks, because IP routers have many drawbacks.

The biggest problems with routers are their price and that they require complicated configuration and maintenance. Configuration and maintenance of routers imposes substantial costs, estimated in some cases to be as high as 70% of total operational costs [13]. Misconfiguration is a common cause of network failure, even the most common cause on some networks [8].

The number of IP routers may be reduced by increasing the size of Ethernet subnets. However, there are limits to how large a traditional Ethernet subnet may become before its scalability limitations become apparent.

There is a clear need to design a replacement for Ethernet that maintains all of the simplicity and flexibility of Ethernet while solving the scalability problems. Many recent proposals address precisely this issue.

2.1 Ethernet Requirements

Ethernet is ubiquitous in enterprise networks because of its deployment flexitility and management simplicity. Any viable replacement for Ethernet in enterprise networks must retain this flexibility and simplicity, as well as full, transparent compatibility with existing Ethernet hosts.

Maintaining management simplicity requires the network to automatically configure itself with little or no manual configuration. An automatically configured network also has the added bonus of being more scalable and fault tolerant. The human overhead required to design, configure, and manage a network is costly.

Wiring in an Ethernet network is flexible because the network can be configured into arbitrary topologies. As long as there is link connectivity between hosts, they can communicate. Regular networks are difficult to build, maintain, and upgrade. Wiring misconfiguration is a common source of error, and in some enterprise contexts, building a regular network is not even feasible because of geographical constraints. Therefore, a viable replacement for Ethernet in enterprise networks should allow arbitrary network topologies.

This does not mean that regular topologies can not be used for enterprise networks. Regular networks can provide high bisection bandwidth and path redundancy [3], which can improve performance and fault tolerance while reducing cost. However, a regular network topology should not be required for correct operation when misconfiguration and geographical constraints are possible.

Lastly, a replacement must maintain full compatibility with traditional Ethernet. There are many existing devices on enterprise networks that either cannot be reprogrammed or are difficult to reprogram. For example, it would be unacceptable for a campus network to require its users to install a custom driver in order to use the network. Similarly, there may be many legacy devices, including both servers and routers, that would be difficult to reprogram. An replacement for Ethernet in enterprise networks must retain full compatibility with these existing hosts, including both the physical layer and the software stack running on the hosts. This includes support for transparently allowing hosts to use standard Ethernet frames and protocols, such as ARP and DHCP.

2.2 Scalability Limitations

Ethernet achieves simplicity and flexibility by flooding packets in the network. Switches learn the correct egress port for hosts by remembering the ingress port a previous packet from that host used. If a switch has not yet seen a packet from a host or the forwarding table of the switch is full, correct forwarding is achieved by simply flooding the packet.

The direct scalability limitations of flooding are well understood: flooded packets traverse unnecessary links in the network and waste bandwidth. However, flooding imposes other limitations, as well. Flooding restricts scalability by requiring the active forwarding topology to be a cycle-free tree so that flooded packets do not persist indefinitely in the network.

Because Ethernet allows cycles in the physical topology and disables them with RSTP, failures cause additional control overhead. During failures, RSTP consumes usable application layer bandwidth. RSTP requires communication between all of the switches in the network, even if most of the switches are unaffected by the failure.

Forwarding tables need to be implemented with fast hardware to be able to transmit at line rate. For example, forwarding minimally sized packets on a 1Gbps and 10Gbps link at line rate requires that a new forwarding table lookup can commence every 608ns and 60.8ns, respectively. To meet this rate, forwarding tables are typically implemented with content addressable memories (CAMs). A CAM width of 8 bytes is necessary to lookup an Ethernet destination address and VLAN. In order to match a flow, a TCAM with a width of at least 34 bytes is necessary. TCAMs, unlike binary CAMs, can match entries that contain wildcards, which allows flows to be defined on any subset of the protocol headers. A consequence of CAMs is that are of limited size, typically having between 4,096 and 32,768 entries.

In Ethernet, forwarding along a tree means that the switches near the root of the tree require a disproportionate amount of forwarding table state. The switches nearer the root of the tree exhaust the free entries in the forwarding table faster, causing thrashing. Having a sufficiently sized forwarding table is important for scalability because substantial bandwidth is wasted on flooding every time the forwarding working set of a switch is larger than the forwarding table.

Lastly, the topology restricts scalability by preventing traffic engineering. Even if there are redundant links, there is only a single path between any pair of hosts. This eliminates the ability to perform any traffic engineering, which prevents hosts from using all of the available network bandwidth and increasing the network bandwidth by adding extra links.

In summary, the scalability of Ethernet is limited by broadcast and control overhead, forwarding table state, and a lack of ability to perform traffic engineering.

3. PROPOSED DESIGNS

There are many proposed network designs that meet the criteria for replacing Ethernet. Some of these designs were explicitly designed for scalability. The proposals that satisfy the requirements of an enterprise network and their underlying architectural mechanisms are discussed first in this section, after which the proposals that do not are discussed.

Ethane [7] and OpenFlow [17] with Nox address Ethernet's lack of access control through the use of simple switches and a central network controller that implements an access policy. The broadcast overhead problems of Ethernet are solved by integrating some simple services like DHCP and ARP into the controller. The control overhead and forwarding state issues of Ethernet are not addressed. Forwarding in Ethane is performed on the granularity of the flow, which can be defined according to any fields in the packet header, including wildcards. This, combined with allowing arbitrary topologies, allows Ethane and OpenFlow to perform traffic engineering.

SEATTLE [14] assumes that the switches use a link-state protocol to learn the full switch-level topology and compute all pairs shortest paths. The broadcast overhead is eliminated and control overhead is well distributed by performing location and address resolution through a lookup on a DHT maintained across all the switches. SEATTLE does not address forwarding table state, and SEATTLE restricts load balancing by requiring minimal length paths.

Virtual id routing [16], like SEATTLE, proposes using a DHT for address resolution to eliminate broadcast overhead. However, while SEATTLE distributes the switch level topology to all the switches, routing in virtual id routing is restricted to a tree so that the forwarding state requirements are reduced from $\mathrm{O}(N)$ to $\mathrm{O}(\log N)$, where N is the number of switches in the network. Virtual id routing restricts traffic engineering by restricting routing.

