Proceedings of the Asia-Pacific Advanced Network ***2011*** *v.32, p. 14-25.*

Wider Adaptation and Enhancement of OpenFlow

Othman Othman M.M.1, Koji Okamura 2

1 Department of Advanced Information Technology, Graduate school of Information Science and Electrical Engineering, Kyushu University. 6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan. +81-92-802-2722

2 Research Institute for Information Technology, Kyushu University. 6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan. +81-92-642-4030

E-Mails: omo_5@ale.csce.kyushu-u.ac.jp, oka@ec.kyushu-u.ac.jp

http://dx.doi.org/10.7125/APAN.32.3   ISSN 2227-3026

Abstract: This paper shows our design of three enhancements to the current OpenFlow technology. OpenFlow is a promising future internet enabling technology that has a great potential to improve the current Internet by providing new functionalities and a new control scheme and thus enabling new smarter applications to be created. Our study aims to provide OpenFlow with three new features; network equipment to equipment flow installation, low level header description, and a new type of inactive flows. Through which we aim to extend OpenFlow's usability, by making it more self-aware and traffic-aware, by relieving some of the load off the OpenFlow controller, by enabling it to have the ability to forward and manipulate user defined protocol headers, and by providing OpenFlow with a method to support flows with strict timing requirements. Those modifications are proposed as a step forward towards encouraging a wider adoption of OpenFlow as an easily programed flow-based network technology that holds a great potential as a future Internet technology that can support newer and smarter class of applications.

Keywords: OpenFlow; Flow-Based Networks; Network Control.

1. Introduction

The Internet; is one of the greatest means of communication over the current and past centuries. The Internet plays an important role in our lives as it delivers information through the World Wide Web, and helps people to communicate using E-mail. More over the Internet plays an important complementary role to the traditional communication methods like telephone by providing an alternative audio and video calling service. Not to mention the wide variety of contents that it provides. All of those capabilities of the Internet promoted its use as a promising ground for many trends of commerce like the e-trade, and many other types of businesses that depends on the Internet for making its profit. And for the previously mentioned reason and many other reasons the Internet has become an indispensable part of our daily lives. And due to the importance of the Internet, researchers have been studying ways to improve the Internet and to provide new services and capabilities to it.

One of the concepts studied by researchers is the flow concept, where a flow is a sequence of packets from a source computer to a destination, which may be another host, a multicast group, or a broadcast domain and could consist of all packets in a specific transport connection. Grouping sequence of packets into flows is very reasonable, since that different services (applications) on the Internet have different characteristics and require different levels of support by the network, and thus flows can be used to group communications according to the type of service they belong to and then apply some rules for each group. As an example flows are used in [9] to assure different levels of Quality of Service (QoS) for different types of applications depending on the application's needs, while in [10] flows are used as a mean to apply security policies.

On 2008, OpenFlow [1] was first introduced. OpenFlow is a part of Stanford University's clean slate project. OpenFlow provides a specially designed way to control flows on the network equipment by the OpenFlow controller (which is a dedicated server for managing flows) through using the OpenFlow Protocol. Also, it enables more freedom and flexibility in decision making by splitting the decision making or routing from packet forwarding. According to OpenFlow; decision making can be done and modified freely by the OpenFlow controller according to layer 2, 3 and VLAN headers while the forwarding or routing is still done by routers or switches, in addition to, their original functionality. Moreover, OpenFlow defines actions to be performed on flows that can be either collection of statistics or usage data, forwarding packets, dropping packets, or manipulating packet's headers. This freedom, flexibility, and the wide range of actions performed on packets enable it to play a crucial role in developing the future Internet along with its main target which is running researchers' experiments on production networks. Despite this great flexibility of OpenFlow, there have been many concerns about the scalability of OpenFlow due to the way that the OpenFlow controller controls the OpenFlow network equipment, which forces a tight coupling between the controller and the network equipment. This would mean that the controller can be one of the bottlenecks in the system.

This paper shows our design of three enhancements of OpenFlow; aiming to take OpenFlow one step towards encouraging a wider adoption of OpenFlow by giving OpenFlow more flexibility and more diverse control methods. Those enhancements will help to relieve some load off the controller, also will help to make OpenFlow self-aware and able to react when under heavy loads in a manner that takes into account the current status of traffic. The proposed enhancements do not impose radical changes to OpenFlow, we propose three enhancements that can be an extension to OpenFlow. Those three enhancements are: Network equipment to equipment flow installation, user defined low level header description, and a new type of flows that are the inactive-programmed flows.

