Energy Efficient SDN Commodity Switch based Practical Flow Forwarding Method

Amer AlGhadhban and Basem Shihada
Computer, Electrical, and Mathematical Sciences and Engineering (CEMSE) Division
KAUST, Saudi Arabia
{amer.alghadhban, basem.shihada}@kaust.edu.sa

***Abstract*—Recent SDN researches suffer from over-accumulation of unhealthy flow-load. Instead, we leverage the SDN controller network view to encode the end-to-end path information into the packet address. Our solution *EncPath* significantly reduces the flow-table size and the number of control messages. Consequently, the power consumption of network switches is in orders of magnitude less than other evaluated solutions. It also provides flow management flexibility and scalability. We compare *EncPath* with single and multipath routing solutions and single path solution. Also, we operated them in proactive and reactive modes. We find that *EncPath* flow entries in core switches in a multihomed fat-tree with 144 hosts is approximately 1000 times smaller than Equal-Cost MultiPath (ECMP) and random routing. Additionally, the number of control messages to setup the network is reduced by a factor of $200\times$. This, consequently, affords data-plane and control-plane devices space to process other tasks.**

***Keywords*—OpenFlow switch; Data center; Software defined network; Flow-table size**

I. Introduction

SDN-based solutions which use the OpenFlow protocol as the underlying paradigm suffer from several unexpected challenges, such as thousands of flow-entries, controller messages and unacceptable flow-setup delay [1]. These challenges hinder SDN from providing fine-grained flow control of the network in current network services and middleboxes management complexity, (e.g., frequent configuration and errors). The aforementioned complexities increase the flow-table entries in the orders of magnitude. This is evinced in [2] where the authors found a 10 to 1 ratio of flows to each host. This is in agreement with previous findings on OpenFlow switch implementation, where they found that its flow-table contains 78K flow-entries [1], which is close to legacy data center routing table numbers [2]. This quantity of flow-entries requires 15 seconds to collect its flow statistics [1] and hinders the migration of flow-entries to data-plane devices to reduce the flow-setup time. Unfortunately, the current hardware switches, which provide line rate packet forwarding, have a limited amount of expensive and power hungry Ternary Content Addressable Memory (TCAM). Thus, TCAM with an average size of 1.5 Mbits, is not capable of accommodating the large amount of flow entries of current data center network [1]. Moreover, the incremental engagement between fine-grained flow control and the number of flow-entries needs to be positively abstracted without losing granularity.

978-1-5090-0223-8/16/$31.00

To address these challenges researchers proposed several solutions [1] [3] [4] [5]. A source routing techniques by using path information were proposed in [3] [4] [5] to relax the complexity of core network by building a label switching network [3], embedding path information into a new header [4], or into a packet address or MPLS-label [5]. In our solution, defined as *EncPath*, we exploit the ability of a controller to get complete information of a network path before installing the flow. The work herein is designed to reduce the energy consumptions of data-plane devices by reducing flow-entries in flow table. This achieved by encoding the flow path information into the packet IP or MAC addresses while the address rewriting flow-entries are offloaded to be handled by hosts themselves. Also, the solution challenges flow-ID and path length are addressed. *EncPath* combines a reactive and a proactive flow-entries' installation. Being reactive at the edge devices aims at providing enough visibility to the controller to perform necessary functionalities, such as multipath routing. Being proactive aims at reducing the load on the controller and on data-plane devices. Furthermore, the controller needs only to communicate with the edge devices in response to network incidents or for security precautions.

EncPath introduces several advantages. It has a very small flow-table size. The number of flow-entries in every switch in our solution is linearly proportional to the number of switch ports. When the number of flow entries is reduced, TCAM deployments are positively influenced in two domains: the required size of TCAM to accommodate these flow-entries becomes small and the energy consumption of TCAM or any other deployed memory is reduced. Due to most flow-entries are proactively installed and only edge devices' flow-entries are reactively added, the number of control messages is reduced, in consequence. Finally, it has the flexibility to work in the network and data-link layers, and it also has the flexibility to work with multipath and single path routing algorithms.