TRILL [29] allows for arbitrary topologies to be run with Ethernet by running a link-state routing protocol between the switches and introducing a time-to-live field. TRILL does not eliminate broadcasts or affect forwarding table state, but it does allow for more traffic engineering by allowing for redundant links, although it does restrict routing to minimum length paths.

MOOSE [27] uses location specific addresses within the network. MOOSE reduces the broadcast overhead of Ethernet by requiring a central address resolver, but otherwise suffers from the same broadcast and control overheads of Ethernet. Renaming in MOOSE reduces forwarding state to the switch level topology. Because MOOSE still allows broadcasts, it still suffers from the same topologies restrictions as Ethernet, which prevents traffic engineering.

The Axon [28] is a proposed switch that transparently uses source routing to improve the scalability of Ethernet. Axons eliminate broadcasts by registering all services with a central controller. All of the forwarding state for a host in source routing is located at the local switch, which can reduce forwarding state. The implementation of the Axons described in [28] uses a central controller to register services and install routes and forwards on the granularity of the IP source and destination pair flow. This allows for traffic engineering, although no load balancing mechanisms are proposed.

Converged Enhanced Ethernet (CEE) is a proposed IEEE 802.1 standard from the Data Center Bridging Task Group allows for lossless communication over an Ethernet network [2], primarily for supporting Fibre Channel over Ethernet. CEE enhances Ethernet with mechanisms for assigning bandwidth to 802.1p priority classes and priority flow control. A network that supports lossless communication has low latency, regardless of network load. Also, TCP is unnecessary on lossless networks, which lightens the computational load of hosts and control packet overhead on the network. Eliminating TCP eliminates TCP incast collapse [21], which has a catastrophic affect on the control overhead of Ethernet. Lossless Ethernet has no effect on forwarding table state, and restricts traffic engineering by requiring that routes be deadlock free.

This paper sets strict requirements on dimensions of the network that are required for enterprise networking. Many Ethernet replacement proposals do not meet these requirements, although table 1 shows that they still can be described in terms of the network dimensions described in this paper.

PortLand [22] eliminates the need for any explicit topology information by requiring a strict topology and routing implicitly using location specific addresses, assuming a correct topology. This approach is overly restrictive and difficult to maintain or incrementally deploy. Address rewriting also has the drawback that out-of-band address sharing between hosts will no longer share correct addresses. PortLand is implemented using OpenFlow, and forwards packets at the TCP flow granularity. PortLand is a hop-by-hop and destination routed network with arbitrary routes.

Hedera [4] improves upon PortLand by using link load estimates at the controller to dynamically reroute flows in the network. This achieves near optimal traffic engineering for the workloads and topology that they evaluated. Hedera is a hop-by-hop and flow routed network with arbitrary routes.

VL2 [8] uses location specific addresses within the network to reduce the control overhead generated from host mobility. VL2 also performs traffic engineering with valiant load balancing, which is where every packet in the network is sent to a random intermediate node before being sent towards its destination. VL2 is not considered because it requires manual Layer-3 configuration. VL2 is a hop-by-hop and destination routed network that restricts routes.

SPAIN [18] extends Ethernet to exploit redundant topologies by using a controller to build multiple spanning trees, each on their own VLAN. The spanning trees are built considering network load, and hosts load balance across across the multiple VLANs. SPAIN requires that every host download the switch level topology and VLAN mappings, and hosts are still discovered with broadcasts on one VLAN. Both of these requirements limit the scalability of a SPAIN network. SPAIN requires modifications to end-host software which makes it unsuitable as a generic Ethernet replacement. SPAIN is a hop-by-hop and flow routed network that restricts routes to a set of spanning trees.

HBR [26] is follow up work to SPAIN that reduces forwarding table state by merging flows into ensembles as they enter the network. Forwarding is then performed on the granularity of the ensemble, rather than the flow. The forwarding table state in HBR is constant with the number of ensembles. HBR, like SPAIN, requires end-host modifications. HBR is a hop-by-hop and flow routed network that restricts routes to a set of spanning trees.

DCell [11] improves the scalability of Ethernet for the data center by using a regular topology and integrating the servers into the network. In DCell, the servers, as well as the switches, perform forwarding. Addressing and routing in DCell is based on positioning in the regular topology. This reduces the forwarding state of the switches to be relative to the number of levels in the network, which is logarithmic with the size of the network. Routing in DCell is deterministic and destination based. DCell is a hop-by-hop and destination routed network that restricts routes to minimum length paths.

BCube [9] improves upon DCell by having the servers attach source routes to the packets for forwarding. As in DCell, BCube uses both servers and switches to perform forwarding, and requires a regular topology. Using source routes improves upon DCell by allowing for routing in the network to utilize the multiple paths between any two hosts provided by the regular topology. This allows for load balancing traffic among the links in the network. BCube is a source and flow routed network that supports arbitrary routes.

SecondNet [10] improves upon BCube by allowing for arbitrary topologies. Source routing in SecondNet is accomplished with static MPLS mappings in COTS Ethernet switches. SecondNet uses a central controller for topology dissemination and route installation. SecondNet also addresses user isolation issues in cloud computing networks that arise from having SLAs with many distinct tenants in a data center. SecondNet is a source and flow routed network that supports arbitrary routes.

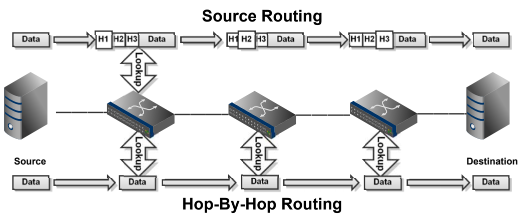


Figure 1: Overview of Packet Forwarding on Source and Hop-By-Hop Networks

DCell and BCube require regular topologies, and DCell, BCube, and SecondNet all require end-host modifications.

3.1 Architectural Mechanisms

From the proposals, we identify three distinct network architecture mechanisms: source routing, flow granularity, and arbitrary routing. A breakdown of the mechanisms of common network architectures can be found in Table 1

Next, we will discuss the three mechanisms that will be evaluated in this paper in more depth.

