Enabling Flow-based Routing Control in Data Center Networks using Probe and ECMP

Kang Xi, Yulei Liu and H. Jonathan Chao
Polytechnic Institute of New York University, Brooklyn, New York 11201
kxi@poly.edu, yliu22@students.poly.edu, chao@poly.edu

***Abstract*—Data center networks often use densely interconnected topologies to provide high bandwidth for internal data exchange. In such network, it is critical to employ effective load balancing schemes so that the bandwidth resources can be fully utilized. A simple and widely adopted scheme is equal-cost multi-path (ECMP) routing, which is generally supported by commodity switches and routers. However, research shows that ECMP cannot always ensure even traffic distribution among multiple paths. Consequently, ECMP cannot guarantee optimal resource utilization. We propose a scheme to complement ECMP with per-flow reroute. The basic idea is to perform ECMP-based load balancing by default. When a congestion occurs on a certain link, we dynamically reroute one or a few big flows to alternative paths to alleviate the congestion. The main contribution of our research is to design a scheme that enables per-flow reroute without introducing any modifications to IP switches and routers. All the flow-based operations and reroute functionalities are implemented in software that are installed on end hosts and centralized controllers. We call this scheme PROBE (Probe and RerOute based on ECMP). PROBE uses a traceroute-like approach to discover alternative paths and modifies packet headers to enable flow-based reroute. We show that PROBE is a low cost, low complexity and feasible scheme that can be easily implemented in existing data center networks that consist of commodity switches and routers.**

***Index Terms*—reroute; ECMP; data center network; load balancing; flow-based routing**

I. Introduction

The purpose of load balancing in communication networks is to route traffic across multiple paths in a good way so that the work load on the links and/or nodes are evenly distributed. In practice, people usually focus on links to design and evaluate load balancing. Typically, routing in an autonomous system (AS) is based on shortest path algorithms, e.g., open shortest path first (OSPF) [14]. Without load balancing over multiple paths, the shortest path from a source to a destination is calculated in advance, and all the according traffic is directed through this shortest path.

Data center networks often use densely interconnected topologies to provide large bandwidth for internal data exchange. For example, fat-tree and Clos networks are widely adopted where a large number of paths exist between each pair of nodes [6], [15]. The newly proposed data center network topologies, including DCell [8], BCube [7], DPillar [11], also share the feature of dense interconnections. In this type of networks, using single-path routing without load balancing cannot fully utilize the network capacity. As a result, congestion may occur even if the network still has abundant unused bandwidth. In Fig. 1, if two shortest paths A-E-F-D and G-E-F-J are selected for single-path routing, link E-F may be overloaded even if paths A-B-C-D and G-H-I-J have unused bandwidth. This problem can be alleviated using equal-cost multi-path (ECMP) routing [9]. With ECMP, multiple shortest paths are calculated from a source to a destination, and traffic is distributed across these equal-cost paths to achieve load balancing. In Fig. 1, if A-E-F-D and A-B-C-D are used to carry traffic from A to D, while G-E-F-J and G-H-I-J are used to carry traffic from G to J, the network utilization will be greatly improved. With ECMP, each router may have multiple output ports for the same destination prefix, which lead to multiple paths. When a packet arrives, the router calculates a hash value based on the packet header and select one of the feasible output ports based on the hash value. It is common practice to use the 5-tuple header fields [source IP address, destination IP address, protocol type, source port, destination port] to calculate the hash value. With this approach, packets belonging to the same flow follow the same path, thus avoiding out-of-order delivery.

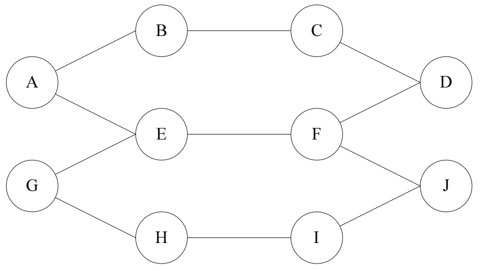

Fig. 1. Traffic load balancing.

