WHITE PAPER

Engineered Elephant Flows for Boosting Application Performance in Large-Scale CLOS Networks

Abstract. Data Center networks built using Layer 3 ECMP-based CLOS topology deliver excellent bandwidth and latency characteristics, but they can also cause short-lived latency sensitive mice flows to be choked by long-lived bandwidth-hungry elephant flows degrading application performance. A representative traffic engineering application is implemented on available hardware switches using Broadcom's OpenFlow 1.3.1 compliant OpenFlow Data Plane Abstraction (OF-DPA) Specification and Software, and an open source OpenFlow 1.3.1 agent (Indigo 2.0) and controller (Ryu). This application demonstrates the ability to implement scalable and workload-sensitive Layer 3 ECMP-based CLOS networks using OpenFlow.

March 2014

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

OF-DPA-WP102-R • March 2014

Connecting everything®

Introduction

OpenFlow, as defined by the Open Networking Foundation (ONF), promises to accelerate innovation in networking. Since the introduction of OpenFlow, the quality and quantity of networking innovation and research has accelerated dramatically. Much of this has focused on data center traffic engineering, with emerging use cases evolving for carrier or service provider networks.

OpenFlow v1.0 has been used in both academic research and commercial networks. It relies on a single table model based on Ternary Content Addressable Memory (TCAM). This has resulted in limited scale and deployments because TCAM is an expensive resource in switch ASICs when it comes to scale of network topologies and number of flow entries that can be supported. OpenFlow v1.3.1 enables use of multiple tables in packet processing pipelines available in existing Broadcom StrataXGS® switch ASICs to alleviate or resolve the cost-effective scaling challenges with OpenFlow v1.0-based switches.

In the recent past, OpenFlow research efforts have concentrated on algorithm development and use of software-based switches. While desirable to prove benefits using real networks built with physical switches, the lack of available representative hardware switches has limited data gathering and research. As a result, most have used simulated network environments. There are many significant network issues that arise with scale and performance measurements in real networks with real traffic that the industry has not been able to explore.

Broadcom’s OpenFlow Data Plane Abstraction (OF-DPA) is based on OpenFlow v1.3.1 and addresses the challenges of cost-effective scalability and supporting commercial and research use cases with available switch hardware. It does so by enabling OpenFlow controllers to program the switch hardware that is deployed in the majority of data centers today. OF-DPA provides a hardware abstraction layer (HAL) for Broadcom StrataXGS switches; it also provides an API library for OpenFlow 1.3.1 compliant agents. It is intended for both real network deployments and enabling research networks. OF-DPA is available as an easily deployable development package as well as a turnkey reference package which can be installed on widely available “white box” switches to enable network innovation.

This white paper demonstrates the advantages of using an OF-DPA-based OpenFlow 1.3.1 switch in a representative traffic engineering application. This application is based on real implementation on available hardware switches using OF-DPA software, OpenFlow 1.3.1 open source agent, controller, and referenced research papers.

This paper is organized as follows. “Data Centers and ECMP-Based CLOS Networks” provides background on the form of data center network infrastructure that is used for this paper. “Congestion Scenario with Mice and Elephant Flows ” describes the problem. “Using OpenFlow 1.3.1 to Build ECMP Networks ” discusses solution approaches. “Using OpenFlow 1.3.1 to Traffic Engineer Elephant Flows ” describes how to use OpenFlow 1.3.1 to build the network of “Data Centers and ECMP-Based CLOS Networks”. “Open Hardware and Software Implementation”describes how to use OpenFlow 1.3.1 to implement the solution approaches in “Using OpenFlow 1.3.1 to Build ECMP Networks ”.

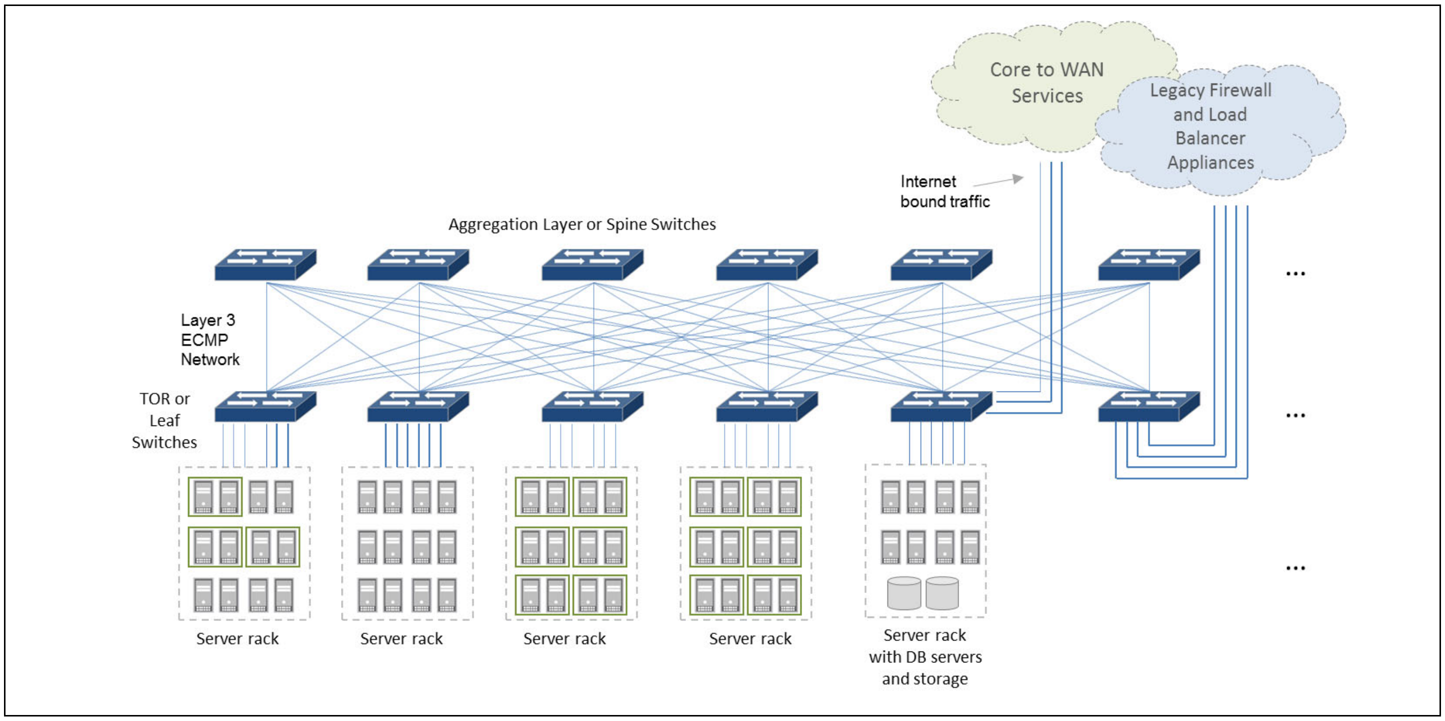
This document assumes familiarity with OpenFlow 1.3.1 and related Software Defined Networking (SDN) technologies and concepts (centralization, open API, separation of control and data planes, switch, agent and controller, etc.).