3.1.1 Source vs Hop-By-Hop Routing

In this study, any architecture where all of the forwarding state for a flow is stored at the local switch is considered source routing. In practice, this is accomplished by the local switch attaching a header that contains the full route to the packet. Axons, BCube, and SecondNet all attach an ordered list of port numbers to the packet. Axons and BCube require custom forwarding hardware, while SecondNet repurposes MPLS support in commodity switches. PARIS [5] decorates packets with an ordered list of link ids instead of port numbers. This study is equally applicable to this style of source routing as well.

Figure 1 gives an overview of packet forwarding for both source and hop-by-hop routed networks. In source routing, the packet contains all of the information necessary to route the packet after the first forwarding table lookup. In contrast, hop-by-hop routing must perform a forwarding table lookup at every hop along the path.

This study assumes that when a source or hop-by-hop route is installed at a switch, only one forwarding table entry is used. This is the case with hop-by-hop routing in OpenFlow. This is also the case with source routing on Axons, where the forwarding table lookup returns an address is SRAM where the full route is stored. Note that with source routing, routes are only installed at the switches local to the source and destination, whereas with hop-by-hop routing, routes are installed at every hop along the path.

Source routing can improve network scalability over hop-by-hop routing. All state needed for a host to communicate is stored at the *local* switch. Therefore, regardless of the size of the network, the state requirements of a single switch are proportional only to the demands of its locally connected hosts. In contrast, hop-by-hop routing requires a switch to store state for every packet that traverses the switch.

If traffic is routed at the flow granularity, it is not pos-

Architecture	Source vs. Hop-by-Hop	Flow vs. Destination	Arbitrary vs. Restrictive
BGP	Hop-by-Hop	Destination (Aggregate)	Arbitrary
OSPF	Hop-by-Hop	Destination (Aggregate)	Restrictive
MPLS	Hop-by-Hop	Flow	Arbitrary
Ethernet	Hop-by-Hop	Destination	Restrictive
MOOSE [27]	Hop-by-Hop	Destination	Restrictive
PortLand [22]	Hop-by-Hop	Destination	Arbitrary
Hedera [4]	Hop-by-Hop	Flow	Arbitrary
VL2 [8]	Hop-by-Hop	Destination	Restrictive
SEATTLE [14]	Hop-by-Hop	Destination	Restrictive
TRILL [29]	Hop-by-Hop	Destination	Restrictive
VIRO [16]	Hop-by-Hop	Destination	Restrictive
OpenFlow [17]	Hop-by-Hop	Flow	Arbitrary
SPAIN [18]	Hop-by-Hop	Flow	Restrictive
HBR [26]	Hop-by-Hop	Flow (Aggregate)	Restrictive
Axon [28]	Source	Flow	Arbitrary
DCell [11]	Hop-by-Hop	Destination	Restrictive
BCube [9]	Source	Flow	Arbitrary
SecondNet [10]	Source	Flow	Arbitrary
Infiniband [12]	Hop-by-Hop	Destination	Restrictive
Myrinet [20]	Source	Flow	Restrictive
Converged Enhanced Ethernet [2]	Hop-by-Hop	Destination	Restrictive

Table 1: Architectural Mechanisms

sible for hop-by-hop routing to have less forwarding table state than source routing; the local switches still need forwarding state as well. If traffic is routed only by destination or multiple flows are aggregated into an ensemble, then sharing can allow hop-by-hop routing to require less forwarding state than source routing.

However, the level of reduction in state is heavily dependent on workload and topology. For example, a destination routed network with an all-to-all communication pattern on a topology with no interior nodes requires exactly the same state on either a source-routed or a hop-by-hop network.

The traditional approach to improve the scalability of hop-by-hop routing is to aggregate multiple routes together in an ensemble. This is the approach taken by IP with CIDR, and recently, this approach has been adapted to existing Ethernet devices [26]. A negative consequence of aggregating lookups is that routing must be performed at the level of the ensemble, so traffic engineering becomes more difficult.

Ensemble routing is complementary to source routing, although it may not be necessary because of decreased state requirements and may not be as effective because ensembles are restricted to flows on the same source switch.

A disadvantage of source-routing is that it requires a larger packet header, which reduces the effective bandwidth available at the physical layer. On Axons [28], the bandwidth is reduced by 8 bits per hop. This is about 0.07% per hop for maximally sized Ethernet packets. SecondNet uses 20 bit MPLS headers per hop for source routing, which reduces bandwidth by about 0.175% for maximally sized packets.

Hop-by-hop routing has the potential to react quicker to network events, such as load and failures, than source-routing. This is because, in hop-by-hop routing, the intermediate switches can independently make routing decisions based on local knowledge. In source routing, the route is fixed at the source. When a failure occurs in hop-by-hop routing, the switch local to the failure can immediately use a precomputed alternate route, whereas source routing requires that the source switch be notified of the failure, after which it can use a precomputed alternate route, if it exists. This bounds the improvement in response time to failure to a single RTT. If alternate routes are not precomputed, then response time to failure is dominated by the time spent computing a new route.

The actual method by which the routes are computed is orthogonal to the dimensions discussed in this paper. Computing routes does not depend on whether the route will be used as a source or hop-by-hop route. Route computation can either be centralized or distributed. A central controller can use global knowledge of the network to compute routes. This is the approach taken by [4,17,18,22]. There are many distributed algorithms for computing routes. In TRILL [29], switches use a link state routing protocol to learn full knowledge of the topology, which can be used to compute routes. SEATTLE [14] uses a link state routing protocol to learn the full witch topology and uses a DHT for IP address and switch location resolution. VIRO [16] builds and routes along an overlay tree and also uses a DHT for address resolution. An evaluation of this network dimension is left for future work.

The costs of installing routes in a centralized architecture and computing and installing routes in a distributed architecture are proportional to the forwarding table state. In a centralized architecture, the cost of installing routes includes the bandwidth used to send forwarding table state to the switches and the processing power used by the switch to install the state. In a distributed architecture, every switch that requires forwarding table state for a route must independently compute and install forwarding table state.

