DomainFlow: Practical Flow Management Method using Multiple Flow Tables in Commodity Switches

Yukihiro Nakagawa Kazuki Hyoudou Chunghan Lee
Shinji Kobayashi Osamu Shiraki Takeshi Shimizu
Fujitsu Laboratories Ltd., Kawasaki, Kanagawa 211-8588, Japan
{yukihiron, hyoudou.kazuki, lee.chunghan, koba, shiraki, shimizu.takeshi}@jp.fujitsu.com

ABSTRACT

A scalable network with high bisection bandwidth and high availability requires efficient use of the multiple paths between pairs of end hosts. OpenFlow is an innovative technology and enables fine-grained, flow level control of Ethernet switching. However, the flow table structure defined by OpenFlow is not hardware friendly and the scalability is limited by the switch device. OpenFlow is also not sufficient for fast multipath failover. To overcome these limitations, we propose DomainFlow in which the network is split into sections and exact matches are used where possible to enable practical flow management using OpenFlow for commodity switches. We applied a prototype of DomainFlow to multipath flow management in the Virtual eXtensible LAN (VXLAN) overlay network environment. The total number of flow entries was reduced to 1/128 using currently available commodity switches, which was not possible before.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design; C.2.3 [**Computer Communication Networks**]: Network Operations

General Terms

Experimentation, Design, Management

Keywords

Ethernet switch, OpenFlow switch, Overlay networks

1. INTRODUCTION

The ratio of traffic volume between servers in a data center to traffic entering/leaving a data center is around 4:1, and the demand for bandwidth between servers inside a data center is rapidly increasing [1]. To satisfy this requirement, a scalable network with high bisection bandwidth and high availability is required, and efficient use of multiple paths between pairs of end hosts is becoming more important.

OpenFlow [2] is an innovative technology in software defined networking (SDN). However, the flow table structure defined by OpenFlow is hugely flexible and not hardware friendly. The Forwarding Abstractions WG (FAWG) of Open Networking Foundation (ONF) [3] is currently working on table type patterns (TTPs) to enhance adoption of the OpenFlow standard on hardware forwarding targets. The TTPs enable pre-run-time description of switch-level behavioral abstraction so that the controller only uses a constrained subset of the table-based OpenFlow standard switch model.

Even though OpenFlow supports multiple flow tables and the pre-run-time description is available, the central controller needs to use switch hardware resources or multiple flow tables efficiently for constructing a scalable network with high bisection bandwidth and high availability. Since OpenFlow enables fine-grained, flow level control of Ethernet switching, we encounter a lack of hardware resources, which limits scalability in a data center network. In addition to this limitation, OpenFlow is not sufficient for fast failover for high availability in the multipath.

To overcome these limitations, we propose DomainFlow, a practical flow management using OpenFlow for commodity switches. With DomainFlow, we exploit the fact that switches contain exact matching tables for the forwarding databases. The key ideas of our scheme are (i) use exact matching where possible, (ii) split network into sections to allow exact matches to be used more often. The flow model consists of two domains, and a controller configures the boundary on a per flow basis. A flow is controlled in each domain by either of exact matching rules or wild-carded matching rules to use hardware resources within the semantics of OpenFlow-capable switches.

We applied a prototype of DomainFlow to a multipath flow management in the Virtual eXtensible LAN (VXLAN) overlay network environment. The VXLAN is a Layer 2 overlay scheme over Layer 3 networks to enable dynamic infrastructure and increase virtual machine mobility [4, 5, 6]. Our prototype demonstrated that multipath flow management is possible with a small number of flow entries using currently available commodity switches.

This paper is organized as follows: Section 2 gives background and Section 3 describes our proposed flow management method called DomainFlow. Section 4 describes our application of DomainFlow, and Section 5 describes our prototype and evaluations. Section 6 describes related work, and Section 7 gives conclusions.

CoNEXT'13, December 9-12, 2013, Santa Barbara, California, USA.

http://dx.doi.org/10.1145/2535372.2535406 .

