CoSwitch: A Cooperative Switching Design for Software Defined Data Center Networking

Yue Zhang[1], **Kai Zheng**[1], **Chengchen Hu**[2], **Kai Chen**[3], **Yi Wang**[4], **Athanasios V. Vasilakos**[5]
[1]IBM China Research Lab [2]Xi'an Jiaotong University [3]Northwestern University
[4]Tsinghua University [5]National Technical University of Athens

***Abstract*—Software Defined Network (SDN) provides flexibility and agility for customizing the Data Center Network (DCN), which is essential element for the full support of multi-tenancy. In the current DCN infrastructure, physical switches are able to support SDN protocols, such as OpenFlow. However, due to the limited resources of embedded CPU and on-chip memory size, physical OpenFlow switch suffers from considerable overhead for handling frequent control plane workload. Worse still, great numbers of mice flows existing in DCN real-life trace generate huge control plane workload, causing the control plane of physical switch to be the serious performance bottleneck. In this paper, based on the performance profiling of virtual and physical switches, we present CoSwitch, a novel switching design for DCN. CoSwitch leverages virtual switch on the server, which has powerful control plane, to cooperate with physical switch, which has high-speed data plane. A close-loop workload balancing mechanism is designed to distribute different kinds of workload into different types of switches in the most efficient way. Through real-life trace experiments on physical testbed, CoSwitch is demonstrated to achieve both scalable control plane and data plane. Without specialized hardware requirement, CoSwitch mitigates the bottlenecks of OpenFlow.**

Keywords-DCN, SDN/OpenFlow, Mice Flow, Control Plane Workload, Virtual Switch, Physical Switch

I. INTRODUCTION

With the rapid progress of cloud computing applications, today, there appear more and more mega multi-tenant data centers, which contain tens of thousands of interconnected servers. The Amazon Elastic Compute Cloud (EC2) and IBM's Blue Cloud are successful examples of public cloud service offerings. In the cases of the Infrastructure as a Service (IaaS) clouds, the virtual machines (VMs) are usually not long lived. VMs are created and deleted dynamically, since the tenants may have very different requirements on the virtual network topology according to the hosting applications.

To satisfy more and more agile requirements emerged in modern DCN, *Software-Defined Networking* (SDN) is a promising approach, which provides on-demand flow based control for data centers. A well-known implementation of SDN is OpenFlow [1], which runs global software based control and management functions independent with the underlying network infrastructure. The decoupling of control plane and data plane allows the control logic [2] to be implemented on a different platform with the forwarding devices, so as to release the low-power control plane CPUs on the switches. The emerging complex network control requirements, such as path computation, QoS, network security etc., can be hosted on the high-efficient central network controller rather than on the switch [3] [4]. In this way, the switches can be made of simply "dumb" forwarding mechanism on data plane, and with a very thin control plane to coordinate with the centralized controller.

Although OpenFlow is born with the philosophy to simplify the switch hardware and form an efficient data plane, through a study of typical data center traffic and the existing OpenFlow compatible switches, we found obvious performance killer right on the thin control plan of the switches. Briefly speaking, data center traffic usually contains huge number of mice flows[1]. The mice flows bring control plane intensive workload for OpenFlow handling, because for every new coming flow, the switch will ask for a flow rule to update into the flow table through control plane. Since the embedded CPU resource is so limited in today's physical switch, the flow table updating speed tends to be slower than the mice flows' coming speed. Even worse, the limited on-chip memory will further aggravate the updating frequency of flow table. For this point, OpenFlow introduces even more control plane overhead on the switch side, which is contrary to our expectation. According to such an observation, we argue that the existing OpenFlow-based SDN solutions need to be re-considered and re-designed.

