Xperience of Programmable Network with OpenFlow

Hasnat Ahmed, Irshad, Muhammad Asif Razzaq, and Adeel Baig

***Abstract*—Today in the era of fast, modern and continuous changing world, we need to enhance and improve the existing routing and switching protocols for better performance. OpenFlow switches enable researchers to test and examine the behavior of newly designed protocol in their local environment without disturbing the existing production flow. This allows us to provide the minimize delay in production packet forwarding and maximum flexibility in controlling the flow of experimental packets through the network. In this paper, we have demonstrated to run a network topology on OpenFlow Virtual Machine Simulator (VMS) using NOX controller. Monitoring of flow tables on various switches has been analyzed. Later, we have developed a number of NOX controller components using C++ code to restrict Address resolution Protocol (ARP) on our network and to define dynamic programmable flows on OpenFlow switches.**

***Index Terms*—OpenFlow, NOX controller, VMS.**

I. Motivation and Need

Today in the era of fast, modern and enhanced communicational technologies, internet users and devices are growing very rapidly. This increase in internet applications require low delay and high performance, which is being provided in terms of high bandwidth using fast optical technologies. IPv4 addresses were initially designed to use for stationary devices, but with the pace of changing environment as mobile devices, requires new protocols to be designed and implemented, to fulfill the need of mobile devices and applications. Main focus is on presenting a feel to the mobile user as the same effect as if it is stationary. Keeping in view address space depletion of IPv4 and future requirements, serious efforts are desired in terms of new protocols to meet future challenges of scalabilities and performance for internet users and applications.

Transition from IPv4 to IPv6 is in progress. Protocols that were in use with IPv4 like Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), Internet Control Messaging Protocol (ICMP) etc. are modified to work with new IPv6 protocol. IPv6 header itself can contain a number of extension headers. Implementation of these new proposals requires to be tested without affecting the production network, so that on the basis of implementation limitations, unattended or missing requirements, these can be modified and enhanced.

A. Why OpenFlow?

By restricting our focus to the premises of a Local Area Network (LAN), it is possible to use a computer to behave as a switch, having multiple Network Interface cards (NIC); each one is connected to a different network. The system receives a packet from one network through one interface, applying processing on the basis of packet header information to decide on which interface or network it is to be forwarded. In this case, we can enjoy the maximum flexibility at the cost of significant delay in packet forwarding. These delays can be minimized by choosing some hardware based solution in terms of formal Ethernet switches. These switches use predefined protocols defined by the vendors and embedded inside the switches. Thus, using switches minimizes the delay in forwarding the packets at the cost of reduced flexibility. What's if someone wants to test or use a new protocol? When a new protocol is designed, it is not possible to implement it on the internet without proper testing, analyzing its behavior, efficiency and performance etc. So, the new protocol must be first properly analyzed on a private network using switches. The limitation is currently available switches use predefined protocols and vendors don't allow the flexibility to implement new protocols on these switches.

Manuscript received September 10, 2012; revised November 20, 2012.

The authors are with the School of Electrical Engineering and Computer Sciences National University of Sciences and Technology (e-mail: 10msithahmed@seecs.edu.pk).

II. Introduction

OpenFlow concept, its open source development and deployment was initiated at Stanford University. Research institutes and university campuses were encouraged to use and spread the idea in their premises. OpenFlow switches enable researchers to test and examine the behavior of newly designed protocol in their local environment, while still providing the minimize delay in packet forwarding and maximum flexibility in controlling the flow of packets through the network. The architecture of an OpenFlow enabled Ethernet switch is shown in Fig. 1[1]. Data path module is the heart of Ethernet switch which performs the actual forwarding of packets. Control path is used to control the decisions for forwarding the incoming packets to an appropriate output interface. Control path make use of protocols between switches in order to control the flow of communication.

