Layer-3 Multipathing in Commodity-based Data Center Networks

Ryo Nakamura
University of Tokyo
Email: upa@wide.ad.jp

Yuji Sekiya
University of Tokyo
Email: sekiya@wide.ad.jp

Hiroshi Esaki
University of Tokyo
Email: hiroshi@wide.ad.jp

Abstract—**Improving availability and throughput is a significant challenge for data center networks. Recent studies have attempted to use a variety of routing and multipathing techniques. However, no method has yet managed to combine availability and throughput improvement with actual deployability, usually because of dedicated hardware requirements. In this study, we focus on commodity-based layer-3 data center networks. We propose a method for layer-3 multipathing via the novel usage of common IP encapsulation techniques for throughput improvement. Our approach requires minimal software modifications at end hosts. The method enables us to construct an efficient data center network using *commercial off-the-shelf* (COTS) switches. In this paper, we describe the approach and an evaluation, in which our approach achieves an above-85% forwarding rate in general, and 100% for an experimental traffic pattern.**

I. Introduction

Improving the availability and throughput of networks is quite an important topic for data center environments. For a data center network, availability means being able to adapt flexibly to topology changes. During the operation of a service network, the topology may change from time to time for a variety of reasons, including maintenance, enhancements and breakdowns. The data center network should maintain its services while adapting flexibly to these topology changes. In addition to maintaining its availability, improvements to the throughput of a given network should be sought. With the emergence of cloud computing and distributed applications, traffic in data centers has been growing rapidly. Nevertheless, administrators cannot simply deploy additional hardware to increase capacity on every occasion because of cost. Therefore, improving the aggregated bandwidth of a given network is an important challenge for data centers.

To address these challenges, many approaches have been proposed by both operator and research communities. A hop-by-hop routing architecture with dynamic routing protocols is a good way to deal with the availability issue. When the topology is changed, the network switches detect the change and recalculate the routing tables, following which the network recovers autonomously. It is not necessary to consider the totality of the network. Effectively, this approach localizes the impacts of topology changes. The use of the traditional IP network is a proven technique from the viewpoint of availability. Constructing data centers using layer-3 networks to support availability has been proposed and the method developed [1], [2]. In addition, some methods introducing this architecture to layer-2 networks have been proposed [3], [4].

To improve the throughput of a given network, there are many approaches applicable to data center networks. One popular technique is the use of multipathing [5]. Prior work on topologies that have alternative paths between arbitrary hosts includes fat-tree, BCube [6], and random graph [7]. Furthermore, various approaches improving throughput by multipathing via these topologies have been proposed [5]. A classic way to achieve multipathing is Equal Cost Multipath (ECMP). ECMP-enabled switches divide traffic into multiple next-hops by considering hash value of flows. Alternative schemes achieve efficient multipathing by introducing special addressing schemes and modified hardware to both layer-2 and layer-3 networks [8], [9].

Although existing approaches achieve good availability and/or throughput via multipath topologies, many of these approaches have yet to be deployed. Economic cost is a significant problem for data centers [10], therefore, administrators should use commodity products on most occasions. However, existing proposals for availability [3], [4] and throughput improvement by multipathing [8], [9] require hardware that is not commercially available. This is a barrier to deployment and increases the cost of building a network. To achieve good throughput while using commodity hardware, ECMP is the usual approach [1], [2], [11]. However, it is known that ECMP cannot use multiple paths efficiently because ECMP does not account for flow bandwidth, which can lead to oversubscription [8]. Accordingly, there is no current method that can achieve optimal both availability and optimal throughput while using commodity switches.

In this paper, we focus on multipathing for a layer-3 data center network with unmodified commercial off-the-shelf (COTS) switches. Layer-3 multipathing offers both availability and throughput improvement, and using COTS switches eliminates barriers to deployment. In the proposed method, traffic is split into multiple paths using a novel usage of IP encapsulation technique. We evaluated the proposed method for a fat-tree topology. The results show that our approach achieves a forwarding rate of 100% for a particular traffic pattern and over 85% for all traffic patterns. This is comparable to the performance of existing proposals that require dedicated hardware.