The remainder of this paper is organized as follows. We first show the first enhancement that is the Network equipment to equipment flow installation in Section 2. We then show the second enhancement of the user defined low level header description in Section 3. While, in Section 4, we show the third enhancement; that is a new type of flows that are the inactive-programmed flows. We then explain about our evaluation to those enhancements in Section 5 and conclude in Section 6.

2. Network Equipment to Equipment Flow installation

According to the current OpenFlow design; flows can be programmed by the controller. This means that the controller is the only entity that is responsible for installing and maintaining flows on the network equipment. For simplification let's call this type of flow installation the "controller to equipment flow installation". The controller to equipment flow installation has many advantages like having tight control over all of the equipment by the controller.

However, the advantages of the controller to equipment flow installation come with some cost. First is the probability that the controller would be a source of bottle neck in the whole system. This can be inferred by the number of flows that can be installed by the NOX controller as shown in [5] that are 30K flows per second, while, maintaining a sub-10ms install time, and the flow arrival rate in [6] that is 100K flow per second. Second, by limiting the flow manipulation to "the controller to equipment" installation method OpenFlow can miss some opportunities that the "network equipment to network equipment" can provide.

And so, we propose a new method for installing flows, that is, the "network equipment to equipment flow installation" method. Through using this method, the controller does not have to program (install) flows to each one of network equipment one by one; instead it can ask the equipment to spread this flow to other equipment on behalf of the controller, this can be useful in cases where the controller needs to program non critical-start up time flows. And thus relieving some load off the controller. Also, the network equipment to equipment flow installation method can be used to make the OpenFlow network more self-aware by having the network equipment cooperate and carry loads for each other upon the need and traffic situation by having the overloaded equipment delegating some of its flows to another network equipment.

2.1. Methodology

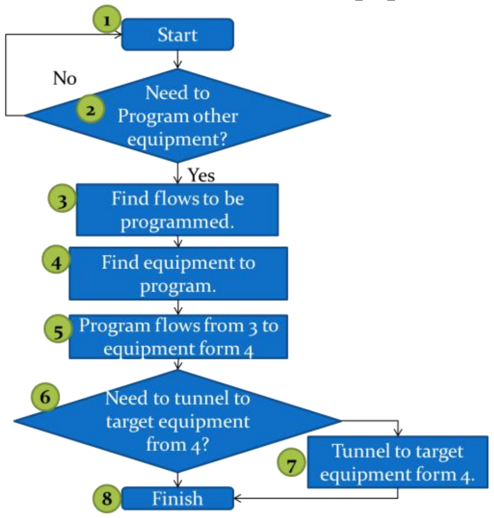
The main method used by the network equipment to network equipment flow installation shown in Fig.1 (a).

Were step 2 can be either; a request form the controller to spread a flow, or network equipment is overloaded and wants to delegate some of its flows to another network equipment (the method used for this case shown in Fig.1 (b)).

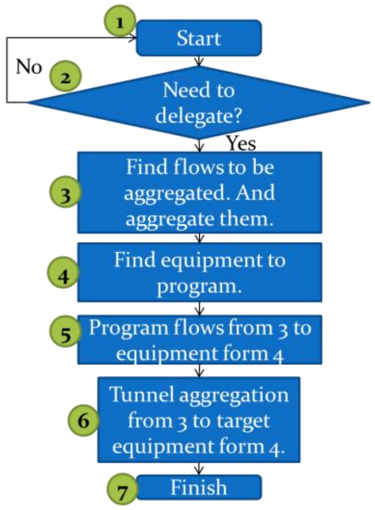
Step 3 will lead to find the flows that the controller requested to spread in case that the controller asked to spread those flows, or can lead to find an aggregation flow that aggregates other flows (it must aggregate one or more flows that are currently handling a specific percentage of the traffic). Where the network equipment that possesses those aggregated flows will delegated them to another network equipment, which means that the delegating network equipment will delegate a number of flows responsible for a specified percentage of the traffic and replaces those flows with one delegation flow in order to tunnel the traffic of the delegated flows to the delegated network equipment.

While step 4 can be easily carried out by identifying neighbor equipment, through using the Link State Database or any other method.

Step 5 will be done using the newly proposed type of packets in the OpenFlow Protocol, which we designed to enable the network equipment to equipment flow installation.