Without disturbing the Ethernet switched functionality, OpenFlow adds an open flow module residing with control path inside the Ethernet switch. This open flow module is used to communicate with OpenFlow controller. The communication between controller and OpenFlow module must be safe through a reliable and secure channel. Open flow switch contains the internal flow table totally managed by OpenFlow controller. Flow table is managed through a standard interface which enables the addition, updating and removal of flow table entries. Routes between the nodes arecontrolled through these flow table entries. OpenFlow environment is described in the Fig. 2[1].OpenFlow switches enable the vendors not to expose the existing functionality of the switches and make it flexible for new protocols. For each flow there is an entry in flow table. Each flow table entry consists of three parts: Rule, Action and Stats. Rule is the

exact matching of packet header information and flow table entry. Action defines where to go if certain rule valid. Stats are the information associated with the flow entry like time elapsed after last use or packets and byte counters. By implementing the rule property in flow table, network researcher can experiment with the layer 2 as well as layer 3 functionalities. OpenFlow is not only being added as a feature to commercial switches but also to wireless devices such as WiFi Access Points and WiMAX base-stations.

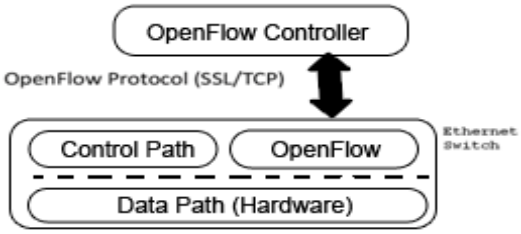

Fig. 1. Openflow switch.

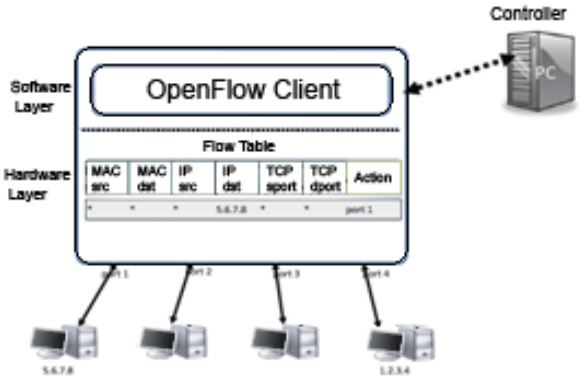

Fig. 2. Openflow system architecture.

III. Related Work

Before the emergence of OpenFlow enabled technology, researchers work on developing new protocols was limited. GENI [2] were facilitating the researchers to experiment the new routing protocols through allocating slices of the network resources. Researchers could use these slices to experiment their routing Algorithm. GENI assumes to implement the new routing protocols on nationwide scale. So it was a costly solution. OpenFlow has enabled the new networking experiments in a campus area [3].

OpenRoads [4] platform is engaged in exploring and experimenting with new solutions for Mobility including new routing protocols and controllers using OpenFlow. Community based open source framework development is being done to support mobility where the controller is handled with OpenFlow and device settings are managed through SNMP. Technology level abstraction is used by defining high-level control interfaces so that the development remains reusable, flexible and can be easily enhanced to support heterogeneous network. OpenRoads was tested on a topology consists of 5 switches, 30 Wi-Fi APs and one WiMax base station. Seamless handovers between different wireless technologies were successfully managed among mobility managers. To run researcher's networking experiment parallel with production traffic safely, there is a need of network slicing. So in OpenFlow [5] is introduced. Flow Visor is a special purpose controller that allows number of researchers and network administrators to work parallel in a network.

Demo [6] illustrated the experiment of demonstrating the network connection between two buildings of Stanford campus, through OpenFlow enabled routers and switches. Mobility support at data link layer and network layer were presented using OpenFlow controller. Several mobile clients each one is running a game application, which is connected with a game server, which is running inside a virtual machine. There are a number of different hosts on which the virtual machine can be migrated. Main focus was to maintain the lowest delay between mobile node and virtual machine. When a mobile client is moved, its connected virtual machine is migrated to the nearest host to the mobile client. The migration of VM is seamless to the mobile client application without requiring any change in IP or MAC address and maintains the existing connection between application program at client and server program running inside VM.

N-casting [7] means when a packet is received, it is multicast to n output interfaces, where at each interface possibly different radio devices is attached. In traditional way, it requires huge amount of bandwidth but OpenFlow made it possible with comparable low bandwidth exploiting OpenRoads with only an additional code of 227 lines.