The contributions of this paper include the following.

- A proposed method for layer-3 multipathing with COTS switches.
- The design and implementation of modifications for the end host software stack.

In the following sections, related work is summarized and compared, we describe our approach and we give the results of an evaluation of its throughput improvement. Finally, we summarize the results of this study and suggest future work.

II. Related Work

Various approaches addressing issues for data center networks have been considered, proposed, and developed. Table I shows a comparison of characteristics of prior work based on SPAIN [12] work. We have added a Topology Change column to the table, which indicates availability under topology changes.

As mentioned above, a layer-3 network localizes the impact of topology changes by using hop-by-hop routing with dynamic routing protocols. Facebook [2] and VL2 [11] achieve their availability by constructing a backbone network based on a layer-3 network. In addition, both approaches use COTS switches, which support deployability. However, they use ECMP to improve throughput on a multipath topology. Ordinary ECMP splits traffic into multiple next-hops by hashing 5-tuple *without* any regard for flow bandwidth. In this way, ECMP will tend to direct traffic to a particular path. Therefore, ECMP cannot use the multiple paths efficiently [8].

In contrast to ECMP, SPAIN [12] uses VLAN for multipathing on an arbitrary topology with COTS switches. SPAIN proposes a greedy algorithm to identify each path in a VLAN. In SPAIN, end hosts split traffic into multiple paths for identified VLANs. SPAIN achieves both good deployability and throughput improvement. However, recomputing and reconfiguration are required when the topology is changed such as switches being added or dropped. Furthermore, SPAIN lacks the characteristic of availability.

TRILL [3] and SEATTLE [4] offer availability by introducing a hop-by-hop routing architecture to a layer-2 network. However, they require dedicated hardware for the network switches, resulting in a deployability barrier. On the other hand, PortLand [9] achieves throughput improvement with multipathing by introducing a dedicated address assignment scheme and topology restrictions. The dedicated addressing scheme for multipathing on an IP network was first proposed by Al-Fares *et al.* [8]. PortLand updates this scheme for the MAC address of Ethernet. However, it also requires the installation of dedicated hardware switches.

In this paper, we propose a method for achieving multipathing in a layer-3 network that is different from ECMP with unmodified COTS switches. It provides availability as a benefit of using the layer-3 network, throughput improvement by using a multipathing method and deployability by using COTS switches.

III. Approach

We propose the method on the assumption that the commodity data center network is based on IP routing and forwarding technologies that are already available via COTS switches. Enabling a layer-3 network design provides both availability and deployability. In the proposed method, the end hosts select a path among multiple paths for each flow of traffic. Hardware COTS switches do not have a functionality of efficient traffic balancing, and ECMP does not consider the bandwidth of an individual flow which can lead to over-subscription. We therefore shift the functionality of traffic balancing from the network switches to the end hosts, similarly to SPAIN. However, in contrast to SPAIN, the end hosts use the IP encapsulation technique supported by COTS switches to specify path in the layer-3 network.

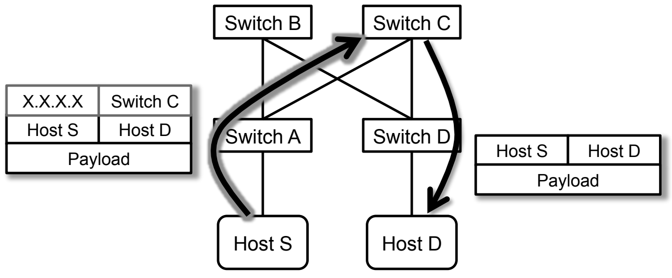


Fig. 1. How to specify the extra path using encapsulation techniques. When the shortest-path from host S to D passes through switch B, host S specify the extra path through switch C by adding outer IP header to switch C.