1
Start
No
2
Need to Program other equipment?
Yes
3
Find flows to be programmed.
4
Find equipment to program.
5
Program flows from 3 to equipment form 4
6
Need to tunnel to target equipment from 4?
7
Tunnel to target equipment form 4.
8
Finish

(a)


1
Start
No
2
Need to delegate?
Yes
3
Find flows to be aggregated. And aggregate them.
4
Find equipment to program.
5
Program flows from 3 to equipment form 4
6
Tunnel aggregation from 3 to target equipment form 4.
7
Finish

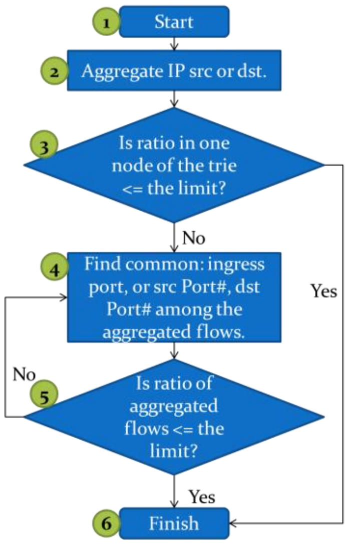
(b)

Figure 1. (a) Main skeleton for the network equipment to equipment flow installation method. **(b)** The case of overloaded equipment aggregation.

Figure 2 shows the aggregation algorithm which is designed to be used by an overloaded network equipment in order to aggregate many flows (that are responsible for a specified percentage of the traffic) into one aggregation flow, and then delegating the aggregated flows to another network equipment. Through this process, the aggregation flow is used to tunnel packets for the aggregated flows to the delegated network equipment. This can be useful to reduce the traffic in an overloaded network equipment; by delegating some flows that are responsible for some percentage of the traffic to another network equipment and replacing those flows by one flow that aggregates them and then using this flow to tunnel packets to the delegated network equipment.

Figure 2 (b) shows in details the IP aggregation algorithm, which appears as step 2 in Fig.2 (a). IP aggregation is done by building a trie that has nodes containing counters. This is done by looping through IP addresses in the flow table and adding one bit from each of them in each iteration, starting from the most significant bit. In case of not finding a node in the trie that has that value, a new node carrying the value will be created, and its counter will be updated by adding the percentage of the traffic related to the IP address in the processed flow. While in case of finding a node having the same value, then only its counter will be updated. After completing each iteration of the loop; all of the nodes in the trie must be checked if their counter has a close number to a predefined value that represents percentage of traffic to be delegated to another network equipment. If a trie node has a counter with the desired value, then the loop stops and this node will be marked. This marked trie node represents two key values that are; the length of the aggregation flow's IP address, that will be used to generate the mask for the aggregation flow, and the IP address, that will be used to as the aggregation IP, which is obtained by traversing the trie form the root down to the marked node.

The purpose of steps 3 through 5 in Fig.2 (a) is to refine the aggregation flow in case that the IP aggregation was not sufficient for the aggregation. This is done by finding common characteristics as the ingress port, Dst port number, or Src port number among flows aggregated by the IP aggregation algorithm.


1
Start
2
Aggregate IP src or dst.
3
Is ratio in one node of the trie <= the limit?
No
4
Find common: ingress port, or src Port#, dst Port# among the aggregated flows.
Yes
No
5
Is ratio of aggregated flows <= the limit?
Yes
6
Finish
(a)

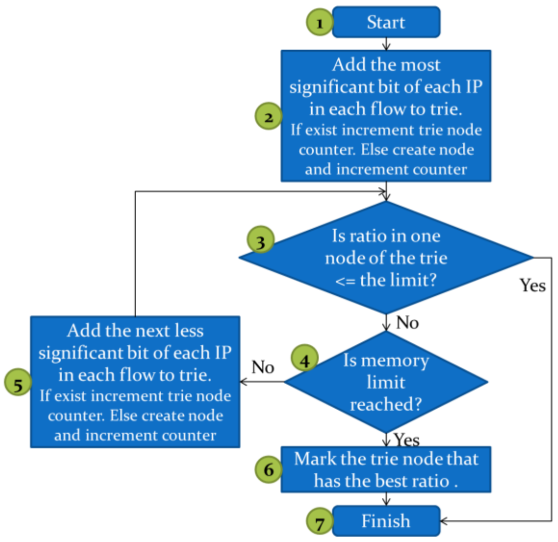

1
Start
2
Add the most significant bit of each IP in each flow to trie. If exist increment trie node counter. Else create node and increment counter
3
Is ratio in one node of the trie <= the limit?
Yes
No
5
Add the next less significant bit of each IP in each flow to trie. If exist increment trie node counter. Else create node and increment counter
No
4
Is memory limit reached?
Yes
6
Mark the trie node that has the best ratio .
7
Finish
(b)