N-casting [7] creates a network topology consists of devices working with different network wireless technologies WiFi and WiMAX and uses [4] OpenRoads to demonstrate a flow between these devices. The packets received from any of the device working with any network wireless technology can be powerfully examined and programmed and flow path can be defined to deliver it to any of the device.

IV. Achievements and Contribution

Challenges in this project have been identified in the form of following milestones:

1) Understanding of OpenFlow and running simple network using OpenFlow.
2) Exploring OpenFlow components architecture, its working and configuration of devices.
3) Developing NOX controller components in C++ that defines decision based dynamic flow entries in the OpenFlow switches.

V. Research Approach /Methodology

OpenFlow is an open source project, implemented using python scripts and C++ programming language. We have downloaded the currently running code, simulating network traffic flow of a simple network topology. By examining this code and after investigating how traffic flow can be monitored and controlled, we have extended this work first to develop our own controller components that can make decision to define dynamic overflows on the basis of received packet captured information. Our future work will develop controller components to support hierarchical mobility support.

VI. Benefits of the Project

A. Experiments

Different experiments can be carried out in a network topology using OpenFlow switch such as packet capturing, observing network performance and delay on changing packet size, packet loads, load balancing and traffic engineering etc. Moreover, new protocols can be tested and verified.

B. Cost

OpenFlow switches can be used to work on Layer 3 as a router. Buying or utilizing a router for a test bed costs very high as comparable to perform testing with OpenFlow switches.

C. Noninterference

New protocols can be tested with OpenFlow switches, even in production or existing network without disturbing its flow.

D. Heterogeneous Network

OpenFlow switches can be used to configure to work with heterogeneous network. Different radio devices can be connected to different interfaces of the OpenFlow switch and we can enable these devices to communicate with each other.

E. Consistent

It provides assurity to handle production traffic and experimental traffic consistently.

VII. Beneficieries of the Project

Following are the direct customers who can take advantage of using OpenFlow:

A. Researchers

Newly designed protocols functionality can be tested and their dependencies, new issues and performance factors can be verified using OpenFlow switches.

B. Educational Institutions

Academic campuses can simultaneously run their production network as well as network lab, where different experiments on network protocols and devices can be carried out.

C. Network Provider Industry

Infrastructure or service providers can test communication among devices using different technologies.

D. Network Manufacturing Industry

Industry is encouraged to produce flexible switches that enable customers to control and monitor the traffic flow using the software part of switches.

VIII. Outputs Expected from Project

Final results of this project will be demonstration of how can we dynamically define flow in the switches and make decision in defining flow on the basis of captured packet header information.

IX. Experiments and Results

A. Run OpenFlow VMS

Debian or Ubuntu along with GUI or desktop packages is the first requirement. We have installed Debian Linux (Lenny) on our system. One can use a virtual machine like VMWare or Virtual Box to run Linux on top of Windows. Then we used Synaptic manager utility to install all packages dependencies for OpenFlow VMS. After that, downloaded and setup OpenFlow VMS. Finally, we configured our network topology in the form of xml and verified it by running, as shown in Fig. 3.

B. Run OpenFlow Controller

We have successfully configured the controller by downloading all the dependent packages and enabled the controller's listener port. We have installed NOX controller. After successfully running controller, we start our network topology, as defined in xml configuration.

It virtually creates four interfaces and four command screen are displayed as shown in Fig. 4. We can verify through some basic commands that all host and open flow switches are connected or not.

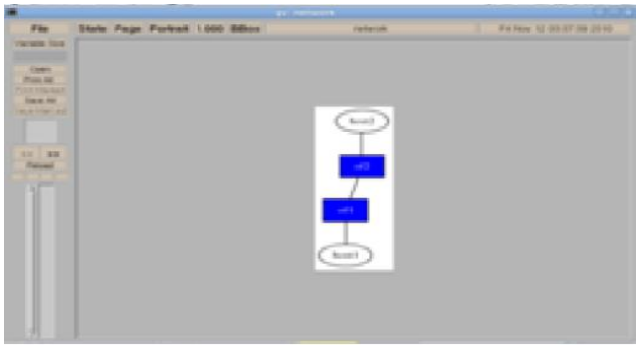

Fig. 3. Openflow virtual machine simulator.

