Controlling Traffic Ensembles in Open Cirrus

Wenjie Jiang, Yoshio Turner, Jean Tourrilhes, Mike Schlansker

HP Laboratories
HPL-2011-129

Abstract:

Datacenter network architectures must change to accommodate the unprecedented scaling and low cost requirements of cloud computing environments. This paper describes our work on managing the performance of multi-path Ethernet fabrics spanning a large datacenter. Our approach is to develop an Ensemble Routing Controller, which achieves management scalability by reasoning about traffic at the granularity of a tractable set of flow ensembles, i.e. groups of flows. To support research on novel datacenter network architectures and management frameworks, the Palo Alto Open Cirrus site has been augmented with an experimental multi-path Ethernet network, which operates separately and in parallel with the production Open Cirrus network fabric. We implemented and deployed a prototype Ensemble Routing Controller to manage traffic in this multi-path network. We present preliminary experimental results showing that our controller effectively manages the fabric resources by dynamically assigning traffic to efficient network paths delivering close to optimal network goodput to applications. We also describe our experiences using the Open Cirrus platform and suggest features for Open Cirrus resource allocation that would further facilitate our research on datacenter networking.

External Posting Date: August 21, 2011 [Fulltext] Approved for External Publication
Internal Posting Date: August 21, 2011 [Fulltext]

Controlling Traffic Ensembles in Open Cirrus

Wenjie Jiang
Princeton University
Email: wenjiej@cs.princeton.edu

Yoshio Turner Jean Tourrilhes Mike Schlansker
HP Labs
Email: {yoshio.turner,jean.tourrilhes,mike_schlansker}@hp.com

***Abstract*—Datacenter network architectures must change to accommodate the unprecedented scaling and low cost requirements of cloud computing environments. This paper describes our work on managing the performance of multi-path Ethernet fabrics spanning a large datacenter. Our approach is to develop an Ensemble Routing Controller, which achieves management scalability by reasoning about traffic at the granularity of a tractable set of flow ensembles, i.e. groups of flows. To support research on novel datacenter network architectures and management frameworks, the Palo Alto Open Cirrus site has been augmented with an experimental multi-path Ethernet network, which operates separately and in parallel with the production Open Cirrus network fabric. We implemented and deployed a prototype Ensemble Routing Controller to manage traffic in this multi-path network. We present preliminary experimental results showing that our controller effectively manages the fabric resources by dynamically assigning traffic to efficient network paths delivering close to optimal network goodput to applications. We also describe our experiences using the Open Cirrus platform and suggest features for Open Cirrus resource allocation that would further facilitate our research on datacenter networking.**

I. Introduction

Datacenter networks for cloud computing environments need to scale to thousands of physical machines, and must be cost-effective to retain the economic benefits of cloud computing. Current Ethernet networks are hard to scale to large size because of the single-path constraint of the Spanning Tree Protocol, and the consequent use of very expensive high port count core switches. Thus, researchers have proposed multi-pathing techniques to scale Ethernet datacenter networks [1], [2], [3], [4], [5], [6].

The choice of multiple paths raises the need for smart traffic management to choose paths that optimize network performance. In previous work, some of us proposed *Ensemble Routing* as an approach to realize the combination of multi-path Ethernet and scalable management [7], [6]. The key insight of Ensemble Routing is to use state compression to efficiently and scalably manage a large network hosting a tremendous number of flows coming and going over time. Instead of managing each flow individually, which seems intractable, Ensemble Routing operates at the granularity of flow ensembles, or collections of flows. Each flow, identified by packet header n-tuple, is classified into a *traffic class*, which determines the QoS treatment of the flow, and *hash class*, which is calculated based on a simple symmetric hash function. Together, the traffic and hash class comprise the *routing class*, which determines the route that packets of a flow will take through the network. Each switch records statistics for each routing class and provides the statistics to a logically centralized controller. The controller gathers per-routing class statistics from each switch, and then executes an optimization algorithm to map each routing class to one of multiple routing networks providing path diversity.

Ensemble Routing offers to globally optimize traffic usage in the network through load balancing. Ensemble Routing also offers great flexibility to implement network management policies. For example, it can provide physical isolation or dedicated paths to a traffic class with guaranteed bandwidth, by allowing an administrator to restrict traffic classes to use specific sets of paths. It can also tune the granularity of traffic management by allowing an administrator to adjust the number of hash classes used for each traffic class. Furthermore, Ensemble Routing enables fast failure recovery, as the network controller can rapidly shift traffic away from failed network components without waiting for a complex distributed recovery protocol to converge.