Fig. 1 shows the overview of the proposed method. Each path from a host to others is specified by a corresponding relay switch. When the shortest path from switch A to D passes through B, the end hosts can identify an extra path by specifying switch C, and split their traffic by specifying either switch B or C for packets. Therefore, if an identifier is added to a packet that specifies the address of the relay switch, an end host can select a path explicitly.

To route a packet through a specified relay switch, the source route option can be used. Source route options, IPv4 loose source and record route option and IPv6 routing header, are standardized as a part of the IP stack specification, and almost all COTS switches therefore support them routinely. When a host sends a packet to another host, the sender can add a source route option with the address of the corresponding relay switch to specify a through path explicitly. However, the source route option causes degradation of the forwarding performance because it is usually processed by the CPU. The source route option is not suitable in practice.

Instead of using a source route option, we use an IP encapsulation technique as an identifier to specify the relay switch. In the proposed method, the end hosts add a tunneling header and encapsulate it in an outer IP header when sending a packet to other hosts. The destination address of the inner header is the destination host, and the source address of the inner header is its own address. Further, the destination address of the outer header is the relay switch, and the source address of the outer header is a proper address that does not exist in the network. As shown in Fig 1, host S sends a packet to host D encapsulated in an outer IP header with switch C as the destination. When the packet is received by switch C, the outer IP header and tunneling header are removed, with the original packet then being forwarded to the original destination, host D. In this way, end hosts can specify a path from a set of multiple paths by using tunneling protocols. Note that the encapsulation technique will incur throughput degradation because of the reduction of packet size. However, this may not be significant because the jumbo frame is often adopted in data center environments.

TABLE I. COMPARISON OF RELATED WORK BASED ON SPAIN [12].

	Wiring Topology	Usable paths	Uses COTS switches?	Topology Change
Facebook [2]	Multiple Tree	ECMP	YES	YES
TRILL [3]	Arbitrary	ECMP	NO	YES
SEATTLE [4]	Arbitrary	Single Path	NO	YES
PortLand [9]	Fat-tree	ECMP	NO	NO
SPAIN [12]	Arbitrary	Multiple Paths	YES	NO
VL2 [11]	Fat-tree	ECMP	YES	YES
Proposed method	Arbitrary	Multiple Paths	YES	YES

When using tunneling protocols for this approach, a tunnel scalability problem may occur. To utilize all links in a data center network efficiently, each end host should be able to specify any relay switch. The number of tunnels will then be given by the number of $hosts \times relayswitches$, whereas COTS switch hardware supports at most hundreds of tunnels. Therefore, this will not be suitable for large-scale data center network that accommodates thousands of servers. The proposed method avoids this scalability issue by using a *non-existent address* for the source address in the outer IP header. The *X.X.X.X* shown in Fig. 1 indicates this address. In this method, the packet direction through a tunnel is always unidirectional. In contrast to conventional tunnel usage, packets are always encapsulated and sent from end hosts to relay switches (no encapsulated packet is sent from relay switches to end hosts). Therefore, we can use a single IP address that does not exist in the network as the source address for all tunnels. As a result, each relay switch will have only one tunnel between *X.X.X.X* and the relay switch itself, thereby avoiding the tunnel scalability issue.

End hosts maintain sets of addresses of relay switches for each corresponding destination network. Relay switches identify corresponding paths from a host to a destination. When applications send packets, the destination address of each packet is checked against the destination prefixes. If there is a matched, packet is encapsulated with a tunneling protocol to a relay switch selected from the set of relay switches for that prefix. In this way, end hosts can split the traffic to destinations via the set of corresponding relay switches.