In this paper, we present CoSwitch, a cooperative OpenFlow switching framework, to efficiently implement the conceptual OpenFlow-based SDN. At the heart of framework is a novel flow classification mechanism which strips the mice flows from the majority traffic volume, i.e. the elephant flows[2]. Virtual switches deployed on the server side are utilized to handle the mice-flows, in a distributed manner. In this way, the control plane intensive workload, such as frequent flow table updates and communications with the controller, is handled by virtual switch with more powerful control plane capability. The slow control plane of physical switch no longer contaminates the efficiency of data plane operations, and therefore mitigates the performance issue denoted before. The contributions of this paper are as follows.

1) We observe interesting workload pattern in operation data center network, where presents two different types of workload intensive flows. The significantly large population of mice flows generates heavy control plane intensive workload for OpenFlow. 2) Based on the performance profiling of real-life OpenFlow switches, we figure out three potential bottlenecks of the network devices. 3) A novel workload balancing approach is proposed as CoSwitch, to make cooperation of virtual switch and physical switch. The benefits of CoSwitch are verified through test-bed experiments.

[1] Mice flow: short in duration, huge in amount.
[2] Elephant flow: last in a long time, containing many packets.

II. PROBLEM STATEMENT

Before the detailed description of the proposed design, we first articulate the problems in building a software defined data center network. In this section, we demonstrate three challenging issues through experiments on commercial OpenFlow switches and real data center traces, which motivate us to propose CoSwitch.

A. Data center workload analysis

We have analyzed several real-life traffic traces collected from an IBM production data center locating in Europe, consisting of more than 5k VMs for hosting services. The traces are dumped into separate files every five minutes by NetFlow running in the DCN at 2008-01. As in common practice, the traffic flows is identified by the 5-tuples (source/destination IP, source/destination port, and protocol type) of the packet. Usually, the network devices process the traffic at a per packet basis, e.g., routing, ARP, etc., and therefore the per-flow workload is related to the flow size (number of packets contained). The distribution of the flow size is investigated here on the real DCN traces mentioned above.

As indicated in Figure 1, the Cumulative Distribution Function (CDF) of flow number and packet number is demonstrated together, with the increasing of flow size. We can find that the number of flows changes in a different way with the number of packets. The rate of the flow number curve is obviously bigger than the rate of packet number curve, when the flow size (per flow packet) is smaller than 18. And the opposite case turns out when the flow size is bigger than 36. In other words, there are more mice flows containing fewer packets, and fewer elephant flows containing more packets. Figure 2 depicts the comparisons between mice flows and elephant flows with regard to their population and total traffic volume (indicated by total packet number). It can be seen that the mice flows take over 80% of flows population, and contain only less than 15% number of the total traffic volume. While the small numbers (10%) of elephant flows contain more than 75% of the total traffic volume. The polarized traffic patterns will bring unbalanced workload to the DCN.

In the OpenFlow network, workload patterns of these two types of flows are quite different. The processing of mice flows requires frequent flow table operations on the control plane, such as flow add/delete/lookup actions. The traffic volume in the mice flows is relatively small, so the cost for packet processing in data plane is actually not very significant. Thus we call the mice flows as “control plane intensive workload”. On the other hand, in the case of elephant flows, since the flow population is relatively much smaller, the flow operations on control plane are therefore much fewer. However, there are huge amount of packet data in the elephant flows, leading to high data plane operation overhead. So we call the elephant flows as “data plane intensive workload”.

B. Performance analysis of network devices

Two types of OpenFlow enabled switches are studied here. One is dedicated physical switch, the other is software-based virtual switch. Figure 3 depicts the hierarchical networking infrastructure of data center. The virtual switches deployed on server connect to VMs directly. Access layer switches connect the virtual switches, located at the top of the rack, also known as ToR switches. On top of ToR, aggregation or core switches are deployed. In some of the fat-tree based DCN designs [5] [6], the core switches are composed with the same type of ToR.