This paper presents a prototype implementation of Ensemble Routing on the HP Labs Palo Alto Open Cirrus testbed. Our prototype uses host-based virtual switches to implement multi-pathing. Using host-based virtual switches may be suitable for production deployment in a cloud computing environment that controls the system software (e.g., Xen hypervisor) on all physical hosts. Alternatively, our implementation based on virtual switches can be viewed as a software emulation of future deployments that could use enhanced physical switches.

We present experimental results showing that our Ensemble Routing Controller can manage multi-path traffic to load balance network resources. We study two traffic patterns with very different optimal routing policies, and we show that our controller automatically and quickly discovers a routing policy that is near-optimal in each case. We show that dynamically changing the routing policy, causing a flow to shift between network paths, is unlikely to cause a significant performance disruption, for example due to TCP retransmissions caused by packet reordering.

Our experiments use a special research test network on the Open Cirrus testbed. We discuss some implications of conducting cloud datacenter network-related research on the shared Open Cirrus testbed and suggest possible improvements.

The rest of the paper is organized as follows. Section II reviews Ensemble Routing. Section III describes our prototype on the Open Cirrus testbed. Section IV presents preliminary experimental results. Section V discusses our experience using Open Cirrus, and Section VI concludes.

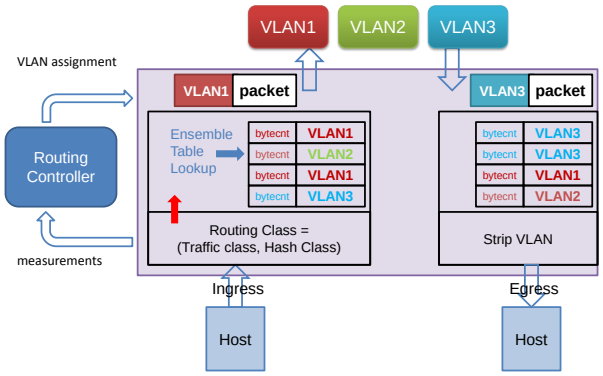


Fig. 1. Switch Architecture

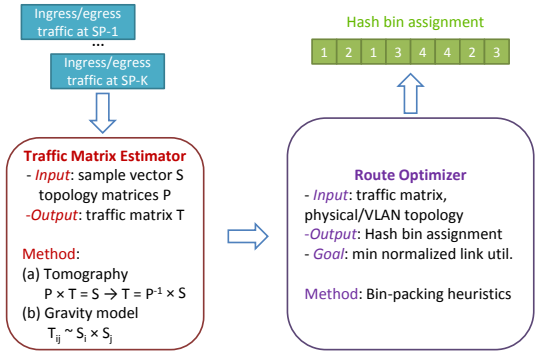


Fig. 2. Optimization Heuristics

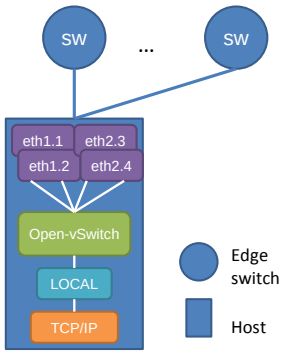


Fig. 3. Host Architecture

II. Ensemble Routing

Ensemble Routing operates at each edge switch attached to hosts, while non-edge switches can be unmodified commodity switches. Figure 1 shows the edge switch architecture for Ensemble Routing. Outbound packets transmitted by hosts are received at the edge switch, which takes the packet and classifies it into a traffic class, and applies a hash function to identify a hash class. Together, the traffic class and hash class are concatenated to form the routing class. The routing class is used to index a table, e.g. a TCAM (ternary content-addressable memory) common in modern switches, to look up the routing network to use to forward the outbound packet through the fabric to a remote edge switch connected to the destination host. The TCAM lookup also updates counters associated with the routing class to record the amount of traffic (packets and byte counts) transmitted using the routing class. Our implementation uses Layer-2 VLANs to create the multiple alternative routing networks that each packet can traverse. Unlike the common use of VLANs to provide logical isolation, these "routing VLANs" are used to provide path diversity (e.g. similar in SPAIN [4]). This approach could be extended, e.g. with VLAN stacking, to provide both logical isolation and multi-pathing. The outband packet is VLAN-encapsulated and forwarded to the destination edge switch, which decapsulates the packet, records the arrival in statistics counters associated with the routing VLAN, and strips the VLAN tag and forwards the packet to its destination host.