However, using ECMP cannot guarantee good load balancing because of two major reasons. First, hash based traffic distribution is per-flow based, not per packet based. Thus the result is to balance the number of flows on different paths, not the bit rate. Even if two paths carry the same number of flows, the traffic loads may not be equal since the flows have different bit rates [1]. Second, from the network-wide viewpoint, using ECMP may still lead to overload on certain links. In Fig. 1, if A–D and G–J evenly distribute their traffic between the two paths, the load on link E-F would be twice of the load on any other links. One may think about adjusting the hash function in a sophisticate way to achieve network-wide load balancing. Unfortunately, this is infeasible because the traffic fluctuates all the time and route recalculation occurs each time there is a topology change. Therefore, tuning hash functions can barely follow such dynamic changes, let aside the considerable complexity.

A common approach to solve the problems of ECMP is flow-based routing. OpenFlow [13] defines a framework where switches and routers maintain flow tables and perform per-flow routing. Such flow tables can be dynamically modified from a remote station. Hedera [1] shows how to use OpenFlow in data center networks to achieve load balancing. There is no doubt that OpenFlow is a powerful scheme with great potential. However, it is not supported by existing commodity switches and routers, and the flow table configuration and maintenance are non-trivial.

In this paper we present a scheme called PROBE (Probe and RerOute Based on ECMP). PROBE exploits the multipath routing capability provided by ECMP to realize flow-based rerouting, thus enabling fine granular traffic control. The best advantage of PROBE is that it achieves flow-based rerouting without requiring flow tables in the routers. Actually it does not require any modifications of the existing routers. Instead, all the operations are performed at the end hosts. With this property, PROBE can be easily deployed in existing data center networks to achieve performance improvement.

II. Architecture of PROBE

A. Problem Description

We consider a network that employs ECMP for load balancing but do not introduce any restrictions to the hash function that is used by each router to determine traffic distribution. Consider the situation in Fig. 2 where a big flow from host s to host d experiences congestion on link B-C, our goal is to design a scheme that can discover an alternative path (i.e., either A-B-F-D or A-E-F-D in this example) and reroute the flow through this path to bypass the congested link.

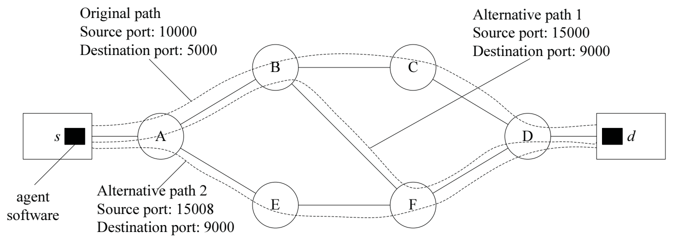

Fig. 2. Example of PROBE. By translating ports [10000, 5000] to [15008,9000] in packet headers, the source agent reroutes the flow from the original path to alternative path 2. The destination agent translates [15008, 9000] back to [10000, 5000].

B. Overview of PROBE

The essence of PROBE is to let the source host probe the network to discover alternative paths using a traceroute-like probe [12]. Since routers perform multi-path routing based on the 5-tuple header fields, our scheme conduct multiple probes with various 5-tuple field values to obtain alternative paths. Among the 5-tuple fields, we only change the protocol port numbers and keep the other fields unchanged. From the probe results, the host is able to create a table indicating the 5-tuple values for each alternative path. When the host needs to send packets along a specific alternative path for rerouting, it only needs to modify the 5-tuple header fields with the corresponding values from the probe results. Fig. 2 shows an example.

1) Set up: A TCP flow from s to d is currently taking A-B-C-D with source/destination ports [10000,5000]. Hosts s and d run an agent software that implements the PROBE scheme.
2) Probe: The agent software in host s starts multiple traceroute-like probes to discover alternative paths by setting the port numbers to various values. The IP addresses and protocol type are set to the same values as the original flow. From the probe responding packets, the source agent software finds that ports [15000,9000] corresponds to path A-B-F-D, while [15008,9000] corresponds to path A-E-F-D.
3) Reroute: When the original flow needs to be rerouted to avoid the congestion on link B-C, the agent software in host s creates an entry in its flow table to translate ports [10000,5000] to [15008,9000]. This entry is sent to the agent software at host d before any packet is rerouted. After that, the source agent translates each packet with ports [10000,5000] to [15008,9000] and sends it to the network. The routers perform ECMP-based multipath routing and deliver these packets along A-E-F-D to host d, not aware of the port translation. The destination agent at host d performs a simple table lookup and translate