Figure 2. (a) Step 3 in Figure 1 (b): finding flows to be aggregated and aggregate them(Aggregation Algorithm). **(b)** Step 2 in Figure 2 (a): aggregating IP src or dst.

2.2. Protocol

In order to enable the network equipment to equipment flow installation to be adopted by the OpenFlow Protocol, three new packets have to be introduced. Those new packets are; equipment to equipment (e-e) flow installation request, e-e flow installation reply, and an e-e flow installation confirmation.

The first packet is the e-e flow installation request. This packet holds an OpenFlow header, list of flows to be programmed, address of the equipment that sent the request and an identification value, address and identification of the originator of the request (who requested for the flows to be programmed in the first place, i.e. controller or an equipment), Level of Flow Installation, the Time To Live (TTL) of the IP protocol, and a temporary identifier for this request. Where, the Level of Flow Installation (LFI) is somewhat similar to the TTL field in the IP protocol. However, the TTL in IP protocol indicates how many hops a packet can travel, where the TTL value will be decremented when passing in a network equipment and the packet will be discarded when the value reaches 0. While, in the case of the LFI, it is decremented each time an equipment accepts the installation. LFI and TTL are used together to control the propagation of the e-e flow installation request.

While the second packet is the e-e flow installation reply. It contains an OpenFlow protocol header, the identification of the e-e flow installation request, the reply to the request which can be either an acceptance or a rejection, address and self-identification of the equipment that sent the reply, the value of the LFI which will be the same in case of rejection or decremented in case of acceptance.

And the third packet is the e-e flow installation confirmation. Its purpose is clear; to confirm to the equipment that sent the reply that its reply has been received.

2.3. Usage Scenario

To shed light on the network equipment to equipment flow installation, we will show two usage scenarios for it.

The first usage scenario is the case of using the network equipment to equipment flow installation in order to spread (program) a flow to the whole network without having the controller to program each of the network equipment individually as in the original OpenFlow's flow installation method (see Fig.3 (a)). Instead, using our method the flow will be programmed by the controller to some network equipment, and they will be asked to spread the flow to other equipment and so on until it reaches all network equipment (see Fig.3 (b)). This would be useful in case a flow with a noncritical start time has to be programmed into the whole network.

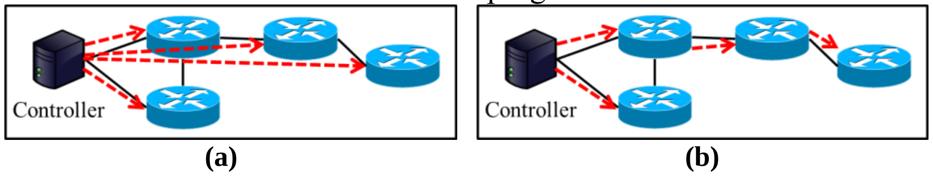


Figure 3. (a) Regular flow installation: from controller to each network equipment. **(b)** The flow installation using the proposed network equipment to equipment flow installation.

The second usage scenario (shown in Fig.4) is the case of using the network equipment to equipment flow installation to help the network to develop a self-aware behavior by having the overloaded equipment to delegate some of its flows to another equipment and tunnel traffic for those flows to the delegated equipment without having the controller to interfere with this process.