A logically centralized Ensemble Routing Controller operates in a control loop, periodically sampling traffic statistics and optimizing routing settings throughout the network. Each iteration of the control loop queries a set of sample points located in the network to collect per-routing class and per-VLAN traffic statistics. For our implementation, sample points are located at each edge switch. The routing controller uses the gathered statistics to compute optimized assignments of routing classes to VLANs to achieve load balancing across the network resources. At the end of each control loop iteration, the controller programs each edge switch TCAM with the resulting optimized assignment.

The optimization procedure is diagrammed in Figure 2. The techniques shown in the Figure are described in detail elsewhere [7] for the case of symmetric routing, where each bi-directional flow takes a congruent path through the fabric in both directions. In this paper we relax that restriction in some experiments and allow asymmetric routing, where bi-directional traffic may take different paths in each direction to improve performance through better load balancing.

III. Ensemble Routing Prototype

Our prototype implementation adds Ensemble Routing support to virtual switches running in software at each host. Figure 3 shows a host running Open vSwitch (openvswitch.org) which we modified to implement Ensemble Routing. The Open vSwitch has a number of ports that are associated with various network interfaces and with the local host TCP/IP stack. Using standard Linux configuration, we created a network interface object for each routing VLAN (four such interfaces are shown in the figure as eth1.1 through eth2.4). The host is attached to unmodified physical switches that have VLANs setup on alternative paths in the fabric.

Figure 4 shows our additions to the Open vSwitch design to support Ensemble Routing. Each packet coming from the local host is classified based on its header fields into a traffic class. Each traffic class is associated with a programmable hash table (so each traffic class can have its own hash table), and the hash table maps each hash bin to a routing VLAN. It is the responsibility of the central controller to program the hash bin mapping for each traffic class.

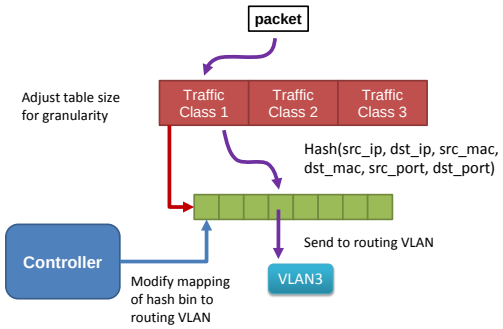


Fig. 4. Open vSwitch Modifications

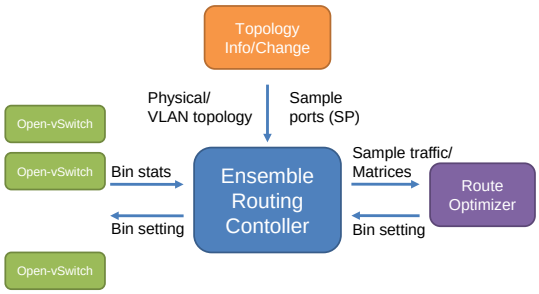


Fig. 5. Controller Overview

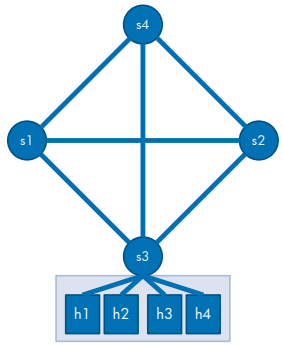


Fig. 6. Topology of Experiment Network in Open Cirrus

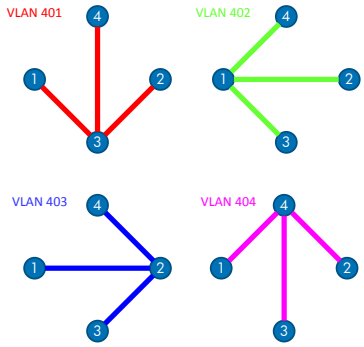


Fig. 7. VLAN Configuration for Multipath Routing

Figure 5 shows the controller, which runs in software on a dedicated host, and its inputs and outputs. Inputs include hash class (bin) statistics queried periodically from each Open vSwitch instance on all the hosts, a description of the physical topology and the routing VLAN layouts. The Controller takes this information and generates matrices summarizing this information to the Route Optimizer component, which executes a linear algebra package and a heuristic bin packing algorithm [7] to generate new optimized hash bin settings which are presented to the controller. The controller takes these bin settings and programs each Open vSwitch instance to change the routes associated with each routing class at each host.