Variety of OpenFlow supported switches can be found in the markets, such as IBM RackSwitch G8264 for the core, NEC PF5240, and Pronto 3290 for the ToR. When the packet comes into the switch ports, it will be processed and redirected to the destination port based on the flow table rules. Otherwise, if there is no entry match in the flow table, the packet will be directed to the controller through control plane. After getting the responding flow rule from the controller, the switch stores it into the flow table. From this point of view, OpenFlow actually introduces two kinds of potential performance bottlenecks to the switch. One is the embedded control CPU, because it is usually too weak to afford the frequent flow table update under heavy control plane workload. The other is the limitation of flow table size. For the sake of cost reduction, the entry number of on-chip memory, e.g. TCAM, is not large (usually 2k or 4k) on the ToR level switches, which will meet serious performance degradation (e.g. frequent cache miss) when handling simultaneous active flows more than that number.

Compared to the physical switches, generic servers provide relatively much more powerful control CPU and unlimited storage for the virtual switches to use. Further more, since the virtual switches are deployed distributedly, the workload can also be handled in the distributed way. As the most well known one type of virtual switches, OpenVSwitch (OVS) [7] implements software based standard Ethernet switching with OpenFlow supported, resided within the hypervisor. However, the software implementation is a double-edged sword. Though the resource can be used is relatively adequate, it is too generic to be as efficient as dedicated hardware. The software forwarding mechanism of OVS is orders of magnitudes lower than that of the physical devices. Therefore, the forwarding capability (mainly CPU) of virtual switch can be a potential bottleneck when the data plane workload is heavy.

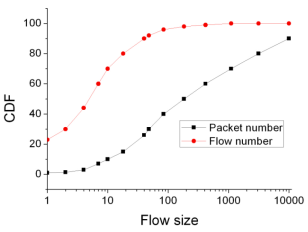

Figure 1. CDF of flow number and packet number

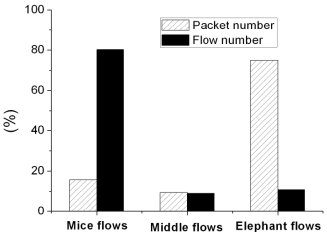

Figure 2. Comparison of mice flows and elephant flows

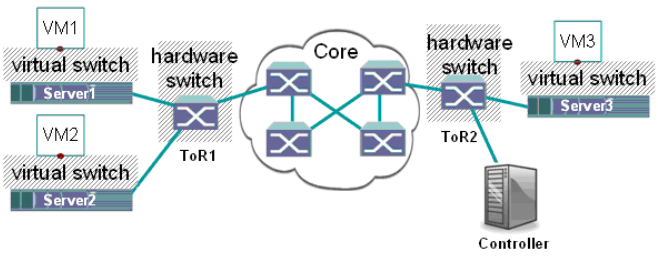

Figure 3. Physical infrastructure of DCN

C. *OpenFlow networking bottleneck*

Based on the analysis above, the three potential bottlenecks of DCN are measured in this part for both types of switches, including data plane CPU cost on virtual switch, control plane CPU cost and flow table entry cost on physical switch. Here we base our experiments on the operational OpenFlow test-bed as depicted in Figure 3. As the earliest-announced physical OpenFlow supported switch, Pronto 3290 is used as ToR and OVS is used as the virtual switch. Server1 and Server2 are both connected by the same physical switch ToR1. The testing traffic are replayed based on the real trace, and is sent from VM1 on Server1 to VM2 on Server2.

We start from the data plane first. To measure the data plane processing power, static OpenFlow rules are pre-installed on the corresponding switches. Only a few wildcard rules are used to handle all the traffic without involving the control plane. The result in Figure 4 shows that the physical switch can achieve line rate (1Gbps), and the embedded control plane CPU on the switch stays idle. As for the virtual switch, which does packet processing through software, the corresponding CPU core usage is also shown in Figure 4. By zooming in the input load, we find that the server CPU utilization is proportional with the traffic amount, and finally reaches the processing bottleneck under the extremely heavy load, about 80k pps. This constrains the general-purpose CPU resource of the server. The extra CPU cost will always waste valuable computing resource for the tenants.