3.1.2 Flow Granularity

Flows can be defined at many different granularities, from destination only flows to IP pair to the 4-tuple of source and destination TCP port and IP address. Using a finer flow granularity allows for finer granularity traffic engineering and security policy at the expense of adding extra overhead. A finer granularity is guaranteed to increase routing state. If routes are not computed where they are installed, the network overhead of installing routes also increases. The increase in overhead is proportional to the flow granularity.

There are cases where extra traffic engineering and security can be beneficial for an enterprise network. For example, Pang, *et al.* [23] showed that on the LBNL enterprise network, between 66–95% of total bytes in the network are carried by 8–26% of the connections. If the traffic is engineered so that the heavy flows are path disjoint from the rest of the traffic, even between the same IP addresses, the other traffic would see a latency reduction and throughput increase.

Although enterprise networks are typically contained in a single administrative domain, there are still security concerns. For example, there may be distinct user groups that should be isolated, like students, staff, and researchers at a university. Typically, this isolation is achieved with VLANs, but each VLAN requires its own buffer resources. Enforcing isolation by installing routes at the flow granularity allows a trade-off between device buffer size and forwarding table entries.

However, routing state per switch is finite and a limiting factor on the total size of a network. If too fine a flow granularity is used, forwarding tables will become full and active flows will either have to be removed or requests for new flows will be rejected.

3.1.3 Arbitrary vs Restrictive Routing

Arbitrary routing can more flexibly perform traffic engineering, but some networks choose to restrict routing to either improve performance or reduce forwarding state. Networks such as OSPF or DCell restrict forwarding to minimum length paths to simplify path computation. Virtual id routing [16] restricts forwarding to a spanning tree in order to logarithmically reduce forwarding table state. Lossless Ethernet [2] requires that routes be deadlock-free in order to reduce TCP control overhead, guarantee low latency, and require minimal buffering.

A deadlock occurs when one or more packets get blocked forever because a set of resource requests (*e.g.*, packets traversing the network) form a cycle. The most popular deadlock-free routing algorithm is Up*/Down* [24]. Up*/Down* routing creates a spanning tree of the network and assigns directions to each link based on the spanning tree so that the directed links do not form loops. A valid route never uses a link in the up direction after using a link in the down direction, so there are never any routes that form a loop, and the routes are guaranteed to be deadlock-free.

Up*/Down* routing is less restrictive than the spanning tree imposed by Ethernet and Virtual id routing, and it allows for all working links to be used, although it does impose some routing restrictions. How the spanning tree is built can impact the number of routing restrictions imposed. The original description of Up*/Down* routing builds a BFS spanning tree [24]. Shancho, *et al.* [25] showed that using heuristics to build a DFS spanning tree can reduce the number of routing restrictions by 20%.

4. METHODOLOGY

For all of the evaluations, the following scalability metrics are used: control overhead, forwarding table state, and link bandwidth distribution. Broadcast overhead is ignored because broadcasts can be replaced with either a lookup on a DHT [14] or a request to a central controller [7]. Link bandwidth distribution reflects on both the path diversity of the topology and the ability of the architecture to perform traffic engineering.

A custom simulator was developed to perform the evaluations. The simulator is an extended version of the simulator used in [28]. The simulator is comprised of four main components: switch control plane software, the controller software, a switchboard process, and the host software. The simulations are run with real-world topologies and two different sets of network packet traces: one collected from the Lawrence Berkeley National Lab (LBNL) and one from university data centers (UNIV).

The switch control plane software and controller software are identical to the software that would run on the switches and network controller in a prototype network, respectively. However, rather than connecting to the data plane of a switch, the software interfaces with the switchboard. One instance of the control plane is created for each simulated switch, but only one switchboard is created. The switchboard simulates all of the links and switch hardware in the network and is responsible for forwarding packets and keeping statistics. The host software can either be a virtual machine or a packet generator. Arbitrary traffic can be forwarded through the simulator between virtual machines, while the packet generator is used as a lower overhead means to inject traffic.

The packet generator is used to inject traffic into the network, which causes the controller to install flows. The generator sends TCP packets for TCP flows and ARP packets for all other flows. The traffic is injected in chronological order and is derived from packet traces collected from the Lawrence Berkeley National Lab (LBNL) [23] The LBNL traces have scanning traffic filtered out because the controller has a global view of traffic in the network and could detect and disable ports engaged in scanning. Further, the scanning traffic in the trace is anonymized separately from other traffic, so a host engaged in scanning appears in the traces as two separate hosts.

There are five LBNL traces, each with a duration from 10 minutes to 1 hour, collected over a period of four months. The traces contain between 4500–6000 active internal hosts. Although the traces were collected over a wide time frame, they are limited to a single site. The LBNL traces are collected at routers, so the traces provide a view of the traffic in the core of an Enterprise network. More information about the traces can be found in [23].

Although full network traces are used to derive the traffic injected by the packet generator, the packet generator does not inject every packet in the traces to reduce simulation time. Instead, the traces are preprocessed to calculate the bandwidth of each flow in the network. Then, only two packets that are decorated with bandwidth information are sent for each simulated flow. The first packet is sent when the flow starts with the average bandwidth of the flow, and the second packet is sent when the flow ends with a bandwidth of zero so that the controller removes the flow from the network. The switchboard never disallows flows from the network, which effectively simulates an infinite bandwidth network.

A centralized network controller was chosen instead of a distributed control protocol for ease of implementation, and the results for the experiments, other than the failure experiment, would be identical for either architecture.

The switch control plane software and the controller were validated by controlling an Axon network built with NetFPGAs. The switchboard was similarly validated by connecting to the Axon network and sending traffic from real switches through the switchboard. The state and bandwidth distributions for simple traces and networks were computed by hand and used to verify the output of the simulator.

Unless otherwise noted, the controller uses a weighted shortest path algorithm where the weight of each link is one plus the bandwidth of current traffic on the link in bytes. The traffic is injected with the bandwidth it used in the traces, and the WSPF algorithm will distribute this bandwidth across the links in the network and try to minimise the maximum bandwidth. Routes are computed greedily in the order they are requested in the traces.