II. The Overhead of Flow-Entries

In this subsection we build a simple experiment to measure the number of flow-entries needed by proactive-based solutions, such as [1], to avoid the overhead of control-plane devices in flow setup, and enable network switches to use the proactive flow-entries to forward incoming traffic. However, the performance figures of reactive flow requests are evaluated in previous studies, such as [1] [6].

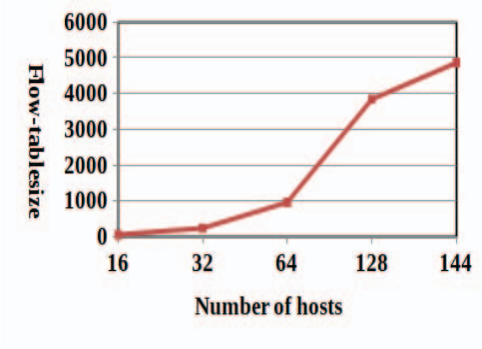


Fig. 1: ECMP flow table size according to different number of hosts.

We build our investigation on an OpenFlow executed on Openvswitch 1.10 in mininet which is installed on top of Intel Xeon CPU X5550 2.67GHz 16 cores with 48GB memory and the latest version of POX controller. The POX controller and mininet are both installed in the same machine. In this experiment we build a fat-tree topology by using Ripcord [7] and Riplpox [8]. Since, ECMP is usually used by proactive based solutions [1], we adopted it in this experiment. We measured the number of flow-entries needed by ECMP to enable a single edge switch to directly handles the network flows without involving the controller. We start the experiment with 16 hosts and stop at 144 hosts. The results are shown in Fig. 1. We can see how the number of ECMP flow-entries are increased exponentially with the increase in the number of hosts.

On the other hand, we measured the effect of different sizes of flow-tables on the flow status request delay. We start the experiment with 1000 until 100,000 flow-entries. The results of this experiments are shown in Fig. 2. The time to pull the flow is increased dramatically with the increase in the number of flow-entries. The experiment demonstrates how proactive based solutions need thousands of flow-entries to run the operated network and how these flow-entries cause an unacceptable delay to collect their flow statistics. Although, most of the proactive based solutions deploy flow sampling solutions to collect the flow statistics, in real data center where the number of hosts are in order of magnitude compared to this experiment, the sampling process will be complex and time consuming [9].

III. EncPath Design

A. OpenFlow-based EncPath

In OpenFlow-based *EncPath*, the controller uses OpenFlow Discovery Protocol (OFDP), which is similar to Link-Layer Discovery Protocol (LLDP), to build a complete view of the network topology; how network switches are connected with each other and how to reach end-hosts in the network. In this work, the controller communicates with the edge devices where it is responsible in rewriting the address of every passes packet with *EncPath* address. After we investigated the OpenFlow features and specifications, we found OpenFlow (v1.1 and later versions) support arbitrary netmask in IP and MAC addresses where the ones and zeros can be inserted arbitrarily in any octets of the netmask. In this work we utilize this feature and IP_{TTL} value, as a hop counter, together to point to the right IP/MAC octet that containing the outgoing port number. In Fig. 3 an example of the basic *EncPath* process wherein Host A aims to send a message to Host B. The traffic should go through port 11 in switch 1 (SW1), port 2 in switch 2 (SW2), port 4 in switch 3 (SW3), and final the out-port 6 in switch 4 (SW4). In *EncPath*, the controller encodes the outgoing port numbers into an IP or MAC address and sends two rewrite flow entries, one to SW1 to rewrites the address of outgoing packets into *EncPath* and another to SW4 to returns it back into its original address. SW1 rewrites the source IP address of the packet into srcIP = 11.2.4.6 and in this example destination IP will not be changed. SW1 forwards the packet via port 11, switch 2 via port 2 and so forth. The last switch in the path will rewrite the source IP addresses, (i.e., *EncPath* address), of the packet back to its original value. In *EncPath*, the controller installs in each in-path switch proactive flow-entries consist of the outgoing port number in the IP octet which represents the switch location in the path. SW4 forwards any incoming packet via the outgoing port that has the same value as the value in the fourth octet of the incoming packet's IP address. The TTL value is used to indicate the switch position along the path, (e.g., when the TTL=255 means the switch need to read the first octet in the IP address). Although, in previous example we used IP address octets as a container to *EncPath* information, we also use MAC address space for *EncPath* address which provides a flexibility to our solution and gains even better performance.