The OF-DPA v1.0 specification and the turnkey reference package are openly available on GitHub at https://github.com/Broadcom-Switch/of-dpa.

Data Centers and ECMP-Based CLOS Networks

Data centers and evolving carrier networks require a scalable infrastructure capable of supporting multiple application workloads. To achieve this goal, a number of large scale data center operators have chosen to deploy folded CLOS network topologies because of their non-blocking multipath and resiliency properties. Such network topologies are desirable for applications that generate east-west (server to server or server to storage) traffic. A representative two-level folded CLOS topology is shown in Figure 1. In this simple "leaf-and-spine" design, servers are attached to edge (top of rack) or leaf switches, and are interconnected in turn by aggregation or spine switches.

FIGURE 1: Two-Level Folded CLOS Network Topology Example



Servers in racks connect to the network using the leaf switches. Connectivity to the WAN and other data centers is provided using the Core to WAN services where Core switches reside. Firewall, load balancing, and other specialized network services are provided by appliances connected to the CLOS network.

A common method for utilizing the inherent multipath in a large CLOS network is to use Layer 3 routing everywhere, including the leaf switches. With each point-to-point link between switches in a separate subnet, standard equal cost multipath (ECMP) routing can be used to distribute traffic across links. This is attractive since cost-effective Layer 3-based leaf and spine switches provide hardware-based hashing for assigning packets to links and support standard distributed routing protocols such as OSPF or BGP.

This Layer 3 routing in a large CLOS network requires switches to maintain large numbers of subnet routes, but relatively few Layer 2 addresses, for directly attached switches and servers. Many cloud data centers use Layer 2 overlay tunnels to create virtual tenant or application networks that allow tenant or application servers to be deployed anywhere, independent of underlay topology.

Congestion Scenario with Mice and Elephant Flows

While CLOS networks that are built using Layer 3 routing and ECMP deliver excellent bandwidth and latency characteristics for east-west traffic in the data center, the main problem with this approach is that CLOS networks built using Layer 3 routing and ECMP are insensitive to workload. Depending on the application, data center traffic tends to exhibit a mix of latency sensitive short-lived flows (referred to colloquially as mice), and bandwidth intensive longer lived flows (elephants) for which throughput is more important than latency. Applications such as Hadoop (Reference [10] on page 14) that use MapReduce techniques cause traffic patterns that include elephants and mice. Elephant flows may also consist of large data transfer transactions and peer-to-peer file sharing, while mice flows are experienced in on-line gaming, small-sized web requests, multimedia broadcasting, and voice over IP. Typically, fewer than 10% of flows are elephants, but they account for over 80% of traffic Reference [1] on page 14.

ECMP is designed to optimize for latency, and hash-based multipath selection assumes traffic is distributed more or less uniformly Reference [2] on page 14 and Reference [3] on page 14. Mice flows tend to be very bursty and short lived, performing well with stateless, hash-based multipathing over equal cost paths. However, stateless, hash-based multipathing can cause long-lived TCP flows to “pile up” on a particular path even though other paths might be free, filling network buffers along that path which can result in congestion events.