To measure the control plane performance, we do not set pre-installed rule in the following experiments. For each new incoming flow, the switch has to get a flow rule from the controller first. The rule is also need to be removed when it is timeout. Switch resource cost under the control plane intensive workload is compared in Figure 5. Since there is no pre-installed rule in this test, the switch needs to spend more CPU time on the control plane to update flow table for each new coming flow, before handling it on data plane. It can be seen that the control plane workload dose not cause performance bottleneck on virtual switch's CPU (Xeon E5420) and memory (RAM), with the input load growing over 10Kpps[3]. However, for the physical switch, the CPU (MPC8541E E500) is too weak to afford frequent flow table updating. The embedded CPU reaches saturation when the input load is over 7.7Kpps. The on-chip memory (TCAM) is also so limited (e.g. 2K) that will be used up when the input load is over 6Kpps, even under very short flow entry timeout (3s).

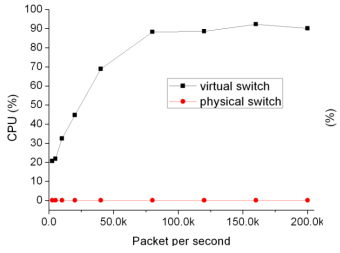

Figure 4. CPU cost under data plane intensive workload

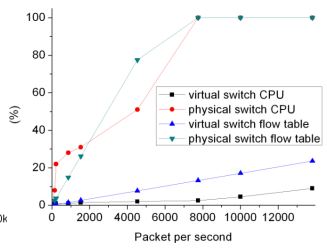

Figure 5. CPU and flow table cost under control plane intensive workload

[3] The real-life workload we used is about 10Kpps and 40Mbps.

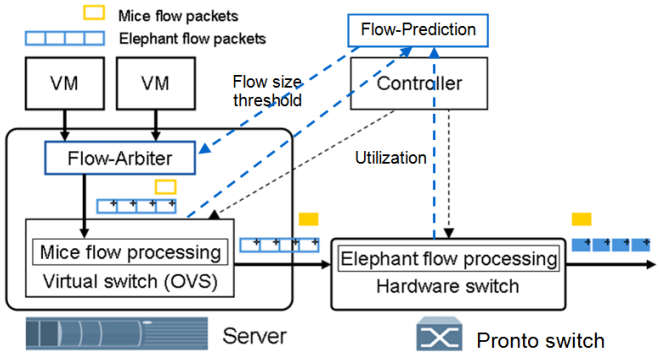

Figure 6. CoSwitch architecture

III. CoSwitch Architecture

As observed in section II, the physical switch is good at handling large amount of traffic with small flow numbers, i.e. the elephant flows. However, the embedded CPU is so weak as to be the performance bottleneck when handling large number of mice flows. Moreover, the limited flow table size leads to short timeout of the flow rules, therefore incurs even more table updating workload to the CPU. On the other hand, we note that the virtual switch with more control plane capacity is better to handle the mice flows. Based on the above observations, the cooperative switching scheme, CoSwitch, is proposed to strip the control plane intensive processing workload of mice flows from the physical switches, and leverage the complementary virtual switches on the server side to handle these large-number-small-volume mice flows, in a distributed and adaptive manner. In this way, both the physical switches and the virtual switches exert their strengths, respectively, and complement each others' weakness in performance, leading to an adaptive balance of the system.

Overall, CoSwitch improves current DCN designs to optimize SDN/OpenFlow capability. As depict in Figure 6, OpenFlow on the data center is provided by the 1) physical switch with OpenFlow supporting, i.e. Pronto 3290, and 2) virtual switch deployed in the host server, i.e. OVS. A close-loop workload distribution mechanism is introduced in CoSwitch, supported by two modules. Flow-Arbiter deployed on the server side, and Flow-Prediction deployed as an OpenFlow controller module. It is worth nothing that both of the modules are software implementation, which means no particular hardware modification is needed.