The network topology simulated by the switchboard is read from a configuration file. For the LBNL traces, the canonical Cisco 3-tier topology (TIER) is used with an oversubscription ratio of 20:1 between the access and distribution layers and 4:1 between the distribution and core layers, as recommended in [1]. Each edge switch connects to 40 hosts and has two uplinks to the distrubtion layer. All the switches in the distribution and core layers, respectively, are connected in a torus. Other topologies were simulated, but the results were similar, and are omitted for brevity.

The topologies use only the number of switches that are required for the simulated number of hosts. Furthermore, the hosts are evenly distributed across the switches. This means that the bisection bandwidth of the network is kept constant. The LBNL traces contain traffic between internal and external hosts. In this case, the communication is simulated as being between the internal host and a border router. In the TIER topologies, the border routers are attached to the core of the network. For each topology, 5 different random mappings of hosts to switches were simulated and the results were averaged across the runs.

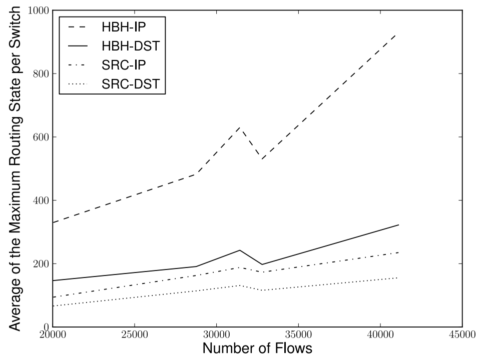
5. RESULTS

In this section, the architectural mechanisms are evaluated on a realistic network. The experimental data presented demonstrates that source routing requires an order of magnitude less state than hop-by-hop routing while being better suited to react to failures by requiring less control overhead to respond per failure. Simulations also show that the same level of traffic engineering that is possible by routing all flows at the TCP-flow granularity is achievable with an order of magnitude less state by routing only heavy flows at the TCP granularity. Lastly, experiments demonstrate that restricting routing affects the network's ability to distribute bandwidth and state across the network. Carefully building an Up*/Down* tree with topological knowledge imposes the least restrictions, although the restrictions are noticeable, followed by restricting routing to minimum hop-count paths. On the other hand, the restrictions imposed by topology oblivious Up*/Down* trees are more significant.

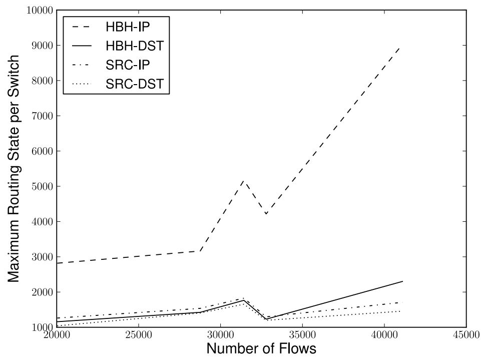
5.1 Reduced Forwarding State

The state requirements of hop-by-hop and source routing was computed from the set of flows installed after packet injection. Although flows are installed and removed during the experiment, for each switch, only the maximum state at any point during the trace is presented because overflowing the CAM is the only way for forwarding table state to affect scalability.

The results of the computation for different routing schemes on the 3-tier topologies sized by bisection bandwidth with two uplinks are presented in Figure 2. The dip in number of flows installed around 32,000 flows is because the LBNL trace with around 32,000 flows has less hosts in the network and thus uses few switches than the LBNL trace with around 31,000 flows because of how the topologies were sized.



(a) Average Routing State



(b) Maximum Routing State

Figure 2: Per-Switch State by Routing Type

HBH-IP is for hop-by-hop routing on the granularity of an IP pair. This is similar to how OpenFlow [17] is commonly used. *HBH-DST* is for destination based hop-by-hop routing. Because multiple routes are installed for each destination, this is most similar to using ECMP in the network. *SRC-IP* is for source routing at the IP pair granularity, and *SRC-DST* is for destination based source routing. Destination based source routing is where all hosts on the same switch use the same route for a destination. The source routing data is similar to two different configurations of an Axon network [28].

The 3-tier topology has a small number of switches in its core, allowing for greater sharing for the *HBH-DST* experiment. The LBNL traces only contain traffic between subnets and the hosts are clustered by subnet in the topologies, so there is significant potential for sharing in the network.

Figure 2(a) shows the average of the maximum routing state per switch for different routing schemes. Destination based hop-by-hop routing requires 100% more state than IP pair source routing, and IP pair hop-by-hop routing requires at least 250% more state than IP pair source routing.

Figure 2(b) shows the maximum routing state per switch.

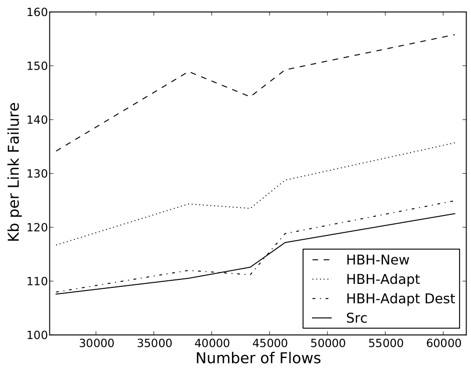


Figure 3: Failure Overhead for Routing Types

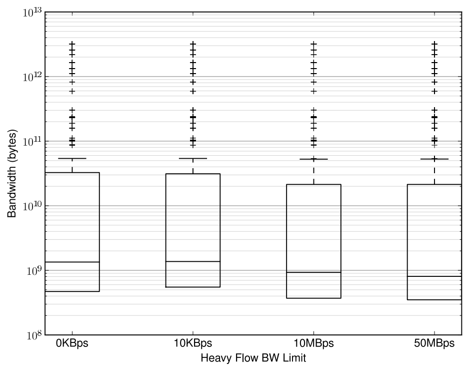


Figure 4: Box Plot of Maximum Bandwidth per Link by Flow Granularity

The maximum routing state for all routing types other than IP pair hop-by-hop routing are almost identical, although hop-by-hop routing still requires more state than source routing. This is because if a single host, such as a default gateway, communicates with a large number of hosts, the switch local to this host must have state for each one of these flows. IP pair hop-by-hop routing has a much higher maximum because flows traversing the worst switch require their own state and cannot reuse the state installed by the host with a high degree of communication.