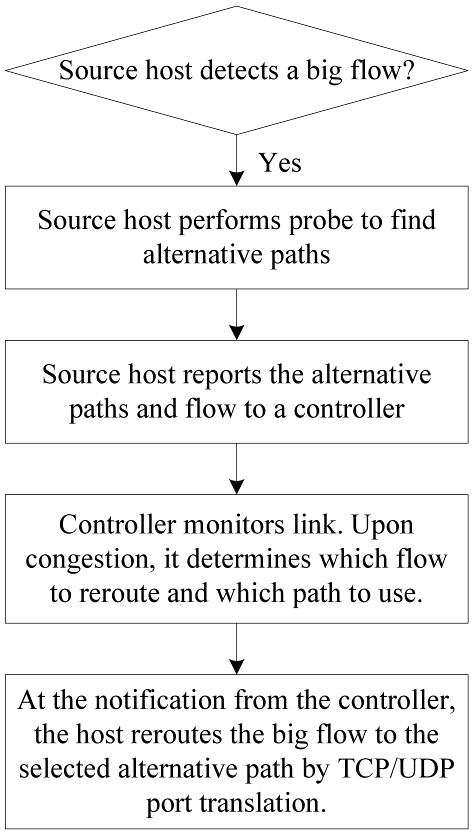
Source host detects a big flow?
Yes
Source host performs probe to find alternative paths
Source host reports the alternative paths and flow to a controller
Controller monitors link. Upon congestion, it determines which flow to reroute and which path to use.
At the notification from the controller, the host reroutes the big flow to the selected alternative path by TCP/UDP port translation.

Fig. 3. Architecture of PROBE.

the ports back to the original values. This is similar to network address translation (NAT) [5]. Finally, the packets are delivered to the applications as usual.

C. Technical Details

The architecture of PROBE is illustrated in Fig. 3. Flow detection, path probe, and flow reroute are function modules in end hosts and are implemented as part of the agent software. The path monitoring and reroute decision-making can be implemented either in a centralized controller or distributed in the agent software. Due to efficiency and simplicity, centralized controllers have been widely adopted in enterprise networks and data centers [1], [3], [6], [15]. In this paper, we focus on the implementation of PROBE with a centralized controller. The details of the function modules are explained below.

- Flow detection: Each end host monitors its out going traffic to identify big flows. A flow is defined using the 5-tuple header fields. A big flow is defined as a packet stream that lasts longer than a predetermined duration threshold and has a bit rate that is higher than a predetermined rate threshold. Reroute is performed only to big flows because moving such flows is effective to alleviate congestion. In contrast, small flows either last a short period or have low bit rate. Rerouting such flows is less effective to resolve congestion and improve quality of service. Flow detection can be performed using various methods. One typical approach is based on flow sampling, where packets are periodically sampled to measure/estimate the rate of flows, such as sFlow [16] and Cisco NetFlow [4]. While flow sampling may generate considerable error for small flows, it is generally good enough to detect big flows since such flows have more packets and have a higher chance of being sampled.
- Path probe: Since end hosts do not participate in routing, they do not have the topology information and routing information that are necessary for flow rerouting. To resolve this problem, we let end hosts obtain path information by probing the network. The approach we take is similar to traceroute. The software agent in the source node initiates multiple probes. Each probe is different from the other in that the probe packets carry different 5-tuple header field values. Since the routers hash the 5-tuple header fields to perform load balancing using ECMP, it is very likely that packets belonging to different probe sessions may take different paths. Similar to traceroute, the agent software sends multiple packets with the TTL fields set to 1, 2, 3, etc, thus getting feedback packets from the routers, with which the agent is able to reconstruct the multiple paths being maintained in the network. Unlike conventional traceroute, the agent software generates probe packets with TCP/UDP headers, just like tcptraceroute [17] and Paris traceroute [2], which is necessary since we rely on port numbers to achieve path control. We would like to clarify that path probe is performed only after a big flow is detected. For small flows, the conventional routing is applied. Since the number of big flows is a small fraction of the total traffic [10], the probe does not generate too much overhead traffic. We discuss this and several other critical issues in the next section.
- Path monitoring: In a data center network (or enterprise network) that employs centralized controllers, the traffic load of each link is constantly measured and reported to a controller. Therefore, it is straightforward to exploit this functionality to let the controller make rerouting decisions. After a host detects a big flow and completes path probe, it reports the big flow and the alternative paths to the controller, as illustrated in Fig. 4. The routers perform periodic measurement and reports the load on each link to the controller. When the load on a link exceeds a threshold, the controller picks a big flow that traverses this link and ensures that it has an