A. Flow Assignment Algorithm

In addition to this method of specifying a path using the end hosts, an algorithm for balancing the traffic load across multiple paths is needed. End hosts should split traffic strategically into multiple paths, aiming to maximize the aggregated bandwidth of the whole network. Traffic sent from a host consists of flows that are identified by a 5-tuple (source and destination IP addresses, source and destination port numbers, and protocol number). Packets belonging to the same flow should be taken through the same path to avoid packet reordering that would cause performance degradation for both TCP and the applications. Moreover, each flow has different bandwidth. Therefore, the algorithm working in the end hosts should select paths for each flow taking account of the bandwidth of individual flows. In this study, we use two algorithms, called the hash-based and the flow-based algorithms to maximize the aggregated bandwidth.

The **hash-based** algorithm is the same as ECMP. A relay switch for a flow is decided by a calculated hash value from the 5-tuple for the flow. This algorithm does not consider the bandwidth of individual flows, so the traffic will tend to be via a particular path. As a result, it will not be able to maximize the aggregated traffic.

The **flow-based** algorithm is based on an algorithm proposed by Al-Fares *et al.* [8]. This algorithm considers the bandwidth of individual flows, and attempts to balance traffic equally among multiple paths. Balancing flows among paths equally is a well-kwon problem and is a variant of the NP-hard bin packing problem [8]. Therefore, Al-Fares *et al.* propose a heuristic algorithm for assigning flows among multiple paths via the modified switch. We adapt this algorithm for use with end hosts. The adapted algorithm is outlined as Algorithm 1.

Algorithm 1 The flow assignment algorithm.

```
function ENCAPSULATEPACKET($packet$)
    Hash 5-tuple of $packet$, and find this flow $f$.
    if such a flow $f$ exists then
        update packet counters of flow $f$;
    else
        Record the new flow $f$;
        Assign flow $f$ to a relay by the hash value;
    end if
    Encapsulate $packet$ to relay switch flow $f$ assigned;
end function

function REARRANGEFLOWFORPREFIX($perfix$)
    for $relay$ in $relays$ of $prefix$ do
        Find the $Rmax$ and $Rmin$ with the largest and
        the smallest traffic respectively;
        Calculate $D$, the difference between $Rmax$ and
        $Rmin$;
    end for
    for $flow$ in $flows$ of $prefix$ do
        Find the largest flow $f$ whose size is smaller
        than $D$;
    end for
    if such a flow exists then
        Switch the relay point of flow $f$ to $Rmin$;
    end if
end function
```

EncapsulatePacket() is called to decide on a relay switch when packets are sent from applications. Packets belonging to known flows are encapsulated with an outer IP header for the relay switch to which flows are assigned. If a packet is not part of an existing flow, it is recorded as a new flow and a relay switch is assigned using the hash value of the 5-tuple. *RearrageFlowForPrefix()* is called periodically for each destination prefix. It reassigns the largest flow that is smaller than the margin between the largest and the smallest flows for that destination.

B. Usable Tunneling Protocols

Various tunneling protocols have been proposed, standardized, and developed. The proposed method adopts a tunneling protocol to specify each path similarly to source routing. To achieve tunnel scalability, the source address of outer IP headers from all end hosts are faked as an IP address, as described above. Therefore, those types of stateful protocols that require maintenance of the session between tunnel end points cannot be used. For example, some tunneling protocols using authentication methods such as IPsec technologies and Layer 2 Tunneling Protocols cannot be used for our approach.

In contrast to stateful protocols, stateless protocols can be adopted for our proposed method. Table II shows usable tunneling protocols. Encapsulation techniques are instrumental for various purposes such as Virtual Private Network (VPN), network virtualization, and software-defined networks. Tunneling protocols have been proposed and developed over time for many uses. Therefore, even if dedicated schemes and hardware for multipathing are not developed, the proposed method can be used throughout actual data center environments. Administrators of data centers should choose among these protocols in accordance with their equipment.

TABLE II. USABLE TUNNELING PROTOCOLS FOR THE PROPOSED METHOD.