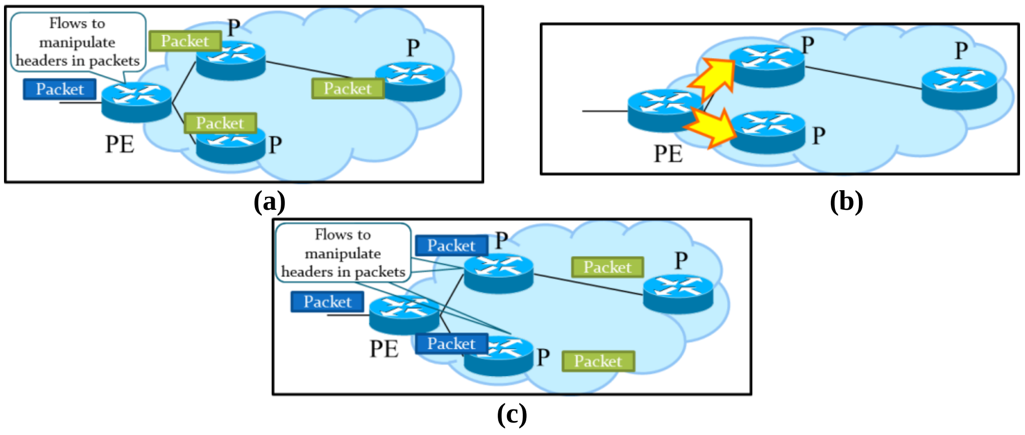


Figure 4. (a) PE router carries out flows that manipulate headers. **(b)** PE router gets overloaded and delegates some flows to the two adjacent P routers. **(c)** PE just forwards packets and the two P routers manipulate the headers.

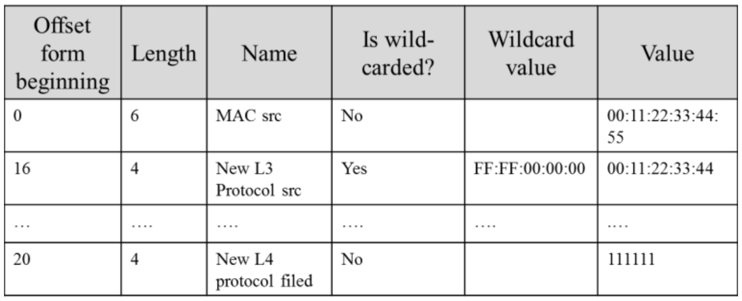
3. Low Level Header Description

According to OpenFlow, incoming packets are checked against the MAC header, IP header, VLAN header, and TCP/UDP header. As a result, researchers and application designers have to use the headers of those protocols to build their applications. However, being tied up to specific protocols headers limits the abilities to create new applications that might require new specially designed protocols in order to take advantage of the OpenFlow functionalities.

And so, we designed a simple method and a packet to be added to OpenFlow protocol to facilitate the use of low-level user defined headers. Using this method, the user will be able to program flows according to his needs by specifying the header fields of concern; by defining their offset form the beginning of the packet, the length of the field, wild-carded or not, wild-card value if it exists, and of course, the value to match against.

3.1. Protocol

In order to apply this enhancement to OpenFlow, a new packet is introduced into the OpenFlow protocol. It is very similar to the regular way that OpenFlow protocol manipulates flows, but has a list of offsets, length, wildcard related info, and values, (as shown in Fig.5) instead of the regular MAC, IP, VLAN, and TCP/UDP header fields.



Offset form beginning	Length	Name	Is wild-carded?	Wildcard value	Value
0	6	MAC src	No		00:11:22:33:44:55
16	4	New L3 Protocol src	Yes	FF:FF:00:00:00	00:11:22:33:44
…	….	….	….	….	….
20	4	New L4 protocol filed	No		111111

Figure 5. An example of a list of low level defined headers.

3.2. Usage Scenario

A simple scenario where the low level header definition will be useful is when the researcher wants to insert a field into the TCP header a head of the source and destination port numbers, which means that the Src and Dst port numbers will be shifted by the size of the new header field. Using the regular OpenFlow will not work, because OpenFlow matches the Src and Dst port numbers according to their position in the original TCP header not the modified one. Thus, using the low level header description method a flow can be programmed using a list of fields of newly customized TCP header.

4. New Type of Flows: Inactive Flows

According to OpenFlow, whenever a flow is programmed into a network equipment it will start matching against this flow and carrying out its actions. This means that the flow is activated and used as soon as it is programmed. This can be considered a significant advantage that the controller has immediate control over the network equipment. However, according to this model, there will be difficulty in dealing with cases that require precise timing, since that everything has to be done by the controller.

Having a centralized control would be of a greater advantage, if it can be coupled with the capability to operate precisely timed actions or flows. And to be able to provide OpenFlow with the combination of those capabilities we designed a new type of flows that are programmed into the network equipment as inactive flows, which means that those inactive flows will not be used by the equipment that they are programmed unless a certain condition activates them and enables the network equipment to use them. By having flows pre-programmed into the network, and by using them in the right time; is the method used by inactive flows to tackle the precision timing difficulty.