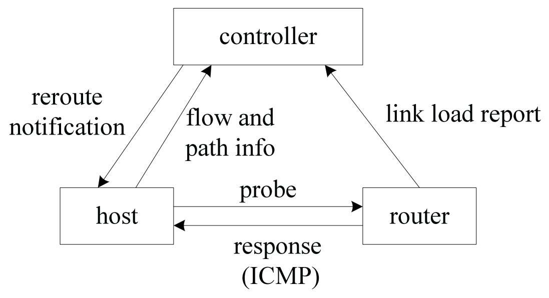

Fig. 4. Reroute decision-making using centralized control.

alternative path with available bandwidth for rerouting. After that, the controller notifies the source host to start the rerouting. The notification includes the flow ID and the path ID, with which the host agent software knows what values to use for port translation.

- Flow reroute: After a host receives a reroute notification from the controller, it first creates a translation bind between the original ports and the new ports and sends the information to the destination agent. The source and destination agents maintain a TCP connection to ensure reliable message exchange. After that, the big flow is rerouted simply by performing port translations at source and destination hosts. The IP address and protocol values are not changed to ensure the correctness of packet delivery.

D. *Advantages of PROBE*

The most prominent advantage of PROBE is that it enables explicit path control to avoid congested links without introducing any modifications to the routers. All the function modules are implemented in software and are running on end hosts and centralized controllers. PROBE can be easily deployed in an existing network using commodity IP routers and switches. It is a low cost, low complexity, and practical solution.

PROBE performs flow-based traffic control. Unlike OpenFlow [13] and Ethane [3], it does not require flow tables in switches and routers. While the software agents need to maintain flow-based lookup tables, the complexity is substantially lower because the table size is small, the update is simple, and the maintenance is totally distributed.

PROBE exploits the feature of ECMP, and complements it to achieve better load balancing. While conventional ECMP is simple, it does not achieve even traffic distribution and does not offer per-flow routing controllability. PROBE runs on top of ECMP and has the ability to fine tune the routing of big flows to fully utilize the resource utilization. PROBE helps to achieve great performance without introducing high complexity.

III. Design Considerations

A. *Path Probing*

One may wonder if the path probe would take too much time to meet the timing requirement of flow reroute. Conventional traceroute probes a path in a hop-by-hop manner. It starts by setting TTL=1 to discover the first router, and then increments the TTL value by one in each time. Overall the delay increases with the hop count. PROBE takes several methods to reduce the delay. The first method is to probe early. Probes are triggered immediately after a big flow is detected, most likely this is before congestion occurs. The second method is to probe multiple hops in parallel, that is, the source agent software sends out multiple packets with the TTL fields set to 1, 2, 3,... Consequently, the probe delay does not increase with the hop count. The third method is to probe multiple paths in parallel. E.g., we can start 10 probes (with 10 different port pair values) all at once. The last method is to configure the routers to respond to such probe packets quickly. Our measurement shows that even low-end commodity routers can easily reach sub millisecond level delay. While certain ISPs restrict the response rate for traceroute due to security considerations, it should not be a problem for data center networks where the hosts are trusted parties. Also the propagation delay is trivial in data center networks where the physical distance is short. We believe for data center networks it is feasible to achieve sub 100 ms probe delay. Noting that big flows often last for seconds or even more than 10 seconds [10], 100 ms is a fairly reasonable number.

B. *Agent Communication and Caching*

The source and destination hosts maintain a permanent TCP connection to exchange control messages. Once a rerouting decision is made, the source host creates a port translation rule and sends it to the destination agent for reverse port translation. Therefore, the source and destination agents have consistent translation lookup tables. The source agent can cache the results of a path probe operation for later use, so that it does not need to perform probe frequently to the same destination host. To keep the table up-to-date we allow the entries to age out.

C. *Traffic Overhead*

While each probe generates extra overhead traffic, it does not introduce impact to data center networks for several reasons. First, the probe is triggered only by big flows. According to traffic measurement [10], only a small fraction of flows (about 20%) are big flows while most flows are small flows. Second, the PROBE agents cache path information for later use, thus they do not probe the same destination frequently.