Protocol name	Specification	Published year
IP in IP Tunneling	RFC1853	1995
GRE	RFC2784	2000
EtherIP	RFC3378	2002
LISP	RFC6830	2011
VXLAN	RFC7348	2014
NVGRE [13]	Internet Draft	2012 ∼
GUE [14]	Internet Draft	2013 ∼
Geneve [15]	Internet Draft	2014 ∼

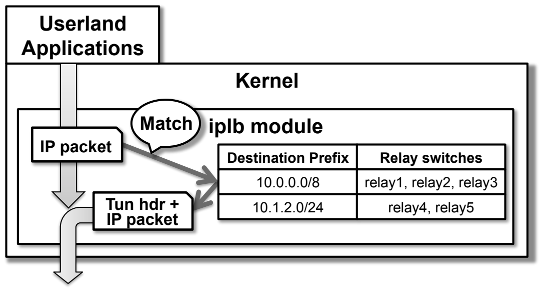
IV. End Host Modification

The proposed method shifts the functionality of traffic load balancing from the network switches to the end hosts. In this section, we describe the design of the modification for end hosts called `iplb`. When a host sends a packet, `iplb` finds a set of relay switches corresponding to the destination prefix of the packet. Then, if the prefix is found, the host selects a relay switch using the algorithm described in Section III-A and encapsulates it in a tunneling protocol.

Fig. 2 shows an overview of the modification. The function `iplb` constructs the longest match-based routing table in kernel space. Each entry in the table contains a set of relay switches for a corresponding prefix. When an application in userland sends a packet, `iplb` checks the destination address of the packet. If the destination address is found in the table, the packet is encapsulated in a tunneling protocol. The relay switch for the destination in the outer IP header is decided by either the hash-based or the flow-based algorithm from the set of relay switches.

We implemented `iplb` as a kernel module for Linux[1]. It provides the functionalities described above and an API to modify and control the routing table and relay switch entries in Linux kernel space. This API is designed and implemented as a protocol family in Netlink [16]. In addition, we implemented

[1]https://github.com/upa/iplb



Destination Prefix	Relay switches
10.0.0.0/8	relay1, relay2, relay3
10.1.2.0/24	relay4, relay5

Fig. 2. Overview of the modification for software stack of end hosts.

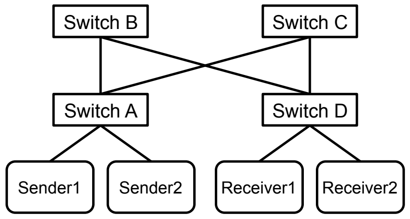


Fig. 3. The experimental topology for the preliminary evaluation.

an extension for the iproute2 [17] software suites. By using this iproute2, users can add, delete, or modify the routing table.

V. Evaluation

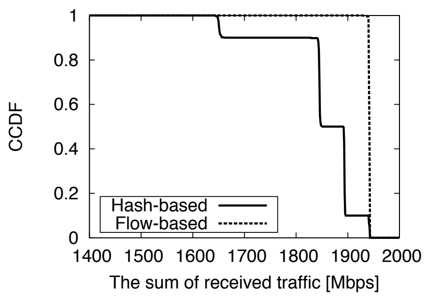
Availability and deployability are achieved by using a commodity-based layer-3 network as described in Section II and III. In this section, we investigate throughput improvement using our proposed method. To demonstrate that the method can use multiple paths efficiently, we evaluated the method in two cases. 1) as a proof-of-concept for the method, we adopted it for a simple multipath topology, and 2) we evaluated it for a fat-tree topology as a typical example of a data center network.