Flow-Arbiter is deployed between VMs and hypervisor virtual switch. The traffic of VMs is filtered by Flow-Arbiter, where the elephant flows are picked out of mice flows. Some developed methods [8] [9] can be used to identify the elephant flows with insignificant overhead. Through the experiment results in section II.A, elephant flows contain much more packets than mice flows. So a feasible way is to pick out the elephant flows with the flow size bigger than a threshold. With the division of mice flows and elephant flows, the workload can be distributed to the virtual switch and physical switch, respectively.

In most cases the processing power of virtual switch and physical switch do not stay steady, either because of the

inherent physical difference among network devices, or the dynamic traffic workload. So the flow size threshold should be dynamically updated at run time. Flow-Prediction is deployed on the OpenFlow controller, i.e. Floodlight in the test-bed, to compute out the flow prediction threshold. The threshold is not obtained from the flow features. Instead, it is based on the resource utilization periodically feedback from the virtual and physical switches. If either of the switches is heavy-loaded and the other is not, the Flow-Prediction module will make proper change of the threshold to shift some workload to the light-loaded switch. In this way, the value of threshold is dynamically adjusted to adaptively balance the workload between the switches.

To be more specific, in CoSwitch, the two types of switches are cooperated as follows: Flow-Arbiter divides the traffic into mice and elephant flows. The large numbers of mice flows are processed locally, by the OVS within each server. The elephant flows will only be marked and forwarded to the physical switch. The marking methods are flexible, either the VLAN tag or TOS tag can be used. At physical switch side, the mark of each packet is checked. For the marked elephant flows, the processing is carried by the switch hardware. Since the huge numbers of mice flows are not marked so as not to be processed on the physical switch, only a small number of static wildcard rules are used to forward the mice flows together, without per flow rules updating cost. In this way, the virtual switch and physical switch are cooperated to leverage their complementary capabilities to handle the data center workload.

IV. Evaluation

In this section, CoSwitch is implemented on the real testbed, where the performance of three SDN/OpenFlow approaches is studied. Experiment results show the comparison of OpenFlow capability provided by virtual switch, physical switch, and the cooperation of both (CoSwitch), respectively. Compared with the other two approaches, CoSwitch shows distinct superiorities on eliminating all the potential bottlenecks of OpenFlow deployment.

A. Testbed setup and CoSwitch implementation

The test-bed infrastructure of prototyping CoSwitch implementation is depicted in Figure 3. Commodity OpenFlow enabled switches (Pronto 3290) are used for ToR, as the physical switches with OpenFlow support. The switches are interconnected with each other via 10GE link, and with 16 IBM system x3550 servers via 1GE link. On each server, two components need to be deployed. 1) OpenVSwitch is deployed to replace the switch of KVM hypervisor, as the virtual switch supporting OpenFlow. 2) Flow-Arbiter is deployed as a net-filter module to intercept the VM traffic before they come into OVS. Besides, the OpenFlow controller (Floodlight) with Flow-Prediction module is deployed in a standalone server, with connectivity of all virtual and physical switches. When the traffic from VM1 comes out to the host Server1, the packets are hooked by the Flow-Arbiter in host kernel space. The Flow-Arbiter divides the flows into mice flows and elephant flows based on the flow size threshold computed by the prediction module on the controller. The mice flows are processed in OVS before sending out of Server1. The other elephant flows are marked with a special TOS tag and forwarded out to be processed on physical switch. The mice flows are forwarded without any processing on physical switch.

B. Data plane performance

As discussed in Section II, the potential bottleneck on data plane is the CPU resource of virtual switch, which can be estimated by the CPU utilization of host server. Figure 7 shows the particular CPU core utilization of virtual switch. Through CPU time sampling, it can be seen that most of the CPU time is cost by the packet processing of elephant flows, which is significantly larger than the processing of mice flows. This is because the packet processing on virtual switch takes up the general CPU, the elephant flows with more packets and bytes will cost more CPU resource. Another part of overhead is the unavoidable network I/O for packet forwarding. With the knowledge of the CPU cost distribution, it is reasonable of CoSwitch to put the handling of elephant flows to the physical switch, where the packet processing is performed by specialized hardware with little data plane overhead.