FIGURE 2: ECMP CLOS Network Topology with Elephant Flows

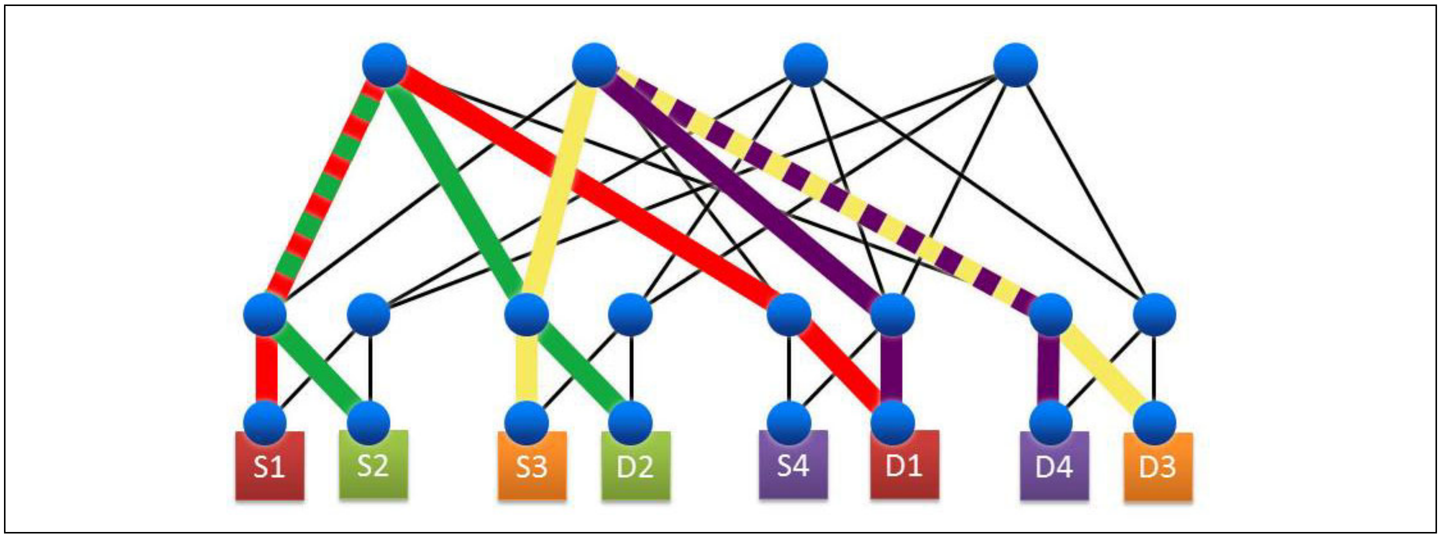


Figure 2 shows a scenario where an ECMP-based network is congested because of elephant flows. S1 through S4 are sources for the flows in the network, while D1 through D4 are the destinations. As can be seen in Figure 2, the red and green flows share links, so do the yellow and purple flows. Also, there are many links that are not utilized at all, and as a result, long lived flows can constrain the short lived flows and cause significant detriment to application performance. At the same time, a large percentage of the bisection bandwidth may be wasted.

The answer is to augment ECMP with some form of traffic engineering. Ideally, link assignments would take into account loading on links and internal switch buffer state to avoid congestion. But even with hardware support for load balancing, the control plane functionality needed is not well understood. In general, it has proven difficult to design algorithms that adaptively respond quickly to changing network traffic patterns based solely on local knowledge. As a result, traffic engineering until recently has remained an interesting research area.

Elephant Flows and Traffic Engineering

Traffic engineering for elephant flows using OpenFlow has been extensively reported in the research literature Reference [4] on page 14, Reference [5] on page 14, Reference [6] on page 14, Reference [7] on page 14, and Reference [8] on page 14. These generally use two steps: identifying the problem flows and modifying the forwarding to treat problem flows differently.

While identification of elephant flows is not the focus of this paper, the techniques for detection are worth summarizing. The most reliable way is to simply have the applications determine them a priori, as in Reference [9] on page 14 or Reference [11] on page 14. For example, MapReduce phases are orchestrated by control nodes capable of estimating traffic demands in advance, including addresses and other identifying information. Another way is to dynamically detect long lived flows by maintaining per-flow traffic statistics in the network and declaring a flow to be an elephant flow when its byte count exceeds some threshold. Alternatively, traffic sampling could be used, for example, by sFlow Reference [12] on page 14. Passive traffic snooping approaches tend to be less reliable in that the detection time has been reported to vary from the order of 100 ms to many seconds due to polling delays. By the time the flows are identified, congestion events may already have taken place. A variant of this is to collect byte counts at the end hosts by monitoring socket buffers (Reference [5] on page 14), and declaring a flow to be an elephant if it transfers more than 100 MB, and then marking the packets that belong to elephant flows (e.g., using the DSCP field in the IP header).

Once identified, OpenFlow can be used to give elephant flows differential forwarding treatment. Hedera Reference [4] on page 14 load-balanced the elephant flows as a separate aggregate from mice flows. Hedera found it acceptable to load-balance the mice randomly using hashing. Since they were using OpenFlow 1.0, this was applied at the controller for every new flow. Hedera used OpenFlow both for identification and placing elephant flows, while Mahout Reference [5] on page 14 used end host detection with in-band signaling of elephant flows to the OpenFlow controller at the edge OpenFlow switch.