B. EncPath Power Model

Since, the ICT consumes about 10% of global annual power and the global community struggling to reduce the carbon-dioxide emissions footprint [10]. In addition, the data center network has about thousands of switches. When their power consumption is reduced by small fraction the total sum of their power reduction will be significant. Since, TCAM has a limited

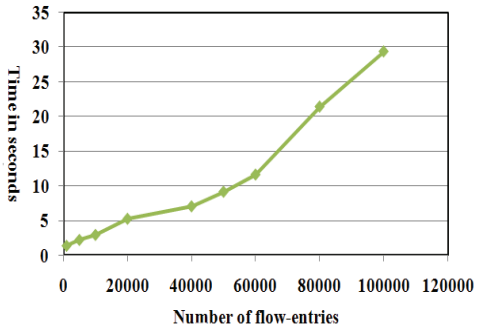


Fig. 2: The response time of flow status requests according to different flow table sizes.

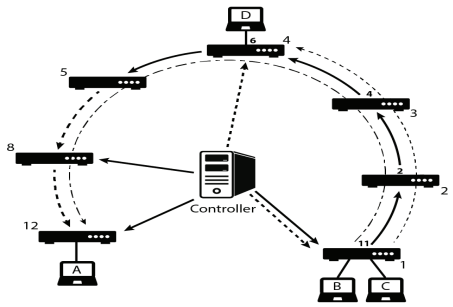


Fig. 3: Host B communicates with Host A, in 4 hops linear topology, and Host C communicates with Host D in 12 hops linear topology.

capacity, where it can not accommodate the expected flow-entries of data center network devices. We measure the energy efficiency of our solution and others based on the energy consumption of DRAM. To measure the switch power consumption of our solutions and others, (i.e., random, spanning-tree and ECMP routings), we use the model explained in the work of T. Vogelsang [11]. The power consumption of a DRAM is the sum over all charging and discharging operations of a capacitance C to voltage V multiplied with their number of occurrence, (i.e., the clock speed),f, where K is the number of flow entries that are examined to find a match. Since, this is a random number, we take its expected value $\overline{K}$ in power consumption evaluations. The values of C,V and f are available in DRAM companies data-sheet.

$$P_{DRAM} = \sum_{i=1}^{K} \omega \Longleftrightarrow \omega = \frac{1}{2} C V^2 f \tag{1}$$

Then:

$$P_{DRAM} = \omega \overline{K} \tag{2}$$

C. Offload Flow-Setup

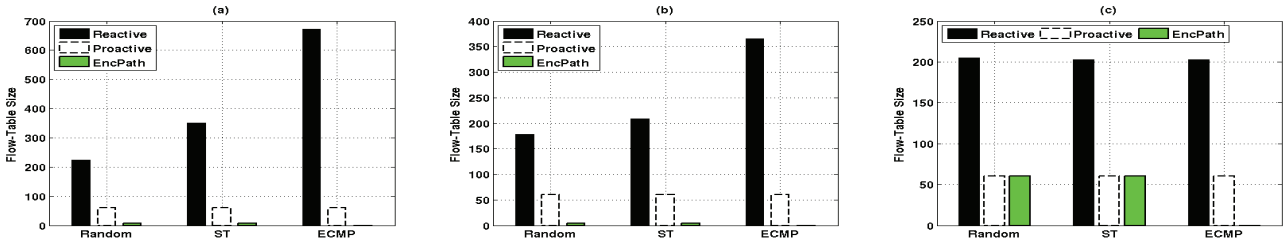
In our solution the edge switches are responsible for communicating with the controller and handle flow setup messages as well as rewriting flow packets with *EncPath* address. However, one of our goals is to relax all physical switches in the network, (i.e., edge, aggregation and core switches), and makes them ready to exploit the simplicity of *EncPath* solution. We test *EncPath* in the a multihomed fat-tree network topology. We find that the number of flow entries is proportional to the number of the flows initiated from end hosts, as shown in Fig. 4(c). The switch power consumption increased with the increase in the flow-table entries. We aim at making its flow-entries systematic and proportional to the number of ports. On the other hand, servers in a data center run with high performance processors and high capacity memories as well as has a fast scale-up procedures not like network switches. Servers hardware prices, such as CPU and memory, are cheap compared to network devices. In order to reduce the load on edge switches and utilize these available resources we build a direct OpenFlow southbound communication between the servers and the controller. In this configuration the flow setup load on the edge switches are offloaded and spread among servers in the same subnet. When a server needs only to handle its traffic flows, the number of flow entries in its table are proportional to its traffic-load. Hence, the overloaded servers in the subnet will not overload the edge switch with flow setup requests and in consequence affect the other servers in the same subnet. The load of this communication on the CPU and memory are evaluated and the results are shown in Fig. 10.