Table 1: 10GbE Switch Specifications

Switch	Dell S5000	Cisco Nexus3064	Arista 7050S-64	Juniper QFX3500
Switch capability	1.28Tbps	1.28Tbps	1.28Tbps	1.28Tbps
VLAN table	4k	4k	4k	4k
Mac table	128K	128k	128K	120K
L3 routing table	16k	16K	16K	8k
ACL	2k ingress/ 1k egress	2k ingress/ 1k egress	3k	1.5k

2. BACKGROUND

In this section we explain why a practical flow management method is required for constructing scalable networks with high bisection bandwidth and high availability in data centers.

2.1 TCAM for Wild-carded Matching Rule

OpenFlow requires wild-carded matching rules. Typically, Ethernet switches use ternary content addressable memories (TCAMs) to implement wild-carded OpenFlow rules.

However, TCAM consumes much higher area and power per entry than static random-access memory (SRAM). Therefore, on-chip TCAM sizes are typically limited to a few thousand entries. As shown in Table 1, high-performance, commodity 10Gb Ethernet (10GbE) switches generally implement an access control list (ACL) flow table using TCAM and support a small number of ACL entries [7, 8, 9, 10]. Therefore, the scalability of OpenFlow-based SDN is limited due to switch hardware resources.

2.2 Overhead for Flow Level Management

OpenFlow features wild-carded matching rules, enabling fine-grained, flow level control of Ethernet switching. Efficient multipath usage is possible by selecting path for every flow [11].

However, the number of matching rules increases when we specify each flow for multipath distribution. For example, assuming we have h hosts and v virtual machines per host, OpenFlow requires h*(h-1)*v*v wild-carded matching rules for traffic distribution as shown in Table 2 and described later in detail in Section 5.

As a result, OpenFlow-based path selection is limited due to the switch hardware resources. OpenFlow provides a group action to select path from multiple output ports, but the distribution algorithm is not defined in the specification. Therefore, this feature cannot be used for flow level management by the controller.

2.3 Fast Failover for High Availability

When a link failure occurs, a fast failover is necessary to achieve high availability. OpenFlow defines a fast failover mechanism for the controller to specify the alternate port, enabling the switch to change forwarding without requiring a round trip to the controller.

However, it only works with a switch in which a link failure is detected directly and does not work with multiple switches in the multipath.

3. DOMAINFLOW

With DomainFlow, we exploit the fact that switches contain exact matching tables and efficiently use them.

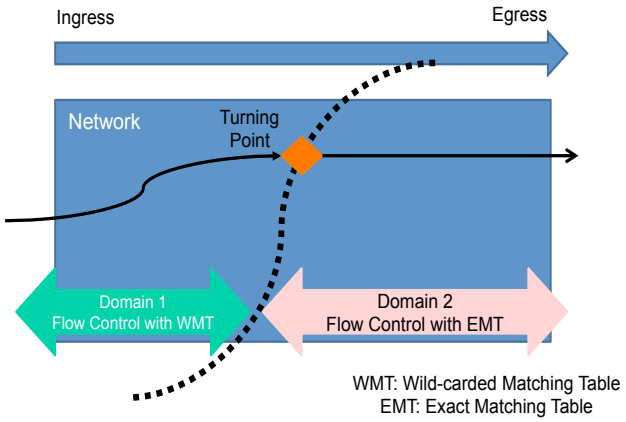

Figure 1: DomainFlow Flow Model

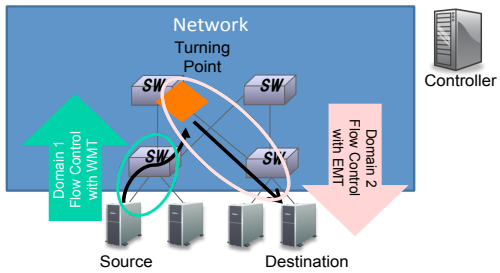

Figure 2: Typical Use Case of DomainFlow

3.1 DomainFlow Flow Model

Figure 1 shows the DomainFlow flow model in a network. A flow control scheme is defined for Domains 1 and 2. In Domain 1, a flow is controlled with a wild-carded matching table (WMT). In Domain 2, a flow is controlled with an exact matching table (EMT). A turning point corresponds to the flow boundary of Domains 1 and 2. The controller configures the turning point on a per flow basis.