A. Preliminary Experiment

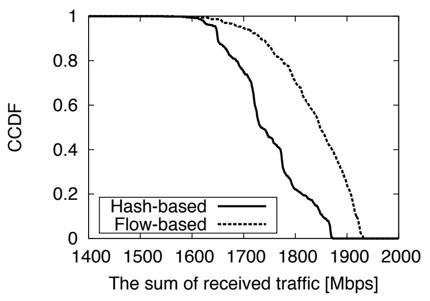
In the preliminary experiment, we applied the method to a simple multipath topology that had two paths between four hosts. The test topology is shown in Fig. 3. We prepared the experimental set up using commodity equipment. The switches were Juniper Networks EX4200-48P, the two sender and two receiver hosts were 1U servers with an Intel Xeon L3426 (1.87 GHz) CPU and 4 GB memory. All network interfaces were 1000BASE-T. In the experiment, Sender1 sent test traffic to Receiver1, and Sender2 sent test traffic to Reveicer2, balancing the traffic via SwitchB and SwitchC using the proposed method. We evaluated the performance in terms of the sum of the received traffic for the two receivers. When not multipathing, the link between SwitchA and SwitchB will overload and the overall received traffic will be 1000 Mbps.

The test traffic from a sender comprised multiple flows. To generate test traffic with multiple flows, we implemented a traffic generator called flowgen[2] witch generates multiple UDP flows to a host. In addition, flowgen can support the following three distribution patterns of bandwidth for individual flows to emulate real traffic.

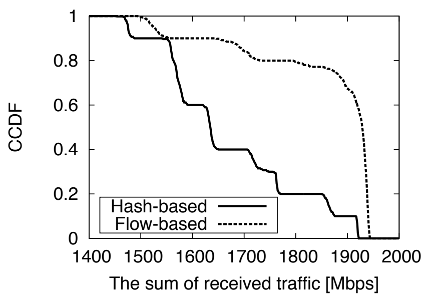
[2]https://github.com/upa/flowgen



(a) Same



(b) Random



(c) Power-law

Fig. 4. Results of the preliminary experiments. The y-axis represents a complementary cumulative distribution functions for aggregated received traffic on receiver1 and receiver2 for Same, Random, and Power-law flow distribution patterns.

- Same: the ratio of bandwidth for each flow is uniform.
- Random: the ratio of bandwidth for each flow is random.
- Power-law: the ratio of bandwidth for each flow follows a power law.

In this experiment, all distribution patterns were evaluated with 20 flows. The packet size for the test traffic was 1024 bytes.

Fig. 4 shows the results of the experiment. The y-axis of the results represents a complementary cumulative distribution function across 20 runs/permutations of tests for each flow distribution pattern, over 1 minute. Fig. 4(a) shows the result for the Same flow distribution pattern. The hash-based algorithm achieves 100% probability up to 1650 Mbps. Further, the traffic trended as expected. In contrast to the hash-based algorithm, the flow-based algorithm achieved 100% link utilization for Same flow distribution pattern. Moreover, for the Random (Fig. 4(b)) and Power-law (Fig. 4(c)) patterns, the flow-based algorithm performed better than the hash-based algorithm. This shows that the flow-based algorithm can use multiple paths better than a hash-based algorithm similar to ECMP. However, the flow-based algorithm cannot achieve 100% link utilization for the Random or Power-law flow distribution patterns. This could be caused by the heuristic algorithm and/or possibility that there is no assignment pattern achieving 100% link utilization. In addition, the reason that not all patterns can achieve 2000 Mbps is the encapsulation overhead. In this preliminary experiment, we used a GRE header for encapsulation. This involves 2.3% (46 Mbps) of throughput reduction with 1024-byte packets. Despite this, for a packet size of 9000 bytes, the throughput reduction is just 0.3%.

B. Evaluation for a Fat-tree Topology

The preliminary experiment shows the proof-of-concept multipathing using the proposed method. We next evaluated it from the viewpoint of throughput improvement in data center networks. We adopted the method for a 3-level 4-ary fat-tree topology based on the evaluation by Al-Fares *et al.* [8]. The fat-tree topology for this experiment is shown in Fig. 5. There are two types of multiple path sets. For inter-pod communication, there are four multiple paths through four root switches. For intra-pod communication, there are two multiple paths through two aggregation switches in each pod.