The data shown in Figure 2(b) shows that even for a network with only 30,000 flows that were generated by 5,000 hosts, hop-by-hop routing requires more state than can fit in all but the largest CAMs. If the topology is changed so that each subnet is half as big and only has one uplink to the core, the maximum routing state increases by 10,000 entries in the worst case for IP pair hop-by-hop routing. This causes the state requirements to be near the maximum sized CAMs available. The state requirements for source routing are unaffected by this change.

5.2 Failure Overhead

Failures should be expected on a network. On a centrally controlled network, the network overhead of per failure consists of failure notification and flow rerouting. The overhead for failure notification should be constant between different network architectures, but the overhead for flow rerouting is dependent on the routing architecture. Failure simulations were performed to evaluate the differences between using a central controller to reroute on a source-routed network and a hop-by-hop network. OpenFlow is a centrally controlled hop-by-hop network, so rerouting on an OpenFlow network is similar to the rerouting scheme used in this experiment.

Failures were performed during the period in the trace with the most active flows. There is no rerouting to be performed if a link connected to a host fails, so only links between switches were failed. Links were selected for failure with uniform probability. Only a single link was failed at a time, and the network was restored to its original state after each failure. Multiple route updates destined for a single switch were grouped into a single packet to reduce header overhead. 5,000 failures were performed per simulation, and the results are averaged across multiple runs.

Single link failure is the lowest overhead case for hop-by-hop routing. In hop-by-hop routing, a packet must be sent to each switch that did not previously have a route along the updated path, and the length of the updated path is likely to increase with the number of failures. Regardless of the size of the failure, only one packet needs to be sent to reroute a unidirectional flow on a source-routed network.

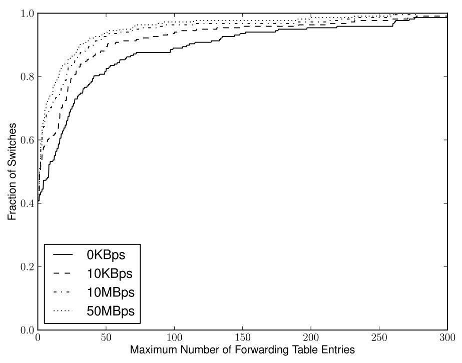
The TIER topology has oversubscription, which skews the distribution of state in the network. To avoid this bias, the failure experiment was performed on a flattened butterfly topology (FBFLY) [15].

Figure 3 presents the results of the failure simulations. This figure shows the average total control overhead in Kb for a single link failure as the number of flows installed on the network increases. This overhead includes the control traffic sent between the controller and switches for both failure notification and flow rerouting. *HBH-New* represents installing a new hop-by-hop route for each IP-pair flow. *HBH-Adapt* represents rerouting the existing routes around the failed link for each IP-pair flow. *HBH-Adapt Dest* represents rerouting the existing routes around the failed link for each destination address. *Src* represents rerouting IP-pair flows on a source-routed network.

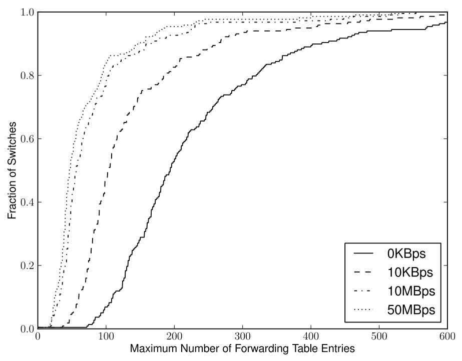
The packets sent to each hop-by-hop switch are smaller than those sent to a source-routed switch. Only a single hop is sent to a hop-by-hop switch while the full path is sent to a source-routed switch. However, multiple switches need to be updated for hop-by-hop routing. The overhead is highest for *HBH-New* because it ignores some of the existing network state. *HBH-Adapt* performs better, but on the FBFLY topology, the adaptive route around a single link failure is two hops long, so the overhead is still greater than *Src*. *HBH-Adapt Dest* performs the best of all of the hop-by-hop routing schemes, but, on average, there are not enough flows on a link that share the same destination for it to outperform *Src*.

5.3 Flow Granularity

The per switch state requirements are known to increase as the granularity of flows becomes finer, but it is not obvious whether making the granularity of the flow finer allows for better load balancing across the network. One way to trade off state and time complexity of routing for ability to load balance traffic is to only route the heavy flows at the transport layer granularity. To understand how flow granularity affects state and load balancing, an experiment was run with



(a) Source Routing



(b) Hop-By-Hop Routing

Figure 5: CDF of Per-Switch State by Flow Granularity

multiple parsings of one of the LBNL traces where each parsing used a different bandwidth limit for heavy flows.

Because of the increased complexity in simulating at the TCP granularity, this experiment was run using only a subnet from the LBNL traces that had 628 hosts. The TIER topology only had one switch in the core of the network with recommended oversubscription ratio, so a k = 14 fat tree topology was used instead. The topology used 14 ports per switch, and was build as described in [22].

Figure 4 presents a box plot of maximum bandwidth per link. The labels 10KBps, 10MBps, and 50MBps on the x-axis are the bandwidth limits after which an IP flow was classified as a heavy flow and all of the individual transport flows were routed independently. The bandwidth limit of 0KBps means that all of the flows were heavy and routed at the TCP granularity. Even on a non-blocking topology, there is not a substantial difference in bandwidth between the differing heavy flow limits.

Figure 5 shows a CDF of state per switch given the same heavy flow limits used in the previous figure. All of the flow limits require a 34B/entry TCAM. Figure 5(a) shows the routing state per switch with source routing, and Figure 5(b) shows the state with hop-by-hop routing. These figures show that the state reduction is substantial between the different heavy flow limits. This implies that it is best to route at the IP-flow granularity because the state reduction is substantial and the traffic engineering gain is minimal.