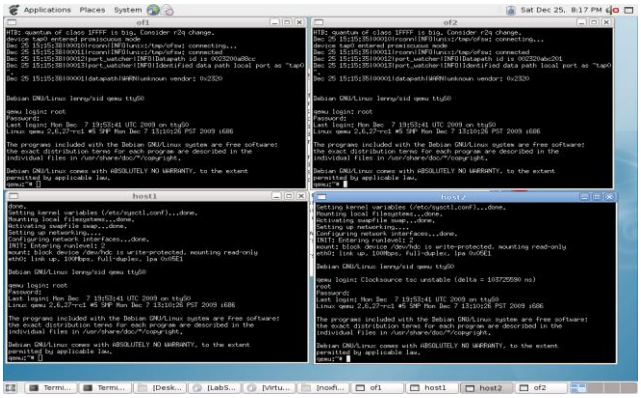

Fig. 4. Openflow VMS running topology.

C. Develop Controller Component (Blocking ARP)

Now we want to create a new component that will restrict the ping functionality on the network. We see that when we ping another host, flow entries are created in OpenFlow switches of1 and of2.

Actually what happens is when an ARP message is sent to switch, there is no entry in OpenFlow switch, means no action is defined what to do for ARP request. The request is forwarded to controller by the switch to get flow entry. The controller analyze the request and posts a flow in which it is define that flood the packet from all interfaces except from

where it is received. By our code, we restrict the ping command by using the strategy that controller will not send a flow to the switch when it analyze that the received request is an ARP request packet. In this way, as no flow is returned to switch, the switch will get no action and ping ICMP packets will not sent to other switch, resulting ping command restricted.

D. Develop Controller Component (Flow Attributes)

Now to start creating another new component hubTimeout. In default hub component, you can see flow entries with idle_timeout value 5, as shown in Fig. 5. If within 5 seconds, this entry or flow is not used by the switch, it will be purged after 5 seconds. We want to modify our hub component code to enhance this value of idle_timeout and hard_timeout from 5 to 100. Actually, the theme of this exercise is how to set the value in a flow. Using the approach in this practice, you can set or modify any field or column value of a flow dynamically before sending it to a switch as shown in Fig. 6.

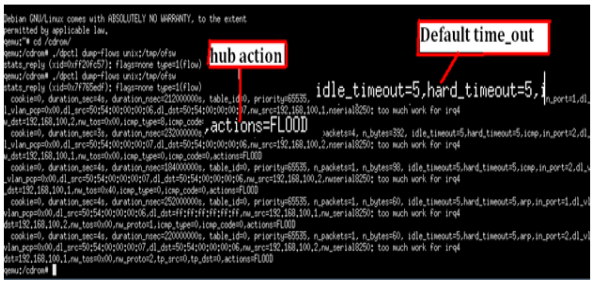

Fig. 5. Idle_timeout flow entries before modification.

E. Develop Controller Component (Routing)

We will modify the hub code to get switch behavior. In hub, the Action for the flow entry is Flood; means transmit the packet from all interfaces except the interface from it has received. In switch, when a packet received, we note the source address (say s) and receiving interface (say I) and define the flow that if destination address is s, action value is output to interface I. We can see these

We can see these entries on OpenFlow switch so we can route the packet with our defined flow.

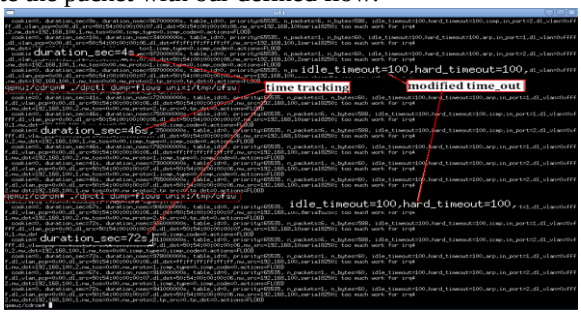

Fig. 6. Idle_timeout flow entries after modification.