IV. Evaluation

We performed experiments using a subset of the HP Labs Open Cirrus testbed. A separate "research network" was setup to perform datacenter network-related research, including Ensemble Routing. Each host in the testbed has two NICs: eth0 connected to the standard network, and eth1 connected to the research network. The research network consists of several edge switches connected in a clique topology – i.e., a fully connected network among the edge switches. All links are 1Gbps Ethernet. (The network also contains additional links and top switches to support a two-level fat tree topology, but we do not use those resources in the experiments reported in this paper). As shown in Figure 6, we used four edge switches s1–s4 and for each switch, we used four attached hosts to run our Ensemble Routing Open vSwitch (note: while the Figure only shows the hosts that are attached to switch s3, we attached similar sets of four hosts to each other edge switch s1, s2, and s4).

A routing VLAN is rooted at each edge switch as shown in Figure 7. To take a single-hop path from one edge switch to another edge switch, a packet can either use the routing VLAN rooted at the source edge switch (i.e. the "Home" VLAN) or the routing VLAN rooted at the destination edge switch. Using any other routing VLAN requires a two hop path: a first hop from the source edge switch to the root edge switch of the routing VLAN, and a second hop from there to the destination edge switch. For example, a packet from edge switch s3 to edge switch s4 can take a one-hop path using VLAN 401 rooted at edge switch s3 or VLAN 404 rooted at edge switch s4, but taking VLAN 402 requires traversing through s1, and taking VLAN 403 requires traversing through s2.

Our initial experiments examine the question: *What is the impact on TCP performance when hash bin mappings are changed dynamically?* The Ensemble Routing Controller dynamically optimized hash bin settings, potentially changing the paths of established TCP connections. This can cause packets to arrive out of order at the destination, leading to a possible concern that the benefit of dynamic traffic load balancing comes at the cost of a harmful impact on TCP congestion control dynamics.

To test this key question, we performed two experiments. In the first experiment, a hash bin mapping is alternated between two routing VLANs at a high frequency of 10 times per second. A single streaming TCP connection is established

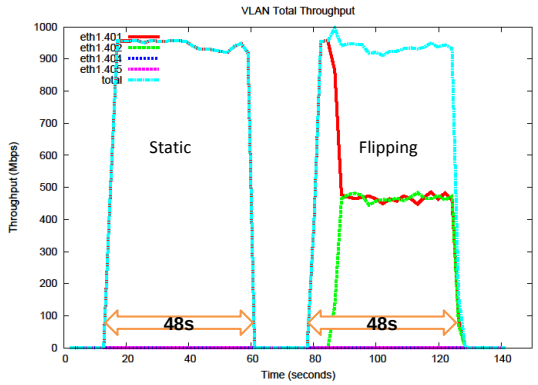


Fig. 8. Flipping without contention

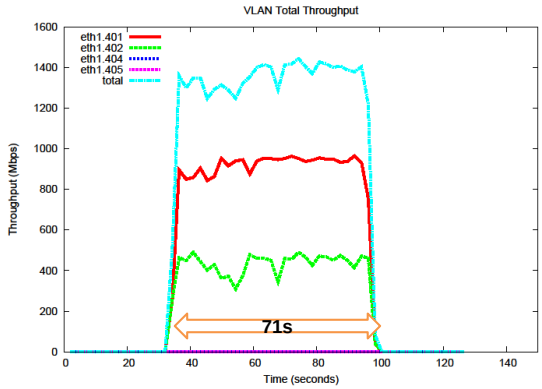
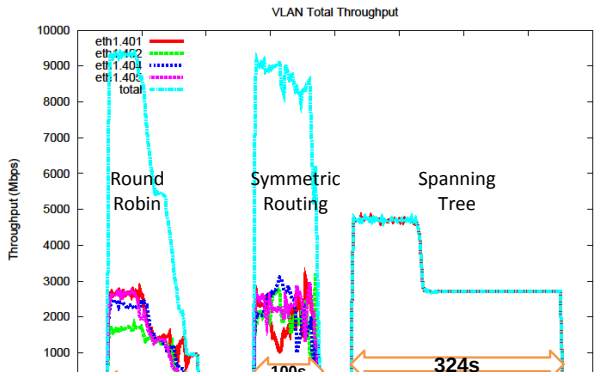