D. Path Length Challenge

The average path length is 5 hops in data center network which is accommodated in the number of octets of source/destination IP and MAC addresses [12]. Nevertheless, *EncPath* provides scalable mechanism to avoid the long-path challenge. When the traffic is planned to be forwarded through a path has a number of hops shorter than the number of octets of used address, (i.e., IP or MAC address that used to implement *EncPath*), in this case the *EncPath* algorithm simply pads zeros into empty octets. In another case, when the path has number of hops more than 8 hops. Any type of address, (e.g., MAC and IPv6), will face similar limitations, however, we focus on IPv4 and MAC address in this work. The controller, when it receives a new flow request, reads the host information and topology databases to discover the outgoing path, when the path is longer than 8/12 hops, at the same time the controller inserts the rewriting entries in the edge devices, it also, inserts a rewriting entry in the 8th/12th switch to again rewrites the packet address of that flow with the subsequent path information. The second challenge is how the 8th/12th switch recognizes the packets that belongs to the longer-path flow? In *EncPath*, we use the combination of source and destination MAC addresses, (i.e., in case of IP*EncPath*), TTL value and *EncPath* information as a Flow-ID which is used by the 8th/12th switch to recognize the targeted flow. For more clarification, in Fig. 3, when Host A sends traffic to Host C, the path between them consists of a number of hops greater than 8-hops. The controller instructs the 8th switch in the path to rewrite a source/destination IP address again with a new *EncPath* that containing the remaining 4-hops outgoing ports number. The 8th switch recognizes the designated flow's packets by the tuples of its MAC and *EncPath* addresses.

IV. Validation and Results

In order to validate our solution, we use riplpox [8] based on Ripcord [7] and mininet [13] emulator. The propose solution is validated in two different scenarios to prove its robustness; three-layer homogeneous fat-tree, k=4 and a linear topology with 12 switches and 4 hosts as shown in Fig. 3. Also, we built a testbed containing of 7 virtual machines to represent a single pod of the fat-tree topology and use it as an example of real implementation of our solution. In the testbed each VM has an openvswitch [14] installed in it and the controller is installed in a physical machine.



(a) Flow-table size in core switch (b) Flow-table size in aggre. switch (c) Flow-table size in edge switch

Fig. 4: Flow-table size of *EncPath* and other solutions in the multihomed fat-tree topology with 16 hosts.

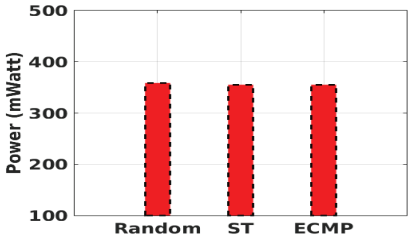


Fig. 5: The power consumption of edge switches in reactive mode.

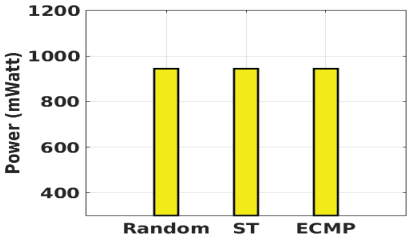


Fig. 6: The power consumption of edge switches in proactive mode.