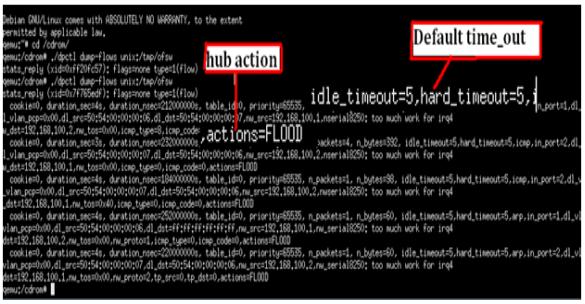

Fig. 7. Flow entries showing hub_action flood.

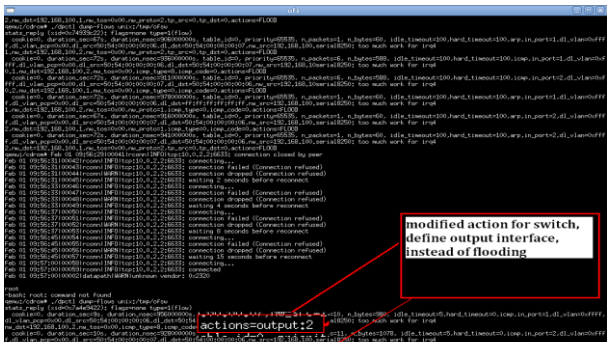

Fig. 8. Openflow switch entries showing switch behavior.

X. Conclusions and Future Work

We have demonstrated the concept of programmable networks using OpenFlow. We installed OpenFlow VMS and NOX Controller, then developed three different controller components using C++. Our future work is developing controller component for hierarchical mobility.

References

[1] OpenFlow Hands - on Tutorial. HOT - Interconnects 2010. [Online]. Available: http://www.OpenFlow.org/downloads/HOTITutorial-OpenFlow.pdf

[2] Global Environment for Network Innovations. [Online]. Available: http://geni.net.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, and L. Peterson, "Jennifer Rexford, Scott Shenker, and Jonathan Turner OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, April 2008.

[4] K.-K. Yap, M. Kobayashi, R. Sherwood, N. Handigol, T.-Y. Huang, M.l Chan, and N. McKeown, "OpenRoads: Empowering research in mobile networks," in *Proceedings of ACM SIGCOMM* [Posteer], Barcelona, Spain , August 2009

[5] R. Sherwood, M. Chan, G. Gibb, N. Handigol, T.-Y. Huang, P. Kazemian, M. Kobayashi, D. UnderHill, K.-K. Yap, G. Appenzeller, and N. Mckeown, *Carving Resarch Slices out of your Production Networks with OpenFlow*.

[6] D. Erickson, B. Heller, D. Underhill, J. Naous, G. Appenzeller, G. Parulkar, N. McKeown, M. Rosenblum, S. Kumar, V. Alaria, P. Monclus, F. B. J. Tourrilhes, P. Yalagandula, S. Banerjee, C. Clark, and R. McGeer, **"**A demonstration of virtual machine mobility in an OpenFlow network," in *Proceedings of SIGCOMM*, Seattle, Washington, USA, August 17-22, 2008.

[7] K.-K. Yap, M. Kobayashi, R. Sherwood, G. Parulkar, T.-Y. Huang, M. Chan, and N. McKeown, *Lossless Handover with N-Casting between Wifi-WiMax on OpenRoads*.

Hasnat Ahmed received the Bachelor degree in the Software Engineering, from NUML, Pakistan and doing MS degree in the IT, NUST, Pakistan from 2010. He has seven years experience of software development. In MS, his research interests include programmable networks, database mining and pattern designing.

Adeel Baig received PHD degree in the Computer Science & Engineering, UNSW, and Australia. He is a Assistant Professor of Computer Science and Engineering in NUST. His research interests include Protocols, IP technologies, Programmable Networks etc.

Irshad did the MCS degree from Karachi University, Pakistan and doing MS degree in the IT, NUST, Pakistan from 2010. He has eight years experience of software development. In MS, his research interests include programmable networks, Ontology's and Database Mining.

Muhammad Asif Razzaq completed the MCS degree from FUUAST, Pakistan and doing MS degree in the IT, Nust, Pakistan from 2010. He has ten years experience of software development. In MS, his research interests include programmable networks, database mining and pattern designing.