In Figure 8, the CPU overhead on the server is compared under the approaches of CoSwitch and handling all flows on virtual switch. As tested in Section II, the real trace is not heavy loaded, about 10Kpps and 40Mbps. Here we zoom in the trace to generate heavy workload. For CoSwitch, since most packets of elephant flows are not processed on the server side, the corresponding CPU core cost is much lower, only required by mice flow packet processing and network I/O. It can be seen that when the input load is over 100MB/s, the virtual switch will meet the CPU bottleneck for processing all the packets. However the core usage in CoSwitch can be reduced to 37%, by merely handling the mice flows. This not only mitigates the server CPU bottleneck for packet processing, but also saves CPU resource. The saving can be significantly valuable in data center environment containing tens of thousands of CPU cores. It is worthy noting that the processing overhead of the server side intelligence is only related to the traffic intensity generated by each physical server, and keeps relatively stable with the increasing scale of DCN.

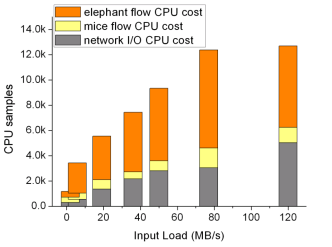

Figure 7. Server CPU cost distribution

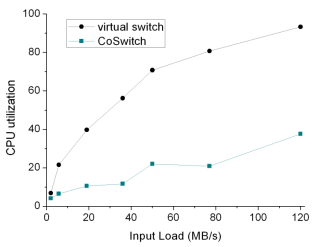

Figure 8. Server CPU utilization

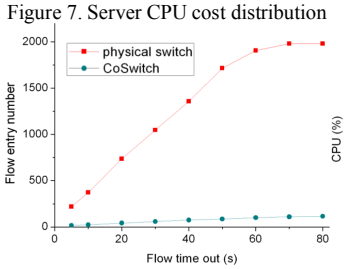

Figure 9. Flow entries cost on different timeout

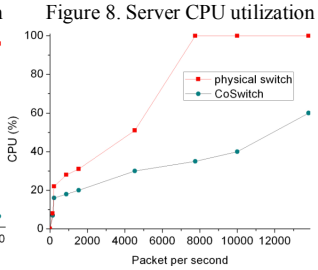

Figure 10. Switch CPU utilization

C. Control plane performance

The two control plane performance issues of OpenFlow are all on the physical switch. One is the storage cost of flow rules, which comes from the limitation of on-chip flow table size. The run-time flow entry requirement is depicts in Figure 9. With the growing of flow timeout, the number of flow entries required is increasing fast to overflow the hardware capability (e.g., 2K on Pronto 3290). By contrast in CoSwitch, only not more than 200 elephant flow rules need to be cached on physical switch, for which is more than sufficient to handle. The other serious bottleneck on control plane is the capability of embedded switch CPU. In Figure 10, the switch CPU utilization is compared between CoSwitch and physical switch approaches. It can be seen that the physical switch CPU will soon get overloaded when the input exceeds 7.7Kpps, because huge amount of mice flows require lots of flow table updating. In CoSwitch, since the flow updating on physical switch only triggered by the small numbers of elephant flows, the increasing rate of switch CPU cost is much lower, means that the design is more scalable and cost-efficient.

In summary, based on the real-life DCN trace analysis, the traffic flows are observed to have different characteristics for mice flows and elephant flows. Real test-bed experiments have demonstrated that the combination of virtual switch and physical in CoSwitch can elastically handle mice flows in the virtual switch, and elephant flows in the physical switch with lower resource cost. As a result, the potential DCN scalability bottlenecks of server CPU, switch CPU and switch on-chip storage are effectively mitigated.