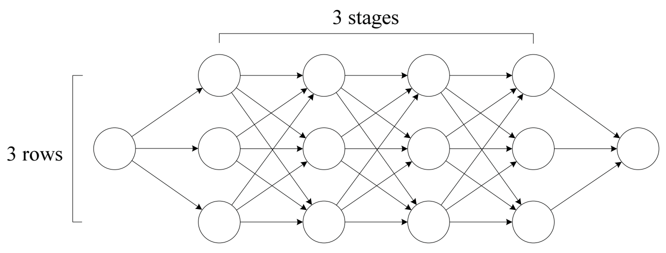

Fig. 5. A multi-stage topology ($R = 3, T = 3$).

Third, the probe packets are small, containing a 20-byte IP header and a 20-byte TCP header (or a 8-byte UDP header). So the total short-term peak rate bound of a probe is

$$r = N \cdot H \cdot L/\Delta, \tag{1}$$

where N is the number of parallel probe sessions, H is the maximum hop count, L is the length of a probe packet, and Δ is the probe duration.

When a physical computer is virtualized into multiple virtual machines, the overhead can be reduced by consolidating the probe operation. E.g., we can integrate the PROBE agent with the hypervisor so that all the virtual machines can share the same probe results.

IV. Performance Evaluation

In this paper we focus on the design of PROBE and its feasibility. While the evaluation of load balancing performance improvement using PROBE is an important issue, we leave it to our future research. In this section we mainly study the effectiveness of path probe. Noting that ECMP distributes packets randomly (using hash function), it is possible that two or more probe operations happen to give the same path. Intuitively, in a data center with rich connectively (e.g., fat tree or Clos network topology), it is fairly easy to discover a number of different paths (not necessarily disjoint, though) using multiple probe. We perform in-depth study of this issue in the follows.

For simplicity, we conduct performance evaluation in a regular topology shown in Fig. 5, where the source node s connects to the destination node d through a R-row, T-stage Clos network.

A. Analysis

Assume all the links have equal cost, it is easy to see that the total number of equal-cost shortest paths from s to d is R^{T+1}. We assume all the routers perform load balancing using independent, uniformly distributed hash functions. Given an existing path, the probability that a random probe finds a non-overlapping path is

$$p_1 = 1 - \frac{1}{R^{T+1}}. \tag{2}$$

Therefore, the average number of probes needed to find a different path is

$$M_1 = \sum_{m=1}^{\infty} m p_1 (1-p_1)^{m-1} = \frac{1}{p_1} = \frac{R^{T+1}}{R^{T+1}-1}. \tag{3}$$

Since data center networks are densely interconnected, the total number of path R^{T+1} is a large number. Therefore, M_1 is close to 1, which means it is very easy to find a different path in just a few probes.

The above analysis does not specifically require the new path to be link-disjoint from the existing one. Although PROBE does not require the reroute to be link-disjoint, it is interesting to study the probability. Based on simple analysis, we find the probability that a random probe gives a link-disjoint path is

$$p_2 = \frac{R-1}{R}\left(\frac{R^2-1}{R^2}\right)^T. \tag{4}$$

Similarly, the average number of probes needed to find a link-disjoint path is

$$M_2 = \sum_{m=1}^{\infty} m p_2 (1-p_2)^{m-1} = \frac{1}{p_2} = \frac{R}{R-1}\left(\frac{R^2}{R^2-1}\right)^T. \tag{5}$$

For a small network with $R = 10$ and $T = 5$, the average number of probes is $M_2 = 1.16$.

B. Simulation

With the above analysis, we have the intuition and with a few probes, we should be able to discover enough alternative paths to get prepared for reroute. We conduct computer simulations to obtain the number of different paths being discovered vs the number of probes in various size topologies. The hash function we use in the routers is salted CRC16, where each router uses a random salt to avoid correlation between different routers in traffic distribution. Note that this is a necessary requirement by the native ECMP to achieve load balancing, not a requirement by PROBE.

In the ideal case, k probes should discover k different paths. In practice, the number of paths is most likely less than k. In our simulation, we perform a sequence of probes to observe the number of paths being discovered. We conduct such experiment multiple times to get the average numbers and worst numbers. Fig. 6 shows the results for a small network with 5 rows and 4 stages. With 20 probes, we are able to find more than 19 alternative paths on average. In the worst case, we find 17 alternative paths. Fig. 7 shows a large network with 20 rows and 6 stages. With 20 probes, we always find 20 different alternative paths.