In Domain 1, the central controller installs flow entries for path selection in the WMT. A destination lookup failure (DLF) occurs in the EMT then the flow is handled by the WMT. In Domain 2, the controller installs flow entries in the EMT for unicast, broadcast, and multicast packets. If a DLF occurs, it is handled by the controller.

In the OpenFlow protocol, there are two programming modes in the flow table modification: reactive and proactive. DomainFlow can be applied to either of programming mode. We focus on the proactive mode in this paper for reducing the flow setup overhead.

It is possible to cascade the DomainFlow flow models, but a single stage of the model is effective enough for our multipath application as shown in Section 5.

3.2 DomainFlow Use Case

DomainFlow can be used in the flow management in various types of applications such as path selection in multipath networks with tree structure, end-host mode to represent servers, and forwarding of FCoE (Fibre Channel over Ethernet) traffic to FCF (FCoE Forwarder).

In multipath networks, one of multiple paths can be selected using WMT toward the root of a tree network while

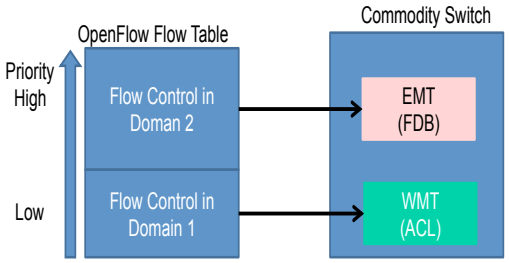

Figure 3: DomainFlow Flow Table Structure

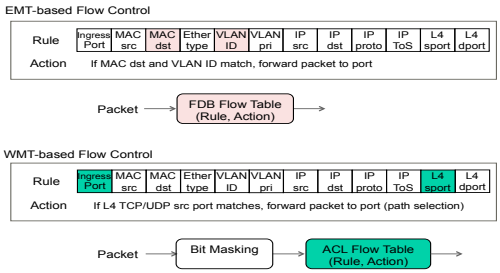

Figure 4: DomainFlow Flow Table Example

the packet is forwarded using EMT from the root to a designated server as described briefly later in this section and more detail in Section 4. In the end-host mode, a given server interface uses a given switch uplink regardless of the destinations. An adjacent switch only stores MAC addresses of servers connected to the downlink. And a packet from a server is forwarded to an uplink using WMT while a packet is forwarded from an uplink to a server using EMT. In FCoE, a packet is forwarded to FCF using EMT at intermediate switches. At a turning point, the packet can be short-cut forwarded using WMT as described in our related work [12].

Figure 2 shows a typical use case of DomainFlow in multipath networks. Domain 1 is defined in which no destination exists at the server-facing ports of a switch. Domain 2 is defined in which a destination exists at the server-facing ports of a switch. The upward flow from a source server is controlled with the WMT towards the turning point. The downward flow is controlled with the EMT towards a destination server.

3.3 DomainFlow Flow Table Structure

Figure 3 shows the DomainFlow flow table structure and Figure 4 shows examples of matching rules and actions. A commodity switch normally has both an EMT and WMT. The EMT is implemented as a forwarding database (FDB) using SRAM and has a large number of entries. The WMT is implemented as an access control list (ACL) using TCAM and has a small number of entries.

The priority of EMT is higher than that of WMT. In other words, wild-carded matching is executed with WMT only when no matching entry was found in the EMT. We use bit masking to further reduce the number of WMT (ACL) entries so that it does not become proportional to the number of virtual machines but proportional to the number of uplinks.