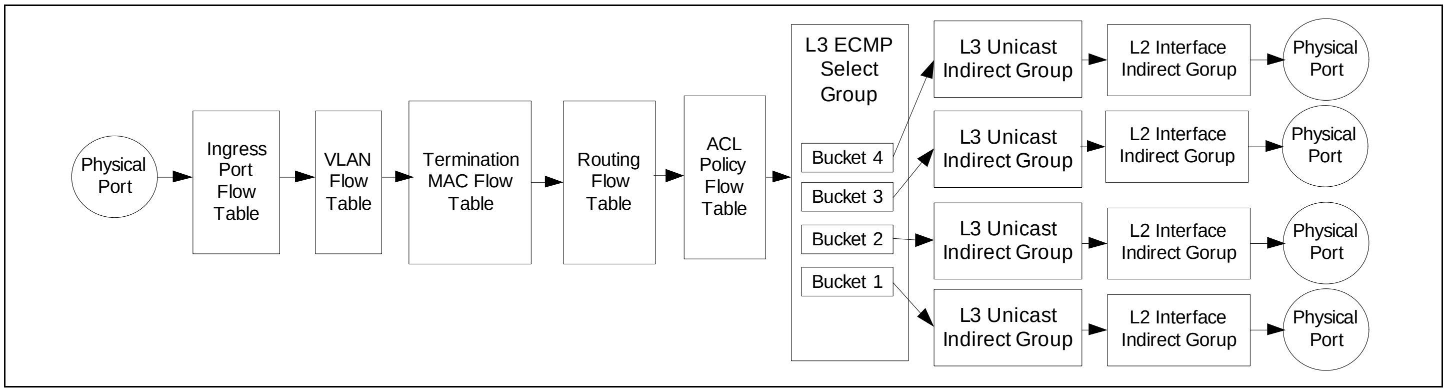
Using OpenFlow 1.3.1 to Build ECMP Networks

OpenFlow v1.0 uses only a single flow table with packet editing and forwarding actions in an action list. With OpenFlow v1.0 wildcard, flow entries can be used to forward packets as macro flows, with little or no controller intervention. But for OpenFlow v1.0 to support multipath load balancing, the controller needs to reactively place each flow independently as a separate micro flow. As a result, large numbers of flow entries are required, as well as significantly more transactions with the controller. Since OpenFlow v1.0 implementations on commodity switches can only utilize a TCAM for load balancing, the total number of flows required can easily exceed what the relatively low cost switches provide.

OpenFlow v1.3.1 by contrast uses multiple flow tables, but more significantly, provides the group table that can be used to leverage the multipath hashing capabilities in switch hardware. The OpenFlow v1.3.1 “select” group type is designed to map directly to hardware ECMP group tables. As a result, most forwarding can be handled using macro flows with significantly fewer entries in the flow table. In addition, multiple tables can be used to minimize the “cross-product explosion” effect of using a single table. With OpenFlow Data Plane Abstraction (OF-DPA), an OpenFlow v1.3.1 Controller could set up the routes and ECMP group forwarding for all of the infrastructure switches. In particular, it can be used to populate the routing tables and select groups for multipath. In this scenario, the controller uses LLDP to map the topology and detect link failures. The controller also polls switches for port statistics and keeps track of the overall health of the network.

The OF-DPA flow tables and group tables that would be used for this application are shown in Figure 3. These are programmed from an OpenFlow Controller using standard OpenFlow 1.3.1 protocol messages. The VLAN Flow table is used to allow particular VLANs on a port or to assign a VLAN to untagged packets. The Termination MAC Flow table is used to recognize the router MAC. Forwarding rules are installed in the Routing Flow table after the output groups are defined. The L2 Interface group entries define the tagging policy for egress member ports, the L3 Unicast group entries provide the packet editing actions for the next hop header, and the L3 ECMP group entry selects an output member based on hardware hashing.

FIGURE 3: OF-DPA Programming Pipeline for ECMP



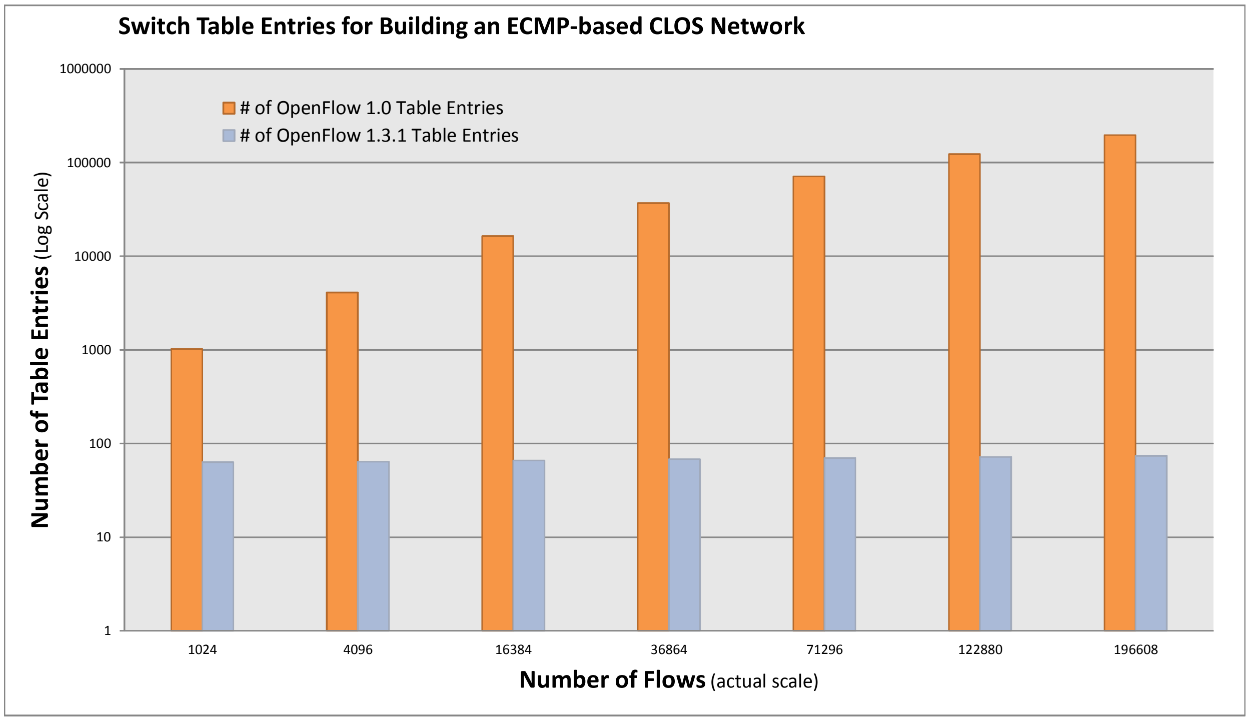
One of the advantages of using OF-DPA is that it can install customized ECMP select groups. For example, rather than being limited to the fixed N-way ECMP groupings that would be installed by a distributed routing protocol, the OpenFlow Controller can use OF-DPA to install finer grained groups with buckets for different sets of ECMP members, e.g., it can assign groups to flows. It can also customize the select groups to realized different weights for different members.