As explained before, that the inactive flows have to be activated in order to make use of them. We designed three conditions that can be used separately or as a combination to activate inactive flows. First condition is to receive a dedicated activation packet that contains a special activation token that can be sent by the controller, or a host, or another network equipment. The second activation method is to have an activation flow, through which a flow can be set as a condition to activate an inactive flow, and so whenever this activation flow is matched then the associated inactive flow will be activated. The third activation method is a specific time, through which a specific time is set upon which the flow will be activated.

And to be able to increase the efficiency of our inactive flows we propose to use network equipment to activate inactive flows on other equipment. I order to do so, we added to the inactive flow a list of addresses of network equipment that are programmed with the same inactive flow, to be used by the first equipment that has its inactive flow activated to activate the same flow on the rest of the equipment having this flow.

4.1. Protocol

In order to enable the use of inactive flows on OpenFlow network, the following changes have to be included in the OpenFlow Protocol. This can be done by having a new type of flow installation packet that is very similar to the original OpenFlow flow installation packet, but differs by having a special type that is the inactive flow, and contains the fields shown in Fig.6 in addition to, its other fields.

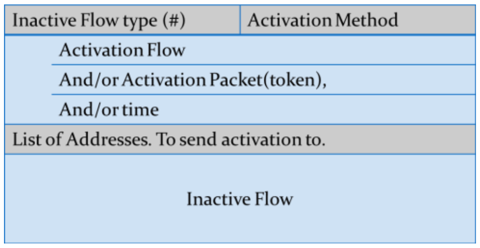


Figure 6. Fields added to the flow installation packet in order to support inactive flows.

4.2. Usage Scenario

To have a better understanding about the inactive flows; we will explain a scenario where a server intends to migrate form one machine to another while maintaining its IP address. Figure 7 (a) shows the path that client's packets take to reach the server as the red arrow, where the server is located in the machine on the right. When the server intends to migrate, it informs the controller about this. Next in Figure 7 (b), the controller would have issued the redirection flows to divert packets targeting the server towards the new machine that the server wants to migrate to as an inactive flows as shown in the dark yellow dashed arrow. After that, in Figure 7 (c) when the server finishes migrating, the nearest equipment to the old server detects the end of the migration by using the activation flow of its in active flow. Then, this equipment (nearest to the old server) activates this flow, and sends packets that have a special token to activate the same inactive flow on the other network equipment. Finally, the client's packets will be redirected towards the new position of the server by using the activated flow.

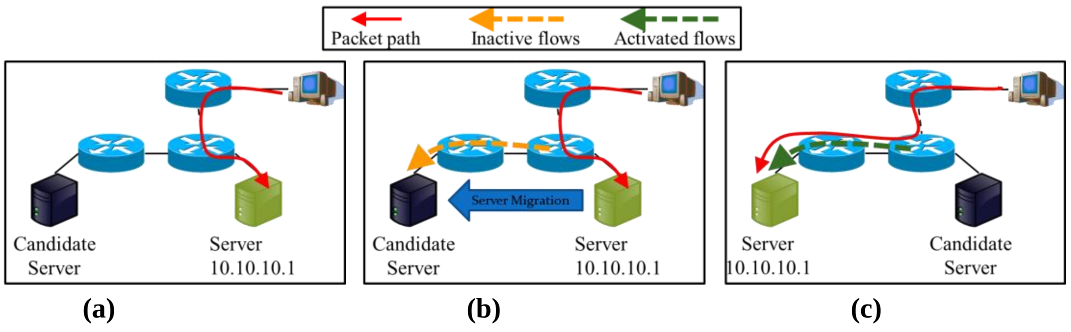


Figure 6. Server migration while maintaining IP: a usage scenario of the inactive flows. **(a)** Initial State. **(b)** migrating server and installing inactive redirection flows. **(c)** activating flows after server have migrated.

5. Evaluation

In order to, assess the effectiveness of our enhancements to increase the usability of OpenFlow. We are building three sets of simulations that we will run on the OMNET++ simulator [8].