V. Related Work

Data center networking has been well studied. Some recent researches provide some level of agility based on fat-tree like topology. PortLand [5] presents a big layer-2 network that takes advantages of layer-2 features. However, it needs special switches to support new forwarding behaviors, limiting its compatibility and deployment practicability. VL2 [6] only modifies server side software to provide agile IP level virtualization, which could be easily deployed today. However, it does not have the potential to support valuable layer-2 features such as “plug-and-play”. Compared to these fat-tree based works, several topology independent solutions have been proposed recently. NetLord [10] focus on providing network virtualization through hypervisor modification at server side, with centralized management module. However, the specialized server side intelligence can not be as agile as provided by SDN to meet the DCN requirements, such as flexible traffic control.

Fat-tree based solutions use low-end physical switches to provide agile and cost-effective non-blocking connectivity. However, the physical switch meets limitations on the on-chip storage size and embedded CPU resource when agile control, such as OpenFlow is required in DCN. General CPU and RAM are also integrated into physical switch to support efficient flow-based forwarding in [11] [12]. Yet at the same time, the building cost of DCN is considerable, because of the specialized switching hardware.

On the other hand, the DCN control plane of OpenFlow is revisited to meet the needs of agility. DIFANE [13] distributes control logic from global controller into hierarchical data plane switches. However, agility is lost by converting software logic into fixed kernel space implementation, which runs counter to the SDN concepts. DEVOFLOW [14] relieves the pressure of centralized controller by merely concentrating fully control on elephant flows. As a result, the agile control of mice flows is sacrificed.

VI. Conclusion

The enablement of OpenFlow on DCN encounters several bottlenecks, of which the most serious one is the handling of mice flows on physical switches, which has limited resource on the control plane. Through a cooperative switching design of virtual switch and physical switch, CoSwitch is presented to mitigate the bottlenecks of OpenFlow deployment, without specialized hardware requirement. The powerful control plane of virtual switch provides scalable processing capability of the mice flows. And the high speed data plane of physical switch preserves the processing performance of elephant flows. Real-life test-bed experiments prove that CoSwitch is a scalable SDN design to provide agile traffic control for DCN.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner. OpenFlow: enabling innovation in campus networks. In Newsletter, ACM SIGCOMM Computer Communication Review, Volume 38 Issue 2, April 2008.

[2] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown and S. Shenker. NOX: towards an operating system for networks. In Newsletter, ACM SIGCOMM Computer Communication Review, Volume 38 Issue 3, July 2008.

[3] A. Tootoonchian, Y. Ganjali. HyperFlow: A Distributed Control Plane for OpenFlow Networks. In Proc. of INM/WREN, 2010.

[4] http://floodlight.openflowhub.org/

[5] R. N. 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 Proc. of SIGCOMM, 2009.

[6] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel and S. Sengupta. VL2: a scalable and flexible data center network. In Proc. of SIGCOMM, 2009.

[7] http://openvswitch.org/

[8] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, Hedera: Dynamic Flow Scheduling for Data Center Networks. In Proc. of NSDI, 2010.

[9] A. R. Curtis, W. Kim and P. Yalagandula, Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In Proc. of INFOCOM, 2011.

[10] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes and Y. Pouffary. NetLord: A Scalable Multi-Tenant Network Architecture for Virtualized Datacenters. In Proc. of SIGCOMM, 2011.

[11] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller and N. McKeown. Implementing an OpenFlow Switch on the NetFPGA platform. In Proc. of ANCS, 2008.

[12] G. Lu, C. Guo, Y. Li, Z. Zhou, T. Yuan, H. Wu, Y. Xiong, R. Gao, and Y. Zhang. ServerSwitch: A Programmable and High Performance Platform for Data Center Networks. In Proc. of NSDI, 2011.

[13] M. Yu, J. Rexford, M. J. Freedman and J. Wang. Scalable flow-based networking with DIFANE. In Proc. of SIGCOMM, 2010.

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