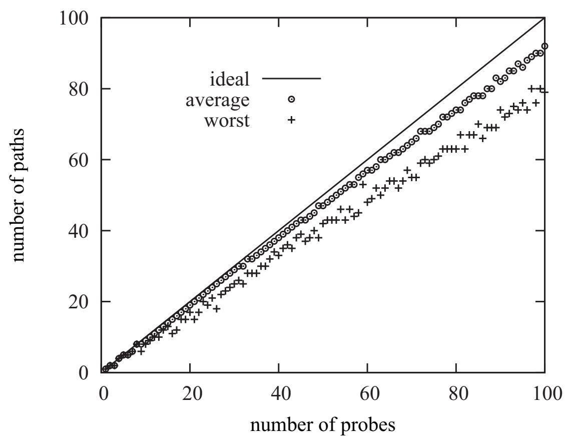

Fig. 6. Number of alternative paths discovered ($R = 5, T = 4$).

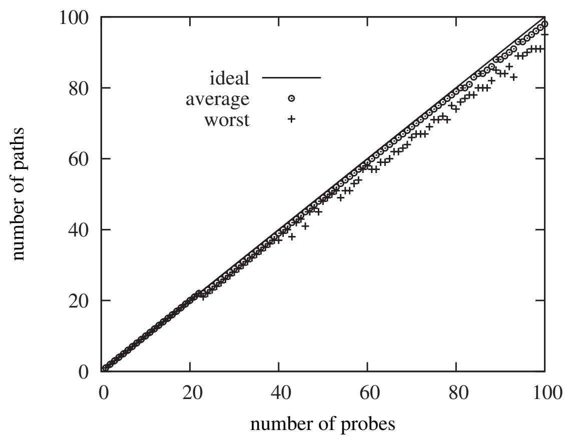

Fig. 7. Number of alternative paths discovered ($R = 20, T = 6$).

V. Conclusions and Future Work

We present a scheme called PROBE to enable flow-based reroute in IP network that uses ECMP. ECMP performs rough and static load balancing, thus cannot achieve even traffic distribution. The proposed scheme complements ECMP to perform dynamic flow-based reroute, thus achieving good load balancing and improving resource utilization. A prominent advantage of PROBE is that although it performs per-flow reroute, it does not require flow tables in IP routers. Everything is implemented in software that can be easily installed on host servers and controllers. Therefore, PROBE is a low-complexity, low-cost, and low-maintenance scheme, especially for data center applications. Since PROBE does not introduce any modifications to the routers, it can be used to upgrade existing data center networks for performance enhancement. This paper focus on the design and feasibility of PROBE, in our future research we will investigate the application of PROBE for network performance improvement and will build a testbed to demonstrate the design.

References

[1] M. Al-fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *In Proc. of Networked Systems Design and Implementation (NSDI) Symposium*, 2010.

[2] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with paris traceroute. In *IMC*, 2006.

[3] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: taking control of the enterprise. *SIGCOMM Comput. Commun. Rev.*, 37(4):1–12, 2007.

[4] B. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), Oct. 2004.

[5] K. Egevang and P. Francis. The IP Network Address Translator (NAT). RFC 1631 (Informational), May 1994. Obsoleted by RFC 3022.

[6] 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 '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 51–62, New York, NY, USA, 2009. ACM.

[7] 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 '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 63–74, New York, NY, USA, 2009. ACM.

[8] 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 '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 75–86, New York, NY, USA, 2008. ACM.

[9] C. Hopps. Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992 (Informational), Nov. 2000.

[10] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 202–208, New York, NY, USA, 2009. ACM.

[11] Y. Liao, D. Yin, and L. Gao. DPillar: Scalable Dual-Port Server Interconnection for Data Center Networks. In *IEEE ICCCN*, 2010.

[12] G. Malkin. Traceroute Using an IP Option. RFC 1393 (Experimental), Jan. 1993.

[13] 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.

[14] J. Moy. OSPF Version 2. RFC 2328 (Standard), Apr. 1998.

[15] 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 *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 39–50, New York, NY, USA, 2009. ACM.

[16] P. Phaal, S. Panchen, and N. McKee. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC 3176 (Informational), Sept. 2001.

[17] M. C. Toren. tcptraceroute. In *http://michael.toren.net/code/tcptraceroute*.