To illustrate the scalability of an OpenFlow v1.3.1 Switch in terms of table entries required, the following example is considered. Shown is Leaf or Top-of-Rack switch with 48 downlink (to servers or hosts) and six uplink ports (to spine switches) in a ECMP-based CLOS network. The Leaf or Top-of-Rack switch processes anywhere from 1K to 192K flows[1]. Table 1 and Figure 4 show the number of switch table entries required when using a single (TCAM) table OpenFlow v1.0 implementation versus a multi-table OpenFlow v1.3.1 implementation using OF-DPA.

TABLE 1: Leaf Switch Table Entries for an ECMP-based CLOS Network

# of Hosts, Ingress Ports	# of IP DEST	# of L4 SRC Ports	# of Connect-ions (flows)	# of Egress (Uplink) Ports	VLAN Flow Table Entries Needed	Term MAC Table Entries Needed	ECMP Group Entries Needed	L3 Unicast Group Entries Needed	L2 Interface Group Entries Needed	L3 Routing Table Flow Entries Needed	# of OpenFlow 1.0 Table Entries (TCAM)	# of OpenFlow 1.3.1 Table Entries
48	1	1024	1024	6	48	1	1	6	6	1	1024	63
48	2	2048	4096	6	48	1	1	6	6	2	4096	64
48	4	4096	16384	6	48	1	1	6	6	4	16384	66
48	6	6144	36864	6	48	1	1	6	6	6	36864	68
48	8	8912	71296	6	48	1	1	6	6	8	71296	70
48	10	12288	122880	6	48	1	1	6	6	10	122880	72
48	12	16384	196608	6	48	1	1	6	6	12	196608	74

FIGURE 4: Leaf Switch Table Entries for Building an ECMP-based CLOS Network



1. These are unidirectional flows.

As is evident from Table 1 and Figure 4 above, when using the OpenFlow v1.0 single table implementation, the switch table size requirement significantly increases as the number of flows increase, whereas with the OpenFlow v1.3.1-based implementation with OF-DPA, the switch's table resources are utilized with much higher efficiently. For example, in a scenario with 48 hosts (servers), with 16K Layer 4 source ports and 12 IP destination ports (resulting in about 196K flows), the number of table entries required with OpenFlow v1.3.1 with OF-DPA is only 74 using economical SRAM-based tables, compared to 196K entries required with OpenFlow v1.0 using expensive TCAM-based tables. This is a remarkable improvement in the utilization of switch hardware resources, enabling much higher scale and cost-effectiveness.

Using OpenFlow 1.3.1 to Traffic Engineer Elephant Flows

The example discussed in the previous section showed how a large ECMP-based CLOS network can be built to process a large number of flows, while conserving switch table entry resources. However, applying ECMP-based load balancing for both elephant and mice flows is not effective as described earlier. OpenFlow v1.3.1 with OF-DPA also makes it straightforward to traffic engineer the elephant flows. As a first order solution, select group multipath forwarding can be used for the mice flows. Only the elephant flows, which typically comprise fewer than 10% of the flows, need to be individually placed by the controller. Furthermore, with application orchestrators identifying elephant flows a priori to the controller, elephant flow placement can be done in advance of traffic appearing on the network.

Figure 5 on page 10, these redirection flows would be placed using entries in the Routing Flow Table if the criterion is the destination host, or in the ACL Policy Flow Table for application- or source-based selection. They could forward directly to an L3 Unicast group entry or to an “engineered” L3 ECMP group entry.