By using the flow table structure, we can use the large-capacity EMT (FDB) and map the flow model to commodity switch hardware within the semantics of an OpenFlow-capable switch. The EMT can be applied to any exact matches including IP addresses.

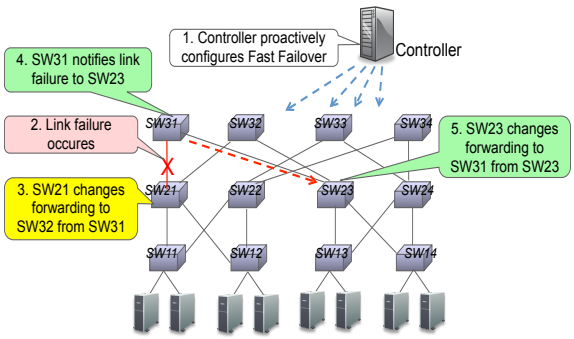

Figure 5: Fast Failover in Multipath

3.4 Fast Failover in Multipath

Figure 5 shows fast multipath failover. We use the fast failover mechanism of OpenFlow to enable the switch to change forwarding without requiring a round trip to the controller. The controller proactively configures a group type of Fast Failover so that a switch can use the first live port when a link failure occurs. However this mechanism works only when a switch detects a link failure on the directly connected link. Therefore we notify other affected switches about the link failure so that they can appropriately change forwarding without intervention of the controller. For example, if the link between SW21 and SW31 failed, SW21 detects the link failure and changes forwarding to SW32. Then SW31 notifies SW23 of the link failure and SW23 changes forwarding to SW32.

4. APPLICATION

We applied the DomainFlow method to multipath flow management in the VXLAN overlay network environment to enable network virtualization. It should be noted that DomainFlow can also be applied to the flow management of the non-overlay network environment.

4.1 VXLAN Packet Encapsulation

Overlay networks construct tunneling of Ethernet frames by encapsulating the original frames with additional headers. Figure 6 shows VXLAN packet handling and packet format in which the additional headers are Outer Ethernet, Outer IP, Outer UDP, and VXLAN. The outer IP header includes the IP address of the VXLAN tunnel end point (VTEP), which originates and/or terminates VXLAN tunnels. The VXLAN header includes 24 bits of VXLAN network identifier (VNI) or VXLAN segment ID and allows up to 16 million VXLAN overlay networks or segments to coexist within the same physical network infrastructure.

4.2 Multipath Traffic Distribution

In overlay networks, traffic from/to virtual machines is aggregated/disaggregated at a TEP, or VTEP in the case of VXLAN. Therefore, it is important to distribute traffic in multipath networks reflecting original payloads for high performance. To achieve this, the VXLAN specification recommends the UDP source port of the outer header to be a

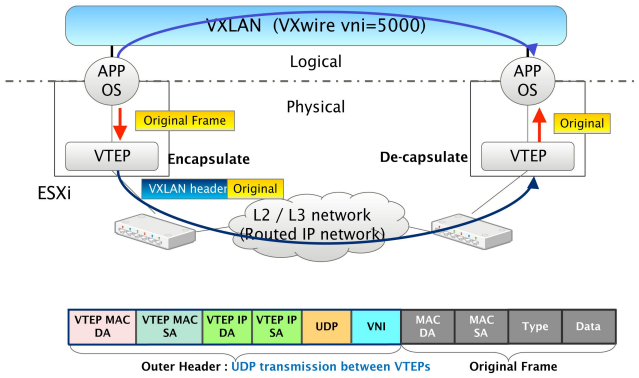

Figure 6: VXLAN Overlay Network

hash of the inner Ethernet frame's headers. This is to enable a level of entropy for ECMP/load balancing of the virtual machine to virtual machine traffic across the VXLAN overlay in the multipath network.

Multipath TCP [13] uses the multipath by changing the TCP source port for additional paths. In the VXLAN overlay network, the TCP source port of the inner header must be included in the hash calculation, and the UDP source port of the outer header should be used for path selection to increase entropy for load balancing.