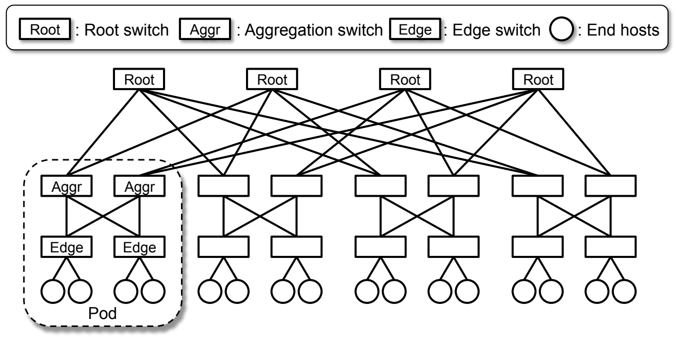


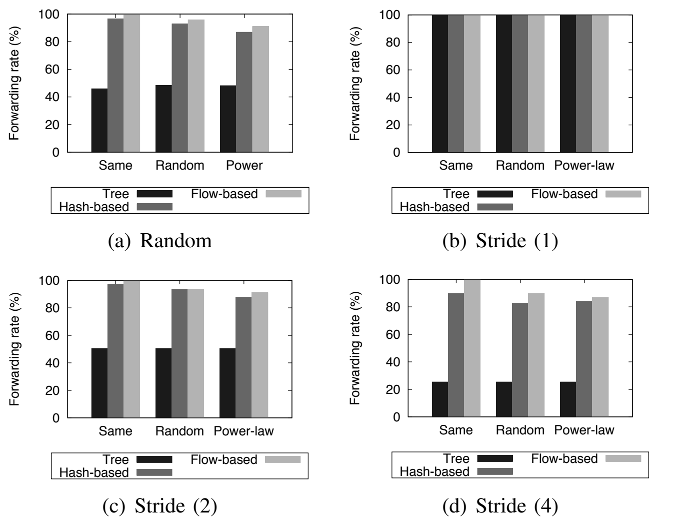
Fig. 5. Experimental topology for the evaluation: 3-level 4-ary fat-tree.

Instead of preparing physical equipment, we simulated the network using ns-3 [18] and Direct Code Execution (ns-3-dce) [19] extension. The ns-3-dce extension can emulate the Linux kernel and its userspace in the ns-3 simulation environment. We simulated the Linux end hosts including iplb kernel module. The fat-tree network comprised Linux hosts as layer-3 switches in the ns-3 simulated network.

The test traffic was the same as for the preliminary tests, namely 20 UDP flows and 1024-byte packets, with GRE being used as the tunneling protocol. The mapping of sender and receiver hosts is configured by benchmark suites proposed by Al-Fares *et al.* [8]. Although they proposed five benchmark suites for their dedicated addressing scheme, we used just two of the benchmarks that were not related to a particular addressing scheme, as follows.

- Random: a host sends to any other host in the network with uniform probability.
- Stride(i): a host with index x will send to the host with index $(x + i) mod 16$.

Fig. 6 shows the results of the experiments with the fat-tree topology. The forwarding rate is a percentage of the sum of received packets to the sum of sent packets (i.e. the aggregated throughput rate). Hash-based and flow-based refer to the balancing algorithms, and tree means there is no multipathing. In the fat-tree topology, there are two paths for communication from two hosts to two hosts in the same pod, and there are four paths for communication from four hosts to four hosts in different pods. Therefore, if the host-to-host mapping is 1-to-1, 100% forwarding rate will be achieved by assigning flows to



(a) Random (b) Stride (1) (c) Stride (2) (d) Stride (4)

Fig. 6. Results of the evaluation for the fat-tree topology. The results are the averages of 30 runs/permutations of the benchmark tests for each flow distribution pattern.

multiple paths completely equally. As a result, when the flow distribution pattern is Same, the flow-based algorithm achieved 100% forwarding rate for all benchmark suites.