Fig. 9. Flipping with unequal contention



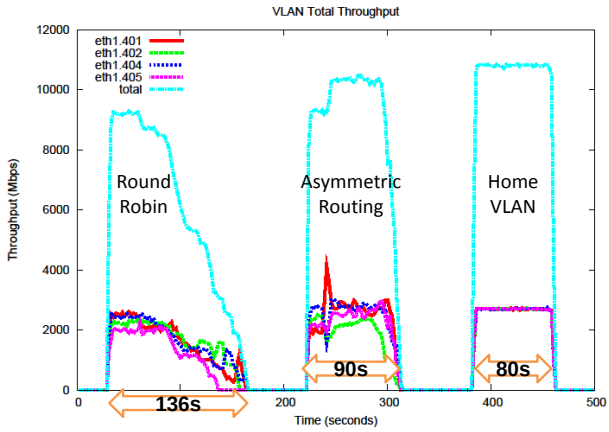


Fig. 10. All Pairs Shuffle

between two edge switches and transfers 500GB. As a result of the hash bin flipping between the two routing VLANs, packets using the established TCP connection alternate between a one-hop path and a two-hop path approximately every 100ms. Figure 8 shows the results of this experiment. The plot shows on the y-axis the traffic on each routing VLAN, and the total traffic across all VLANs, as time advances on the x-axis. Results of two runs are shown. The first is a static run that does not alternate the hash bin map. A streaming TCP connection completes a fixed size data transfer in 48 seconds. The second run performs the same streaming data transfer, but this time the hash bin map is flipped every 100ms. The result shows that the transfer still completes in 48 seconds, and the two VLANs 401 and 402 are approximately equally utilized. Thus, flipping between paths does not hurt goodput in this experiment.

The second experiment also alternates the hash bin map every 100ms, but adds a competing flow to one of the two alternate paths. Specifically, the streaming TCP flow alternates between an empty two-hop path, and a one-hop path shared with another competing streaming TCP flow that does not shift its path. The results, shown in Figure 9, show that the completion time of the streaming TCP job extends to 71 seconds, only 11% higher than the ideal value of $48 * 4/3 = 64$ seconds obtained if bandwidth sharing when the two flows compete is perfectly equal. These results suggest that TCP in the datacenter is likely to well tolerate path changes at frequencies up to several times per second. In the remainder of this paper, the controller limits hash bin changes to take place only every 2 seconds, a 20 times lower frequency than in these initial extreme experiments.

We next examine the question: *How well does Ensemble Routing optimize routes for different traffic patterns?* Our initial test uses an *All-pairs shuffle* traffic pattern, in which each host sends 500MB of data to each of the 12 hosts located at different edge switches. This traffic pattern has a perfectly uniform distribution across the edge switches. For this uniform traffic, the optimal routing policy is to take shortest paths for all traffic. Hence, all traffic should take the VLAN rooted at the source edge switch.

Figure 10 shows the performance obtained with different routing policies. Home VLAN is the optimal policy. Spanning Tree is the use of a single spanning tree as in many conventional Ethernet deployments, in which multiple paths are not used except for fail-over. Round Robin maps hash bins to routing VLANs in a round-robin pattern, approximating a random choice of routing VLAN for each flow. Two independent runs of Round Robin are shown, one in each plot. Finally, Symmetric Routing (SR) and Asymmetric Routing (AR) use the Ensemble Routing Controller with dynamic assignment of hash bin mappings. SR restricts bi-directional flows to congruent paths, while AR lifts that restriction. The result

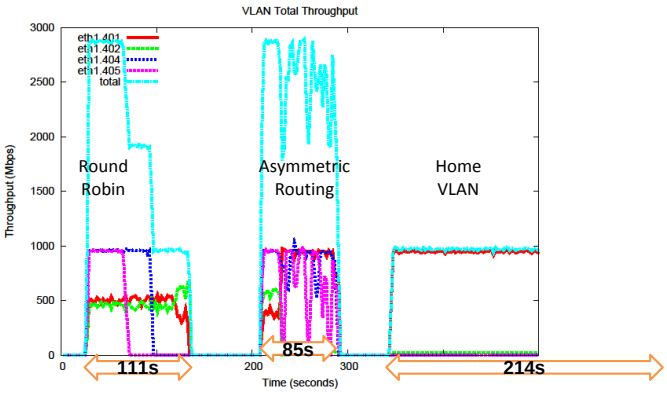


Fig. 11. Skewed Shuffle

shows that SR and AR outperform all but the optimal Home VLAN policy. AR outperforms SR and comes within 12.5% of optimal. Round Robin suffers because as individual data transfers complete, some VLANs become idle, whereas the SR and AR schemes can dynamically shift traffic to less utilized paths. Spanning Tree is the worst scheme because it is limited to 1Gbps, the bandwidth of a single path.