The first set of simulations are intended to test the network equipment to network equipment flow installation; by running two simulations of networks, where the first simulation will run using the regular OpenFlow, and the second will run using an enhanced OpenFlow. In this set of experiments, we will study the efficacy of using the network equipment to equipment flow installation to delegate flows form an overloaded network equipment to another equipment in order to reduce its load. This will be done by measuring load on each of the network equipment in case of the enhanced OpenFlow and comparing that with the load in case of the regular OpenFlow. Also, we aim to measure the traffic generated by using the network equipment to equipment flow installation in case of delegating flows, and we aim to measure the time required to reduce load from the overloaded equipment.

The second set of simulations is designed to assess the use of the network equipment to equipment flow installation to distribute flows to other network equipment. We will run two simulations the first using the regular OpenFlow method, and the second using the enhanced OpenFlow. Trough those simulations, we will measure the time needed for the flow to reach all of the equipment, and the total amount of traffic needed to program the whole network.

The third set of simulations aims to assess the use of inactive flows. This will be done by using two simulations one using the regular OpenFlow and the second using OpenFlow with inactive flows. In those simulations, we aim to measure the delays needed before the redirection takes place in case of server migration.

6. Conclusions

Providing future Internet with technologies that enable it to play its role is extremely important. Because of that, many researchers are studying technologies to be the future Internet enabling technologies. OpenFlow is one of the future Internet enabling technologies, as it provides compelling functionalities that enable smarter applications to be built. However, there have been many concerns regarding the scalability of OpenFlow especially due to its dependence on a central controller.

For those reasons, our main goal in this study is to take OpenFlow one step towards having wider adoption as a future Internet technology. In more details, our study aimed to relieve some load off the controller, improve the usability of OpenFlow, make it more self-aware and able to react to situations like overloading some of its equipment in a manner that takes into account the current status of the traffic, provide users with the ability to define and use their own protocols, and to aid OpenFlow's ability to perform accurately timed operations. And to be able to achieve our aims we designed three enhancements to OpenFlow that are; the network equipment to equipment flow installation, low-level user defined protocol headers, and a new type of flows that is the inactive flows. In this paper, we showed our design of the protocols and algorithms needed by those enhancements to work, also showed some cases where those enhancements can play a significant role.

In order to assess the efficiency or our study; we are currently preparing simulations that we will use to measure the amount of traffic our enhancements will generate, delays needed to perform operations that are critically timed, the efficiency of our enhancements in reducing load on the controller, and the efficacy of our enhancements in introducing self-reactiveness to cases where equipment get overloaded.

References

1. Nick McKeown; Tom Anderson; Hari Balakrishnan; Guru Parulkar; Larry Peterson; Jennifer Rexford; Scott Shenker; Jonathan Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comp. Comm.* March 2008; Rev. 38, 2, pp. 69-74. DOI=10.1145/1355734.1355746 http://doi.acm.org/10.1145/1355734.1355746 .
2. W. Kim; P. Sharma; J. Lee; S. Banerjee; J. Tourrilhes; S.-J. Lee; P. Yalagandula. Automated and Scalable QoS Control for Network Convergence. In *Proc. INM/WREN*, San Jose, CA, Apr. 2010.
3. Othman Othman M.M.; Koji Okamura. Improvement of Content Server with Contents Anycasting Using OpenFlow. In *Proceedings of the 30 APAN of Network Research Workshop,* 2009.
4. S. Kandula; S. Sengupta; A. Greenberg; P. Patel. The Nature of Datacenter Traffic: Measurements & Analysis. In *Proc. IMC*, 2009.

5. Tavakoli, A.; Casado, M.; Koponen, T.; Shenker, S. (n.d.). Applying NOX to the Datacenter. *Proc. HotNets*, October 2009.
6. Kandula, S.; Sengupta, S.; Greenberg, A.; Patel, P.; Chaiken, R.. The nature of data center traffic: measurements & analysis. *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, ACM, 2009; pp. 202–208.
7. Heller, B. (2009). Openflow switch specification, version 1.0.0. Wire. December.
8. OMNeT++ Network Simulation Framework. (n.d.). http://www.omnetpp.org/.
9. Song, J.; Lee, S. S.; Kang, K.-C.; Park, N.; Park, H.; Yun, S., et al. Scalable Network Architecture for Flow-Based Traffic Control. *ETRI Journal* 2008, 30(2), 205-215. doi: 10.4218/etrij.08.1107.0035.
10. Hong, J. W. (2004). A flow-based method for abnormal network traffic detection. *2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No.04CH37507)* (Vol. 1, pp. 599-612). Ieee. doi: 10.1109/NOMS.2004.1317747.