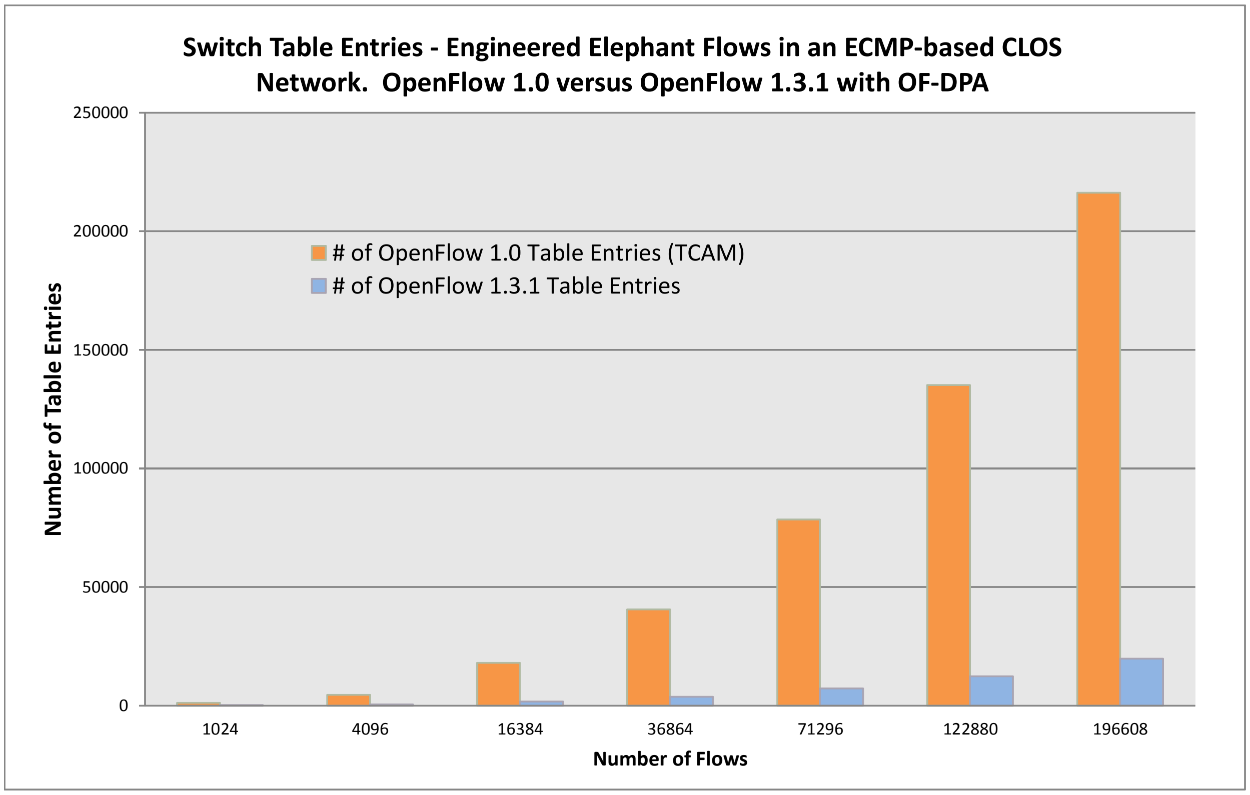
The example below is one where the elephant flows are redirected using application- or source-based selection criterion, therefore requiring entries in the ACL Policy Flow Table.

TABLE 2: Leaf Switch Table Entries for Engineered Elephant Flows in an ECMP-based CLOS Network

# of Hosts, Ingress Ports	# of IP DEST	# of L4 SRC Ports	# of Connect -ions (flows)	# of Elephant Flows (10%)	# of Egress (Uplink) Ports	VLAN Flow Table Entries Needed	Term MAC Table Entries Needed	ECMP Group Entries Needed	L3 Unicast Group Entries Needed	L2 Interface Group Entries Needed	L3 Routing Table Flow Entries Needed	# of ACL Flow Table Entries (TCAM)	# of Open Flow 1.0 Table Entries (TCAM)	# of Open Flow 1.3.1 Table Entries
48	1	1024	1024	102	6	48	1	1	6	6	1	102	1126	165
48	2	2048	4096	410	6	48	1	1	6	6	2	410	4506	474
48	4	4096	16384	1638	6	48	1	1	6	6	4	1638	18022	1704
48	6	6144	36864	3686	6	48	1	1	6	6	6	3686	40550	3754
48	8	8912	71296	7130	6	48	1	1	6	6	8	7130	78426	7200
48	10	12288	122880	12288	6	48	1	1	6	6	10	12288	135168	12360
48	12	16384	196608	19661	6	48	1	1	6	6	12	19661	216269	19735

An OpenFlow v1.3.1 switch implemented with OF-DPA scales for this Traffic Engineered scenario. All the mice flows can use the inexpensive SRAM-based hash tables, while the elephant flows will need entries in the TCAM ACL Flow table. Table 2 and the graph shown in Figure 5 show the switch table entry requirements for OpenFlow v1.0 and OpenFlow v1.3.1 with OF-DPA for implementing traffic engineering of elephant flows in an ECMP-based CLOS network.

FIGURE 5: Leaf Switch Table Entries for Engineered Elephant Flows in an ECMP-based CLOS Network



As is evident from Table 2 and Figure 5, when using the OpenFlow v1.0 single table implementation, the switch table size requirement exacerbates further when the TCAM entries needed to engineer the elephant flows are added. On the other hand, in the case of OpenFlow v1.3.1-based implementation with OF-DPA, the switch's table resources continue to be utilized with much higher efficiency.

For example, in a scenario with 48 hosts (servers), with 6K Layer 4 source ports and six IP destination ports (resulting in about 37K flows and where about 3.7K entries are elephant flows), the number of table entries required with OpenFlow v1.3.1 with OF-DPA is about 3.7K using both TCAM- and SRAM-based tables, compared to 41K entries required with OpenFlow v1.0 using only TCAM-based tables.