Routing Type	Heavy Flow BW Limit			
	0KBps	**10KBps**	**10MBps**	**50MBps**
Source	631	524	277	261
Hop-By-Hop	1268	1052	554	532

Table 2: Maximum Number of Forwarding Table Entries per Switch for Source and Hop-By-Hop Routing by Flow Granularity

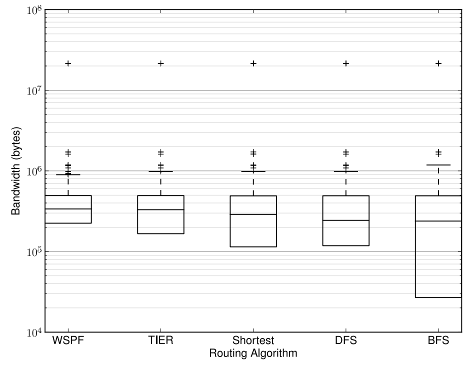


Figure 6: Box Plot of Maximum Bandwidth per Link for Routing Algorithms

Figure 5 also reaffirms the conclusions of Section 5.1. Table 2 show the maximum state per switch for the different heavy flow limits, which further demonstrates the state reduction from source routing.

The control overhead by granularity will increase proportionally with state with a central controller, so control overhead is not presented for this experiment.

5.4 Routing Algorithm Restrictions

Typical lossless networks impose routing restrictions to prevent deadlock. Three different implementations of the deadlock-free Up*/Down* routing were compared against unrestricted routing. One implementation of Up*/Down* routing, described in [24], uses a BFS spanning tree. The second implementation, described in [25], uses heuristics to build a DFS spanning tree, which can lead to fewer path restricts than a traditional BFS tree. The root of both the BFS and DFS spanning tree was chosen as one of the core switches. The third implementation, TIER, assigned each switch to its level in the TIER topology, then assigned directions based on levels, breaking ties by comparing device ids.

Other networks, such as TRILL, SEATTLE, DCell, and VIRO require that only minimum hop count routes be used, so shortest path routing was also used for comparison.

A boxplot of the maximum bandwidth per link for five different routing algorithms on the TIER topology is presented in Figure 6. *WSPF* is the unrestricted weighted greedy shortest path first algorithm, and *Shortest* restricts routing to shortest hop-count paths. *BFS* and *DFS TIER* represent Up*/Down* routing using a BFS, DFS, and TIER spanning tree, respectively. The box extends from the lower to upper quartile values of the data with a line at the median.

Figure 7 shows a cdf of the maximum bandwidth per link for a single run on the same topology. Although only one

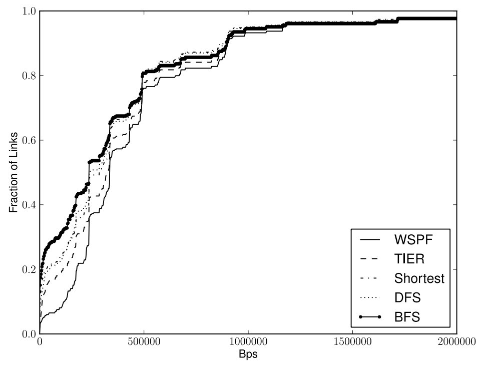


Figure 7: CDF of Maximum Bandwidth per Link for Routing Algorithms

run is presented, other runs had similar distributions.

All four algorithms assign routes using the greedy weighted shortest path algorithm, after routing restrictions have been considered, to distribute flows over links. The restrictions imposed by the routing algorithm can be seen by comparing against the WSPF. The more routing restrictions are imposed, the wider the distribution of bandwidth will be. The unrestricted WSPF routing can incur some path stretch, which causes its average bandwidth per link to be higher, but the algorithm is not meant to optimize for minimum bandwidth per link or fitting in the capacity of the link. Instead, the WSPF algorithm optimizes towards having the active flows path disjoint and distributing bandwidth evenly across the links.

The WSPF algorithm distributes the load the best. The TIER Up*/Down* algorithm performs the next best. This is because the path restrictions imposed by TIER routing are on the toroidal links at the aggregation and core levels and some paths longer than the minimum length, leaving most paths unaffected. Although the shortest path and DFS Up*/Down* algorithms look similar in the boxplot, Figure 7 shows that almost 20% of the links in the network are unused by shortest path routing while less than 5% of the links are unused by the DFS Up*/Down* routing. In contrast, less than 0.5% of the links are unused by the unrestricted WSPF routing.

[24] accurately claims that Up*/Down* routing allows for every single link to be used, and similarly, shortest path routing can also use every link. However, Figure 7 shows that under an enterprise traffic pattern, not all of the links in the network are fully utilized or with both Up*/Down* routing and shortest path routing.

6. RELATED WORK

SEATTLE [14] explores improving network scalability on an enterprise network. However, the focus of SEATTLE is primarily on presenting a novel DHT-based architecture and evaluating it as a replacement for Ethernet. The SEATTLE prototype was implemented using Click and XORP, so it used the network dimensions of those platforms: hop-by-hop, best-effort, and destination routing. This work is unique in that it evaluates the characteristics of the underlying switch architecture. Note that the main contribution of SEATTLE is the DHT directory service, which is complementary to the network dimensions discussed in this paper.

Bondi [6] builds a general system scalability framework. The framework developed in this work identifies subcategories of the load and space categories presented by Bondi. The structural category of scalability is ignored because address scalability limitations are easily solved by increasing the number of bits used for addressing.

Myers, *et al.* [19] address the scalability of an Ethernet network to a million nodes. In their work, they identify the limitations of packet flooding and broadcasting. This is only one of the five scalability factors identified by this work.

Shancho, *et al.* [25] evaluate the decrease in path diversity associated with Up*/Down* routing. However, their work only compares the change in path diversity between a BFS and DFS spanning tree and different heuristics for building the trees. They do not compare against unrestricted routing. Also, their evaluation lacks any real-world traces, and the largest topology used only has 64 nodes, which is not enough for an evaluation of the algorithm at scale.

Ethane [7] estimates forwarding table sizing for hop-by-hop routing. Ethane does not perform any experiments at scale to determine forwarding table size. They extrapolate from the state requirements of a 300 host network that a switch with a fanout of 64 on a university network requires approximately 8,192–16,384 forwarding table entries, but this is a rough estimate. Shafer, *et al.* [28] estimate the forwarding table size for source routing on the LBNL network by calculating reuse distance.