For the Stride(1) benchmark shown in Fig. 6(b), all traffic is forwarded by edge switches, so there is no overloaded link. For the Stride(2) benchmark shown in Fig. 6(c), the test traffic is confined to each pod. All test traffic is not routed across root switches. This is similar to four sets of the preliminary test topology, so the results for Stride(2) are the same as the average of those of the preliminary tests shown in Fig. 4. For the Stride(4) benchmark, all traffic is taken through four root switches. Fig. 6(d) shows the results for Stride(4). Without multipathing, all traffic goes via one root switch in accordance with the shortest-path tree. Therefore, the forwarding rate for tree is 25%. By contrast, the hash-based and the flow-based algorithms achieved over 80% forwarding rates. Overall, the proposed method using the flow-based algorithm achieved a forwarding rate of over 85% for all benchmark and flow distribution patterns.

VI. Conclusion and Future work

In this paper, we have proposed a method for multipathing on a layer-3 network with unmodified COTS switches, by a novel usage of common tunneling protocols supported by commodity hardware. We designed and implemented the required modification of end-host software. We demonstrated that our proposed method could use multiple paths efficiently, with a forwarding rate of over 85% for a fat-tree topology. As a result, the method can enhance availability by enabling a layer-3 network design and improve throughput by multipathing for data center networks. Although we have proposed a method for multipathing, the control plane is not discussed in this paper. We are now designing a control plane system to automate configuration and failure recovery, and to offer a more efficient flow assignment. Furthermore, we aim to investigate the variety of commodity products that are capable of supporting the proposed method. Finally, we will evaluate the method for arbitrary topologies.

References

[1] P. Lapukhov, "Building scalable data centers: Bgp is the better igp," https://www.nanog.org/meetings/nanog55/presentations/Monday/Lapukhov.pdf, June 2010, NANOG 55.

[2] A. Andreyev, "Introducing data center fabric, the next-generation facebook data center network," https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/, November 2014, Facebook, Inc.

[3] D. E. 3rd, T. Senevirathne, A. Ghanwani, D. Dutt, and A. Banerjee, "Transparent Interconnection of Lots of Links (TRILL) Use of IS-IS ," IETF, RFC, May 2014.

[4] C. Kim, M. Caesar, and J. Rexford, "Floodless in seattle: A scalable ethernet architecture for large enterprises," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, ser. SIGCOMM '08. New York, NY, USA: ACM, 2008, pp. 3–14.

[5] M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. Zhani, "Data center network virtualization: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 2, pp. 909–928, Second 2013.

[6] 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 *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 63–74.

[7] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 17–17.

[8] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, ser. SIGCOMM '08. New York, NY, USA: ACM, 2008, pp. 63–74.

[9] 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 the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 39–50.

[10] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008.

[11] 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 on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 51–62.

[12] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "Spain: Cots data-center ethernet for multipathing over arbitrary topologies," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 18–18.

[13] P. Garg and Y. Wang, "draft-sridharan-virtualization-nvgre-07.txt," IETF, Internet Draft, November 2014.

[14] E. E. Crabbe, E. L. Yong, and E. X. Xu, "draft-ietf-tsvwg-gre-in-udp-encap-03," IETF, Internet Draft, October 2014.

[15] J. Gross, T. Sridhar, P. Garg, C. Wright, I. Ganga, P. Agarwal, K. Duda, D. Dutt, and J. Hudson, "draft-gross-geneve-02," IETF, Internet Draft, October 2014.

[16] J. Salim, H. Khosravi, A. Kleen, and A. Kuznetsov, "Linux Netlink as an IP Services Protocol," IETF, RFC 3549, July 2003.

[17] Linux Foundation, "iproute2," http://http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2.

[18] ns-3 project, http://www.nsnam.org/.

[19] H. Tazaki, F. Uarbani, E. Mancini, M. Lacage, D. Camara, T. Turletti, and W. Dabbous, "Direct code execution: Revisiting library os architecture for reproducible network experiments," in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '13. New York, NY, USA: ACM, 2013, pp. 217–228.