4.3 DomainFlow Flow Management

In the VXLAN overlay network, the original packet is encapsulated at the VTEP. The outer Ethernet and IP headers have MAC and IP addresses of the VTEP. If a physical switch cannot detect the overlay network, the traffic may not be distributed well in the multipath. Therefore, the scalability of an OpenFlow-based SDN is limited due to inefficient multipath usage in the physical network.

To avoid this limitation, a physical switch must select a path by using a matching rule on the UDP source port of the outer header. However the number of calculated hash values could be very large depending on the number of flows.

Figure 7 shows examples of path selections in DomainFlow. The controller proactively installs FDB entries with MAC addresses of the hosts at server-facing ports of a switch. For example, S1 and S2 are installed in the FDB flow table at Switch 1, and S1, S2, S3, and S4 are installed at Switch 3. The controller also proactively installs ACL entries for path selection to select one of uplinks. For example, bit-masked UDP source port=0 and 1 are installed in the ACL flow table at Switch 1.

Figure 8 shows an algorithm of EMT (FDB) configuration by the central controller in pseudo code. Given topology information T, EMT configuration flow list L is returned. SearchTree(n) function returns server list under the given node n and it returns node-id when the node is a server. Therefore EMT configuration flow list at each switch level is obtained by calling recursively SearchTree function from a root node.

VTEP1 sends Packet (A) to VTEP2. Switch 1 receives the packet at Port P1. Switch 1 finds the MAC address S2 of VTEP2 in the FDB flow table and outputs the packet to Port P2. VTEP1 sends Packet (B) to VTEP3. Switch 1 receives the packet at Port P1. Switch 1 does not find

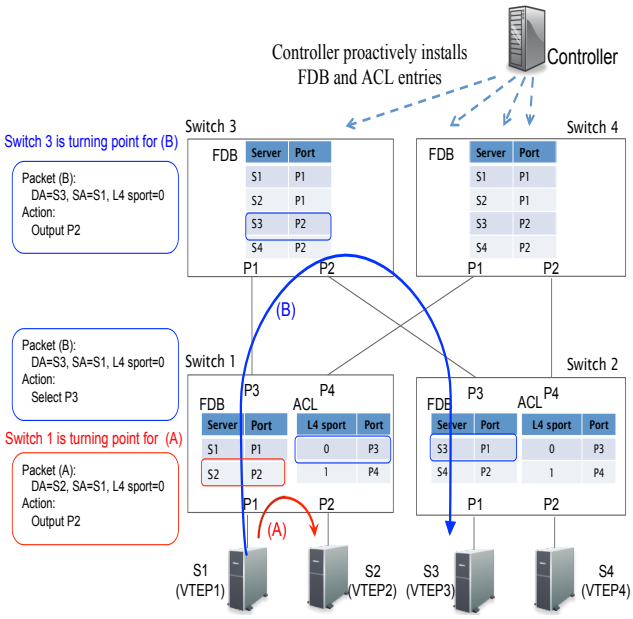

Figure 7: DomainFlow Path Selection

```
Algorithm: Make Flow List for EMT
Input: Network topology tree T
1.flow list L = {}
2.For each root-node r of T // multiple root exist in Fat-tree
3.  SearchTree(r)
4.Return L
function SearchTree(n)
5.If(node-type(n) == Server) Return { node-id(n) } // leaf machine
6.Else S = {}
7.  For each subtree v of n
8.    p = connected-port(n, v) // port No. of n, which is connected to v
9.    Sv = SearchTree(v)
10.   For each element e of Sv
11.  L = L + { { id=>node-id(n), op=>Add, rule=e, act=Forward to p } }
12.  S = S + Sv
13.Return S
```

Figure 8: EMT Configuration by Controller

the MAC address S3 of VTEP3 in the FDB flow table, so Switch 1 searches the ACL flow table. Switch 1 finds UDP source port 0 with bit mask 0x0001 in the ACL flow table and outputs the packet to the selected path, Port P3. We note one bit is sufficient to select one of two uplinks. Switch 3 receives the packet at Port P1. Switch 3 finds the MAC address S3 of VTEP3 in the FDB flow table and outputs the packet to Port P2. Switch 2 receives the packet at Port P3. Switch 2 finds the MAC address S3 of VTEP3 in the FDB flow table and outputs the packet to Port P1.