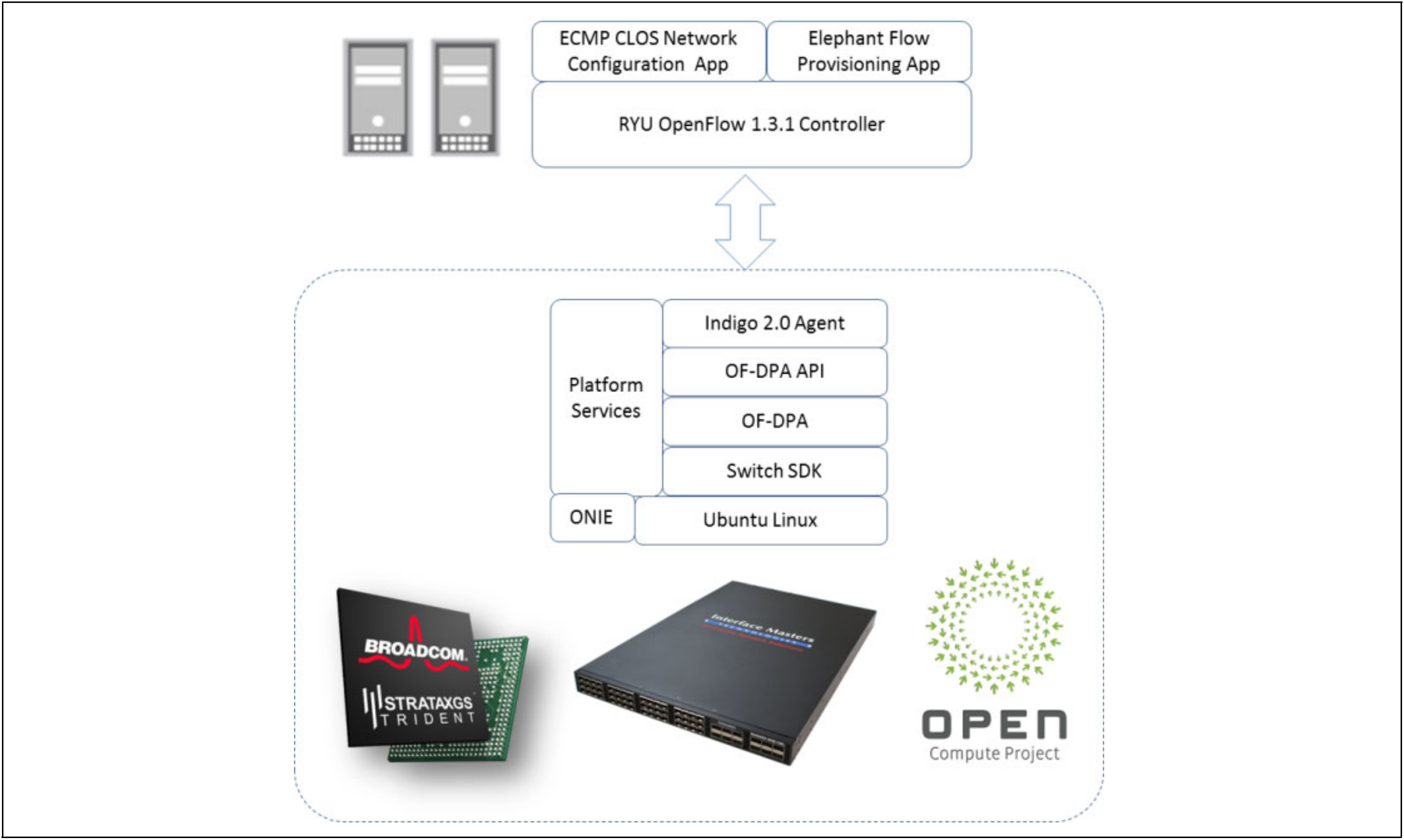
In the case where the elephant flows are redirected using the destination host-based selection criterion, there is no requirement for the extra costly ACL Policy Flow table entries; the inexpensive Routing Flow table can be used instead.

In both cases, not only is the performance of applications improved, but also there is an order of magnitude improvement in the utilization of switch hardware resources, enabling much higher scale and cost-effectiveness.

Open Hardware and Software Implementation

The Layer 3 ECMP CLOS network and engineered elephant flows application is implemented using open hardware and software solutions. The switch hardware system is based on the Broadcom StrataXGS Trident II switch-based OCP (Open Compute Project) open switch specification (draft), for details see http://www.opencompute.org/projects/networking/.

FIGURE 6: Open Software and Hardware Implementation



The switch system includes an open source Ubuntu Linux distribution, open source open network install environment (ONIE) – based installer, switch SDK, OF-DPA software, and open OF-DPA API. An open source reference agent (based on Indigo 2.0, see http://www.projectfloodlight.org/indigo/) is integrated with the OF-DPA API. The northbound interface of the agent is integrated with the open source RYU OpenFlow 1.3.1 Controller (see http://osrg.github.io/ryu/). The ECMP CLOS provisioning and engineered elephant flows applications are built into the RYU OpenFlow 1.3.1 Controller. Figure 6 shows the hardware and software implementation.

Summary and Future Work

OpenFlow v1.3.1 with OF-DPA enables implementation of Layer 3 ECMP-based CLOS networks that can scale to a large number of nodes, while delivering high cross-sectional bandwidth for east-west traffic. Compared to OpenFlow v1.0, OpenFlow v1.3.1 with OF-DPA delivers an order of magnitude improvement in how switch table resources are utilized, enabling much higher scale and performance. ECMP-based load balancing features available in switch hardware can be utilized for the larger number of latency sensitive and short lived mice flows in data center networks. Layer 3 ECMP-based CLOS networks are not sensitive to workloads and specific performance requirements. Elephant flows, unless specifically traffic-engineered, can cause significant degradation of application performance. OpenFlow v1.3.1 with OF-DPA enables traffic engineering of elephant flows in large scale CLOS networks, enabling higher application performance, while conserving switch table resources to deliver greater scale.