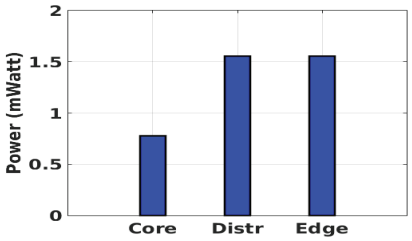


Fig. 7: The power consumption of EncPath.

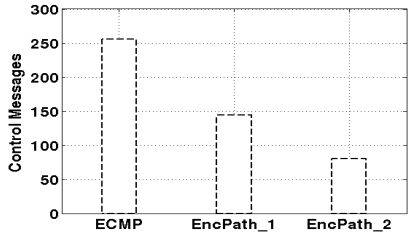


Fig. 8: The network setup load.

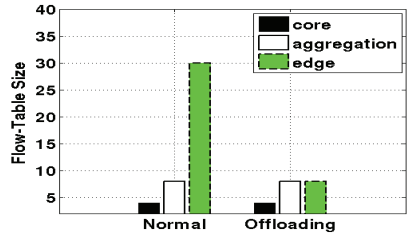


Fig. 9: The flow-table of *EncPath*.

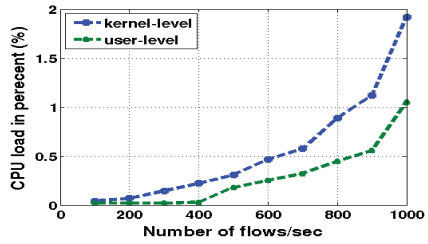


Fig. 10: The CPU load of Nmap flows.

A. *Three-Layer Homogeneous Fat-Tree Results*

In Fat-Tree topology, we compare our solution with three different routing algorithms; ECMP, Spanning Tree (ST), and Random Routing (RN) algorithms. Furthermore, everything runs in a proactive and reactive modes.

1) EncPath Flow-Entry: We evaluate *EncPath* with other solutions by using a fat-tree with 16 hosts and the results are posted in Fig. 4. Respectively, Fig. 4(a) displays the flow-table size of one of the core switches, where the flow-table size of *EncPath* core switch containing only 4 entries and it is $15\times$ smaller than others in proactive mode and in reactive mode it outperform others by a factor of $45\times$ to $92\times$. Fig. 4(b) displays the size of flow-table in aggregation switches and Fig. 4(c), shows the size of flow-table in edge switches. Additionally, we evaluate *EncPath* and the other solutions by using 144 hosts in the same topology and 33% of the hosts initiate flows between each other. We find the number of flow entries of other solutions in proactive mode reached 4,860 entry in each switch, in contrast, the number of flow entries in *EncPath* switches is as what they are when tested with 16 hosts except the edge switches which is 576 flow-entries. Although, the edge switches have flow-entries smaller than other solutions, we need to reduce it further and makes it proportional to the number of ports. This is the purpose of *edge switches offloading* technique.

2) EncPath Energy Efficiency: Since, the ECMP routing and other solutions in proactive mode insert large number of flow-entries to setup the network, their average power consumption in proactive mode is 2.7 times of reactive mode, as illustrated in Fig. 5 and Fig. 6. Due to EncPath's flow-table isn't affected by a change in the number of hosts, its flow-table as well as power consumption stay constant, unless, there is a topology change. The power consumption of EncPath switches are shown in Fig. 7, which is in orders of magnitudes less than other solutions.

3) EncPath Setup: EncPath needs to install proactive flow-entries, as shown in Table 1, in network switches in order to inform the in-path switches how to handle exchanged packets. In this subsection we show how many control messages are needed by *EncPath* to install these proactive messages and compare it with other solutions in proactive mode. Since the solutions in reactive mode do not need to prepare the network before installing any flow, we compare *EncPath* with other solutions in proactive mode. The solutions in *proactive mode* need 256 control messages to prepare the network and this number for only 16 hosts as shown in Fig. 8, and 20,736 control messages when we test them with 144 hosts. In contrast, *EncPath* requires 80 control messages in first configuration, when the OpenFlow communication is between edge switches and the controller, and 144 control messages in second configuration, when the OpenFlow communication are performed between the servers and the controller. Consider that *EncPath* needs 2 reactive messages to setup the end-to-end flow which is not needed for network setup.