We also ran experiments using a *Skewed Shuffle* traffic pattern, where each of four hosts transmits 1500MB to each of four destination hosts. The source hosts are attached to a single edge switch, and the destination hosts are attached to a single edge switch different from the sources. Thus there are 4x4 data transfers, all traversing the same pair of source and destination edge switches. Unlike for the All-Pairs pattern, here it is *not* optimal to use only shortest-hop paths. The reason is that there is only one link between the source edge switch and the destination edge switch, limiting the total bandwidth between the pair of switches to 1 Gb/s if only the shortest path is used. Thus, unlike the All-pairs case in which Home VLAN is the optimal policy, with Skewed Shuffle it is better to use both the shortest-hop path and non-shortest-hop paths. Figure 11 shows results for Skewed Shuffle using Home VLAN, Round Robin, and Asymmetric Routing. The result shows that Home VLAN has very long completion time due to the 1 Gb/s bottleneck. Round Robin performs better by using more VLANs and, hence, all paths, but some VLANs go idle before the entire job is completed, leaving available bandwidth unused. Asymmetric Routing outperforms both static schemes, resulting in the fastest completion time of 85 seconds. The Ensemble Routing Controller is able to dynamically adjust the routing policy to keep all paths between the source and destination edge switches utilized throughout the execution, maximizing total bandwidth utilization.

V. Open Cirrus Experience

We used the Open Cirrus testbed at HP Labs before the deployment of an automated Node Reservation System that would allow users to reserve sets of hosts for specific time periods. To avoid interference between experiments, we had to coordinate carefully with other users of the infrastructure using email and in-person communication. In addition, all users shared the same file systems on each host. This meant that there was always a danger that some user would upgrade Ubuntu packages on a subset of the machines, changing their behavior or performance. Despite the potential for interference and system corruption, we were able to carry out the experiments with only occasional unintended interference, and we are not aware of any instances in which the file system contents were changed in a disruptive manner. We suspect this was possible only because the number of cooperative users at the time of our experiments was small. In addition, for most of our work, we were able to confine changes to a custom kernel module that we could dynamically load and unload, and we would boot each system before and after running experiments to try to ensure a fresh system.

The planned deployment of the Node Reservation System (NRS) should make all of this much easier for researchers. Hosts will be reservable, and strong content isolation will be provided between different users. For research on networking, we advocate extending NRS to allow resources of the research network to be reserved in addition to hosts. For example, we plan research that will require dynamically reconfiguring routing VLANs. This requirement suggests the need to extend NRS to reserve sets of switches and/or links, and VLAN identifiers along with hosts.

VI. Conclusion

The Open Cirrus testbed proved to be an extremely valuable resource for advancing our research on Ensemble Routing for large-scale datacenters. While we had previously tested our Ensemble Routing ideas using fluid-flow simulation, a real implementation was needed to evaluate and understand the effects of Ensemble Routing in the context of the full complexities of real systems and workloads. The results so far are promising, and we plan to further extend and evaluate our prototype on a variety of topologies and network traffic.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference*, Seattle, WA, Aug. 2008, pp. 63–74.

[2] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, S. R. Pardis Miri, V. Subramanya, and A. Vahdat, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," in *Proceedings of the ACM SIGCOMM 2009 Conference*, Barcelona, Spain, Aug. 2009.

[3] 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 *Proceedings of the ACM SIGCOMM 2009 Conference*, Barcelona, Spain, Aug. 2009, pp. 51–62.

[4] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "SPAIN: COTS Data-Center Ethernet for Multipathing over Arbitrary Topologies," in *Proc. USENIX NSDI*, 2010.

[5] M. Schlansker, J. Tourrilhes, J. R. Santos, and Y. Turner, "Killer fabrics for scalable datacenters," HP Labs, Tech. Rep. HPL-2009-26, Feb 2009.

[6] M. Schlansker, J. Tourrilhes, Y. Turner, and J. Santos, "Killer fabrics for scalable datacenters," in *Communications (ICC), 2010 IEEE International Conference on*, May 2010, pp. 1 –6.

[7] M. Schlansker, Y. Turner, J. Tourrilhes, and A. Karp, "Ensemble routing for datacenter networks," in *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '10. New York, NY, USA: ACM, 2010, pp. 23:1–23:12. [Online]. Available: http://doi.acm.org/10.1145/1872007.1872036