For fast failover, the central controller specifies an alternate port and configures it proactively using OpenFlow Fast Failover mechanism. When a link failure is detected, the

```
struct ofp_action_select_vendor {
  uint16_t type;          /* OFPAT_VENDOR. */
  uint16_t len;           /* Length is 8. */
  uint16_t port;          /* Output port. */
  uint16_t bitmask;       /* Bit mask for L4 src port */
  };
OFP_ASSERT(sizeof(struct ofp_action_select_vendor) == 8);
```

Figure 9: OpenFlow Vendor Action

switch locally modifies the ACL flow table to change forwarding. Suppose the link between Switches 1 and 3 failed. Switch 1 changes forwarding to Switch 4 from Switch 3. Switch 3 notifies Switch 2 about the link failure and Switch 2 changes forwarding to Switch 4 from Switch 3.

As described above, we need to bit mask the matching field of a packet to reduce the number of required ACL entries for path selection. However, OpenFlow actions do not provide the bit masking capability on the UDP source port. Therefore, we extended the output action to specify the bit mask. Figure 9 shows the vendor action for path selection.

5. PROTOTYPE AND EVALUATION

We built a prototype of DomainFlow and confirmed operations in a VMware virtualization environment.

5.1 Switch Prototype

Figure 10 shows our 10GbE/40GbE switch prototype and the experimental setup. We used Fujitsu Converged Fabric (C-Fabric) switches [14] as a platform and enhanced the firmware to evaluate the DomainFlow-capable OpenFlow switch. Our initial prototype of the OpenFlow switch is based on OpenFlow 1.0 [15] and includes an OpenFlow agent running on a dedicated external server connected to the switch. The agent communicates with the controller over the secure channel and it configures the switch using the enhanced Command Line Interface. If the agent receives a rule that exactly matches the destination MAC address and VLAN ID, it installs the rule in the FDB flow table unless it fails due to hashing collisions or overflow. Otherwise, it installs the wild-carded rule in the ACL flow table. In vender extension for path selection, the agent installs a rule on the bit-masked UDP source port in the ACL flow table. The path selection rule is applied for the bit-masked UDP source port of an incoming packet when a DLF occurs in the FDB flow table.

5.2 Preliminary Evaluation Results

The experimental setup consisted of four Fujitsu PRIMERGY BX920S1 servers (Intel Xeon X5570 2.93GHz CPU and 32GB Memory, Intel 82599 10GbE NIC) and two prototype switches. We confirmed basic operations of DomainFlow in VMware vCloud Suite 5.1, which supports the VXLAN overlay network in the ESXi hypervisor. In Domain 1, a packet is forwarded based on the path selection rule in the ACL flow table, distributing traffic from virtual machines on two links between switches. In Domain 2, the packet is forwarded to virtual machines based on the FDB flow table. Our prototype showed that the DomainFlow method can be implemented using a currently available commodity switch with a small number of flow table entries.

5.3 Flow Table Efficiency

Table 2 lists the number of flow table entries in our testbed and a testbed with the fat tree topology shown in Figure 5. In the table, the traditional method means matching rules are installed in the TCAM-implemented ACL flow table only. In our testbed, the traditional method requires 768 ACL entries and DomainFlow requires just 2 FDB entries and 4 ACL entries, reducing the total number of flow entries to 1/128. In the fat tree, the traditional method requires 3,584 ACL entries, which is not possible with commodity switches, and DomainFlow requires 8 FDB entries and 4 ACL entries, reducing the total number of flow entries to 1/298. As shown in Figure 11, the number of flow entries is not proportional to the number of virtual machines but proportional to the number of uplinks in DomainFlow.