B. Edge Switches Flow Offloading Results

In this experiment we evaluate the possibility of creating an OpenFlow communication between a virtual server and a controller, also the load of this communication on the CPU and memory is measured. We aim to utilize the available resources and the scale-up facility of the servers to make the flow-entry in edge switches proportional to the number of ports. To represent one pod of the fat-tree we connected 7 virtual machines which are 5 switches and 2 servers, where a controller is installed in a separated machine. The source server has a 4 cores CPU and 4GB memory. We use Nmap software in the source server to initiate several flows from 100 to 1000 flows per second, where the TCP ports being varied in order to have new flow-entry for each Nmap flow. The flow table size of the edge switches and others are shown in Fig. 9, normal case means when the OpenFlow communication is between edge switches and the controller, while offloading case means when the communication is between the servers and the controller. On the other hand, the load of OpenFlow communication between the server and the controller on the CPU and memory are measured during Nmap flows. We find that in worst case the load on the CPU is less than 2% and the results are shown in Fig. 10, where the load on the memory oscillated between 0 to 0.3%.

V. Conclusion

In this work, we introduced *EncPath*, that reduces the flow-entries in OpenFlow switches to a number close to its number of ports. The flow path port numbers are interpreted as an IP or MAC address. In this case, the network switches need only to read the right address octet to understand where to forward the packet. When flow entries decreased, the number of control messages as well as the switch energy consumption are decreased. *EncPath* solution provides flexibility and scalability in several areas of deployment; network layers, routing mode and changing the path direction. The performance figures of *EncPath* is evaluated in detailed set of experiments with different configurations and scenarios including: linear topology and multihomed fat-tree, k=4, topology as well as a testbed of 7 VMs. We find that *EncPath* outperforms other tested solutions in the number of flow-entries, control messages and power savings.

References

[1] A. R. Curtis et al, "Devoflow: Scaling flow management for high-performance networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11, 2011, pp. 254–265.

[2] S. Kandula et al., "The nature of data center traffic: Measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '09, 2009, pp. 202–208.

[3] K. Agarwal et al., "Shadow macs: Scalable label-switching for commodity ethernet," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14, 2014, pp. 157–162.

[4] M. Soliman et al., "Source routed forwarding with software defined control, considerations and implications," in *Proceedings of the 2012 ACM Conference on CoNEXT Student Workshop*, ser. CoNEXT Student '12, 2012, pp. 43–44.

[5] J. Sangeetha Abdu et al., "Towards a flexible data center fabric with source routing," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, ser. SOSR '15, 2015, pp. 10:1–10:8.

[6] C. Rotsos et al, "Oflops: An open framework for openflow switch evaluation," in *Proceedings of the 13th International Conference on Passive and Active Measurement*, ser. PAM'12, 2012, pp. 85–95.

[7] M. Casado et al, "Ripcord: A modular platform for data center networking," EECS Department, University of California, Berkeley, Tech. Rep., Jun 2010.

[8] Riplpox. https://github.com/MurphyMc/riplpox.

[9] C. Barakat et al., "Ranking flows from sampled traffic," in *Proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology*, ser. CoNEXT '05, 2005, pp. 188–199.

[10] Matt, "Ict at 10% of global electricity consumption?" Sept. 2013. [Online]. Available: http://www.vertatique.com/ict-10-global-energy-consumption.

[11] T. Vogelsang, "Understanding the energy consumption of dynamic random access memories," in *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '43, 2010, pp. 363–374.

[12] J. Chabarek et al, "Networks of Tiny Switches (NoTS): In Search of Network Power Efficiency and Proportionality." in *Proc. of the 5th Workshop on Energy-Efficient Design*, 2013.

[13] N. Handigol et al, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '12, 2012, pp. 253–264.

[14] B. Pfaff et al, "Extending networking into the virtualization layer," in *In: 8th ACM Workshop on Hot Topics inNetworks (HotNets-VIII).New YorkCity,NY(October 2009*, 2009.