The forwarding table size estimates from previous work are not obtained from simulations, and they cannot be compared to each other. This paper is unique in that it compares source and hop-by-hop routing directly against each other.

7. CONCLUSION

While the recent proposals that address the scalability of Ethernet compare themselves to Ethernet, it is unclear from the proposals which techniques are best. The primary contributions of this work are the identification of the factors that affect Ethernet scalability, the identification of three distinct mechanisms for addressing scalability, and the real-world, trace driven evaluation of these mechanisms in the context of Ethernet scalability and enterprise networking. The results of the evaluation show that a source-routed, IP flow-managed network with unrestricted routing is the best suited network for the enterprise.

Hop-by-hop routing requires at least 100% more forwarding state than source routing on enterprise workloads and topologies. This also holds true for data center workloads and topologies, although the results are omitted for space. This implies that future scalable Ethernet architectures should focus on using source routing to reduce forwarding table pressure. While Hop-by-hop routing can respond to failures quicker than source routing, the improvement is bounded to within a single RTT, which is insignificant compared to the state reduction of source routing.

On networks that can support routing on a regular network, forwarding table state is proportional to the number of ports on the switch. This is better than both source and hop-by-hop routing. However, exception flows are still necessary on a such networks, either for QOS, performance, or routing through middleware applications. Exception flows, by definition, cannot use the same forwarding table entries as the default routes. The results in this study imply that it is more scalable to use source routes instead of hop-by-hop routes, regardless of whether the routes are default or exception routes.

Other work has shown that traffic engineering individual flows can achieve near-optimal throughput and outperforms static traffic engineering, such as ECMP [4]. This paper demonstrates that, on enterprise workloads and topologies, routing all individual flows at the TCP granularity does not allow for better traffic engineering than routing only heavy flows at the TCP granularity.

Restricting routing, such as requiring deadlock freedom or minimum hop-count paths, has a noticeable affect on traffic engineering. The poor performance of the BFS Up*/Down* routing scheme implies that it should never be used. If CEE is necessary in the network for Fibre Channel support, using a tree topology and building deadlock-free routes with topological knowledge gives the best performance, relatively close to unrestricted routing. If building a tree topology is not feasible, using DFS Up*/Down* routing to support CEE causes 10% of the links in the network to be unused, in practice. However, deadlock freedom of any kind still affects performance, so any flows that do not require lossless delivery should not be restricted to deadlock-free routes.

Architectures that restrict routes to minimum length paths also affect network scalability by reducing path diversity in the network. This has no affect on non-blocking topologies, but enterprise networks are typically oversubscribed. Unrestricted routing allows flows to be more evenly distributed in the network, so enterprise network architectures should allow for arbitrary paths.

8. REFERENCES

[1] Cisco campus network for high availability: Design guide. http://tinyurl.com/d3e6dj.

[2] Ieee 802.1 data center bridging task group. http://www.ieee802.org/1/pages/dcbridges.html.

[3] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu. Energy proportional datacenter networks. In *ISCA*, 2010.

[4] M. Al-fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *NSDI*, 2010.

[5] B. Awerbuch, I. Cidon, I. Gopal, M. Kaplan, and S. Kutten. Distributed control for PARIS. In *PODC*, 1990.

[6] A. B. Bondi. Characteristics of scalability and their impact on performance. In *WOSP*, 2000.

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

[8] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network. In *SIGCOMM*, 2009.

[9] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: A high performance, server-centric network architecture for modular data centers. In *SIGCOMM*, 2009.

[10] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. SecondNet: a data center network virtualization architecture with bandwidth guarantees. In *Co-NEXT*, 2010.

[11] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. DCell: A scalable and fault-tolerant network structure for data centers. In *SIGCOMM*, 2008.

[12] T. Hamada and N. Nakasato. InfiniBand Trade Association, InfiniBand Architecture Specification, Volume 1, Release 1.0, http://www.infinibandta.com. In *International Conference on Field Programmable Logic and Applications, 2005*, pages 366–373.

[13] Z. Kerravala. Configuration management delivers business resiliency. 2002.

[14] C. Kim, M. Caesar, and J. Rexford. Floodless in SEATTLE: A scalable Ethernet architecture for large enterprises. In *Proceedings of ACM SIGCOMM*, 2008.

[15] J. Kim and W. J. Dally. Flattened butterfly: A cost-efficient topology for high-radix networks. In *ISCA*, 2007.

[16] G.-H. Lu, S. Jain, S. Chen, and Z.-L. Zhang. Virtual id routing: a scalable routing framework with support for mobility and routing efficiency. In *MobiArch*, 2008.

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, 2008.

[18] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul. SPAIN: COTS data-center Ethernet for multipathing over arbitrary topologies. In *NSDI*, 2010.

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

[20] Myricom. Myri-10G NICs and software, Aug. 2008. Product brief.

[21] D. Nagle, D. Serenyi, and A. Matthews. The Panasas ActiveScale Storage Cluster: Delivering scalable high bandwidth storage. In *SC '04: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing*, 2004.

[22] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. PortLand: a scalable fault-tolerant layer 2 data center network fabric. In *Proceedings of ACM SIGCOMM*, 2009.

[23] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *IMC*, 2005.

[24] T. L. Rodeheffer and M. D. Schroeder. Automatic reconfiguration in Autonet. In *Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles*, 1991.

[25] J. Sancho and A. Robles. Improving the Up*/Down* routing scheme for networks of workstations. In *Euro-Par 2000 Parallel Processing*, volume 1900 of *Lecture Notes in Computer Science*, pages 882–889. Springer Berlin / Heidelberg, 2000.

[26] M. Schlansker, Y. Turner, J. Tourrilhes, and A. Karp. Ensemble routing for datacenter networks. In *ANCS*, 2010.

[27] M. Scott, A. Moore, and J. Crowcroft. Addressing the scalability of Ethernet with MOOSE. In *Proceedings of DC CAVES Workshop*, 2009.

[28] J. Shafer, B. Stephens, M. Foss, S. Rixner, and A. L. Cox. Axon: a flexible substrate for source-routed Ethernet. In *ANCS*, 2010.

[29] J. Touch and R. Perlman. Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement. RFC 5556 (Informational), May 2009.