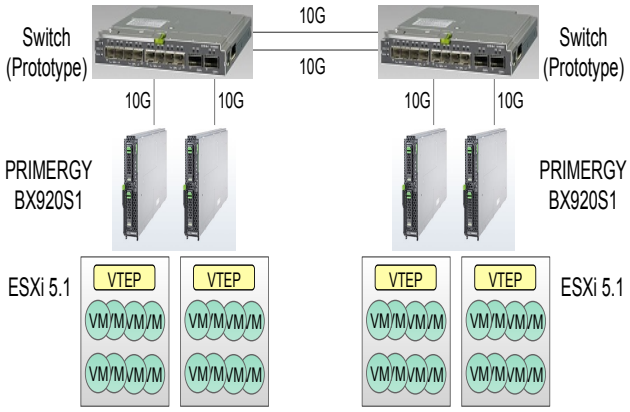

Figure 10: Evaluation System

Table 2: Number of Flow Table Entries

	Testbed (Figure 10)		Fat tree (Figure 5)	
	Traditional	DomainFlow	Traditional	DomainFlow
# of VTEPs or # of hosts (h)	4		8	
# of VMs per host (v)	8		8	
max. # of hosts underneath of switch (s)	2		8	
# of server-facing ports of switch (p)	2		2	
# of multipaths per switch (m)	2		2	
max. # of paths on switch (t= h*(h-1)*v*v)	128		3,584	
max. # of FDB entries per switch (f)	-	2 (= s)	-	8 (= s)
max. # of ACL entries per switch (a)	768 (= t)	4 (= m*p)	3,584 (= t)	4 (= m*p)
Total # of Flow table entries per switch (f+a)	768	6	3,584	12

5.4 Future Work

The OpenFlow 1.3 protocol [16] uses OpenFlow extensible match (OXM) format and supports bit masking. However it disallows bit masking of the UDP source port. We believe that path selection with bit masking on the UDP source port should be included in an updated version of the OpenFlow specifications.

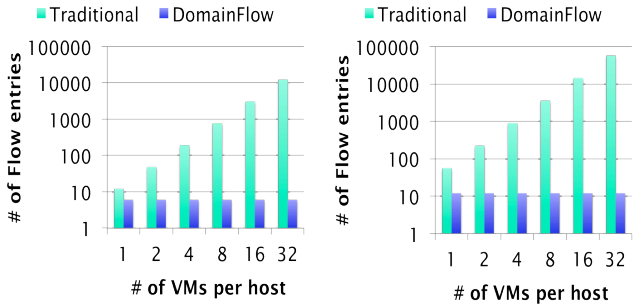

Figure 11: VMs and Flow Table Entries

6. RELATED WORK

In our previous work [17], we developed 10GbE Layer2 switches for server edges [12, 18, 19] and proposed using OpenFlow to solve the scalability problem of IP multicast. In that environment, the FDB is used as a flow table, but we did not introduce the domain based management method.

Curtis et al. [20] proposed DevoFlow to reduce the use of TCAM. The switch locally "clones" a wild-carded rule to create an exact matching rule. The rule cloning requires an enhancement of switch devices.

Al-Fares et al. [11] proposed Hedera, which uses OpenFlow for flow management as usual. It detects so-called elephant flows [1] and reduces their performance impact. It uses ECMP/load balancing for the majority of flows. This technique can be combined with DomainFlow.

Stephens et al. [21] proposed PAST to enable scalable OpenFlow-based SDN using inexpensive commodity hardware. PAST uses the FDB as a flow table for per-address spanning tree routing and does not use the ACL flow table.

Bosshart et al. [22] proposed the RMT (reconfigurable match tables) to allow the forwarding plane to be changed in the field without modifying hardware. RMT uses both wild-carded tables and exact matching tables.