MiceTrap (Reference [13] on page 14), considers traffic engineering the mice flows. It makes the case that random ECMP combined with preferential placement of elephant flows could degrade the latency sensitive mice for applications with short flows that temporarily are assigned to paths with elephants. The solution is to adaptively use weighted routing for spreading traffic. Such capability can be enabled by using OF-DPA to engineer custom OpenFlow select groups with different sets of bucket members. Bucket weights can be supported in OF-DPA by defining more than one Layer 3 ECMP group entry bucket referencing the same layer Unicast group entry, so that output ports where elephant flows are placed can be assigned lower weights for randomly placing mice flows.

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use and are also defined in the table below.

Term	Description
ASIC	Application Specific Integrated Circuits. Represented networking switch silicon in this white paper.
CLOS	CLOS-based networks provide full cross sectional bandwidth to nodes connected to the network. Clos network is a kind of multistage switching network, first formalized by Charles Clos in 1952. Also known as Fat Tree topology.
ECMP	Equal Cost Multi-Path, used for multi-path routing.
Flow	Sequence of packets with the same selection of header field values. Flows are unidirectional.
Flow Table	OpenFlow flow table as defined in the OpenFlow v1.3.1 specification.
Flow Entry	Entry in an OpenFlow flow table with its match fields and instructions.
Group Table	OpenFlow group table as defined in the OpenFlow v1.3.1 specification.
Group Table Entry	Entry in an OpenFlow group table, with its Group Identifier, Group Type, Counters and Action Buckets fields.
Leaf Switch	A top of rack (TOR) switch in a CLOS network where the network is built using spine and leaf switches.
OpenFlow protocol	A communications protocol between control and data plane of supported network devices, as defined by Open Networking Foundation (ONF).
Software-Defined Networking	User programmable control plane. In this document, Software-Defined Networking (SDN) refers to the user ability to customize packet processing and forwarding.
Spine Switch	An aggregation switch connected to top of rack switches in a in a CLOS network where the network is built using spine and leaf switches.
SRAM	Static random-access memory, a type of semiconductor memory used for Layer 2, Layer 3 and hash tables lookup in switch ASICs.
TCAM	Ternary Content Addressable Memory. Used for wide wild card match rules entry and processing in switch ASICs.
Traffic Engineering	Classification of traffic into distinct types and application of quality of service or load balancing policies to those traffic types.

References

Document (or Item) Name
[1] Kandula, S.; Sengupta, S.; Greenberg, A. G.; Patel, P. & Chaiken, R. (2009), The nature of data center traffic: measurements & analysis., in Anja Feldmann & Laurent Mathy, ed., 'Internet Measurement Conference', ACM, pp. 202-208.
[2] Thaler, D., & Hopps, C. (2000), Multipath Issues in Unicast and Multicast Next-Hop Selection, IETF RFC 2991.
[3] Hopps, C. (2000), 'Analysis of an Equal-Cost Multi-Path Algorithm', IETF RFC 2992.
[4] Al-Fares, M.; Radhakrishnan, S.; Raghavan, B.; Huang, N. & Vahdat, A. (2010), Hedera: Dynamic Flow Scheduling for Data Center Networks., in 'NSDI', USENIX Association, pp. 281-296.
[5] Curtis, A. R.; Kim, W. & Yalagandula, P. (2011), Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection., in 'INFOCOM', IEEE, pp. 1629-1637.
[6] Benson, T.; Anand, A.; Akella, A. & Zhang, M. (2011), MicroTE: fine grained traffic engineering for data centers., in Kenjiro Cho & Mark Crovella, ed., 'CoNEXT', ACM, pp. 8.
[7] Mogul, J. C.; Tourrilhes, J.; Yalagandula, P.; Sharma, P.; Curtis, A. R. & Banerjee, S. (2010), DevoFlow: cost-effective flow management for high performance enterprise networks., in Geoffrey G. Xie; Robert Beverly; Robert Morris & Bruce Davie, ed., 'HotNets', ACM, pp. 1.
[8] Farrington, N.; Porter, G.; Radhakrishnan, S.; Bazzaz, H. H.; Subramanya, V.; Fainman, Y.; Papen, G. & Vahdat, A. (2010), Helios: a hybrid electrical/optical switch architecture for modular data centers., in Shivkumar Kalyanaraman; Venkata N. Padmanabhan; K. K. Ramakrishnan; Rajeev Shorey & Geoffrey M. Voelker, ed., 'SIGCOMM', ACM, pp. 339-350.
[9] Seedorf, J., Berger, E. (2009), 'Application Layer Traffic Optimization (ALTO) Problem Statement,' IETF RFC 5693.
[10] White, T. (2009), Hadoop 'The Definitive Guide: MapReduce for the Cloud', O'Reilly
[11] Jain, S., et. al (2013), 'B4: Experience with a Globally-Deployed Software Defined WAN,' SIGCOMM, ACM, http://cseweb.ucsd.edu/~vahdat/papers/b4-sigcomm13.pdf.
[12] Phaal, P.; Panchen, S. & McKee, N. (2001), 'InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks', IETF, RFC 3176.
[13] Trestian, R.; Muntean, G.-M. & Katrinis, K. (2013), MiceTrap: Scalable traffic engineering of datacenter mice flows using OpenFlow., in Filip De Turck; Yixin Diao; Choong Seon Hong; Deep Medhi & Ramin Sadre, ed., IEEE Symposium on Integrated Network Management (IM2013), IEEE, pp. 904-907.