7. CONCLUSIONS

We proposed DomainFlow with fast failover to enable practical flow management using OpenFlow for commodity switches. In the flow model, a flow is controlled by bit masking and wild-carded matching rules in Domain 1 and controlled by exact matching rules in Domain 2 for efficient use of switch hardware resources. We applied a prototype of DomainFlow to multipath flow management in the VXLAN overlay network environment. The total number of flow entries was reduced to 1/128 using currently available commodity switches, which was not possible before. For future work, we will use the benefits of OpenFlow for constructing workload-aware high performance networks.

8. ACKNOWLEDGMENTS

We would like to thank Dejan Kostic for shepherding the final version, and the anonymous reviewers whose comments helped us improve the paper. We are grateful to Takeshi Horie and Jun Tanaka for supporting this work and thank Shinji Yamashita and Masakuni Kawata for their helpful feedback on our prototyping.

9. REFERENCES

[1] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta. VL2: A Scalable and Flexible Data Center Network. In *SIGCOMM 2009*, August 2009.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. In *SIGCOMM CCR*, 2008.

[3] Open Networking Foundation. Charter: Forwarding Abstractions Working Group, April 2013.

[4] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright. VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. *Internet Draft*, October 2013.

[5] M. Sridharan, A. Greenberg, Y. Wang, P. Garg, N. Venkataramiah, K. Duda, I. Ganga, G. Lin, M. Pearson, P. Thaler, and C. Tumuluri. NVGRE: Network Virtualization using Generic Routing Encapsulation. *Internet Draft*, August 2013.

[6] B. Davie and J. Gross. A Stateless Transport Tunneling Protocol for Network Virtualization (STT). *Internet Draft*, September 2013.

[7] DELL. *Specifications: Dell S5000 Unified Storage/Top-of-Rack Switch*, 2013.

[8] Cisco. *Nexus 3064 Switch Data Sheet*, 2012.

[9] Arista. *7050 Series 10/40G Data Center Switches Data Sheet*, 2013.

[10] Juniper. *QFX3500 Switch Datasheet*, 2012.

[11] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *NSDI '10*, April 2010.

[12] O. Shiraki, Y. Nakagawa, K. Hyoudou, S. Kobayashi, and T. Shimizu. Managing Storage Flows with SDN Approach in I/O Converged Networks. In *IEEE MENS 2013*, December 2013.

[13] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving Datacenter Performance and Robustness with Multipath TCP. In *SIGCOMM 2011*, August 2011.

[14] Fujitsu. Fujitsu Develops New Architecture for Network-Wide Optimization of ICT Platforms, May 2013.

[15] The OpenFlow Switch Consortium. *OpenFlow Switch Specification Version 1.0.0*, December 2009.

[16] Open Networking Foundation. *OpenFlow Switch Specification Version 1.3.1*, September 2012.

[17] Y. Nakagawa, K. Hyoudou, and T. Shimizu. A Management Method of IP Multicast in Overlay Networks using OpenFlow. In *HotSDN '12*, August 2012.

[18] T. Shimizu, Y. Nakagawa, S. Pathi, Y. Umezawa, T. Miyoshi, Y. Koyanagi, T. Horie, and A. Hattori. A Single Chip Shared Memory Switch with Twelve 10Gb Ethernet Ports. *Hot Chips 15*, August 2003.

[19] Y. Nakagawa, T. Shimizu, T. Horie, Y. Koyanagi, O. Shiraki, T. Miyoshi, Y. Umezawa, A. Hattori, and Y. Hidaka. *Energy-Aware Switch Design*. IGI Global, Pennsylvania, 2012.

[20] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. DevoFlow: Scaling Flow Management for High-Performance Networks. In *SIGCOMM 2011*, August 2011.

[21] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter. PAST: Scalable Ethernet for Data Centers. In *CoNEXT 2012*, December 2012.

[22] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. Forwarding metamorphosis: fast programmable match-action processing in hardware for sdn. In *SIGCOMM 2013*, August 2013.