OpenFlow Implementation on NetMagic Platform

Chunbo Jia, Jinfeng Huang, Qi Su, Zhigang Sun, Tao Li, Hua Zhang
School of Computer, National University of Defense Technology
Changsha, Hunan 410073, China
jiachunbo1988@sina.com

Abstract—Stanford University proposed a new network switch model named OpenFlow for the research of network innovation. OpenFlow enables researchers to control the behavior of the network by managing flow table. NetMagic is an open network switching platform for the innovative research of next generation Internet architecture. This paper introduces OpenFlow and NetMagic briefly, and implements a customized and scalable OpenFlow model system on the NetMagic platform. Finally, this paper shows the features and advantages of OpenFlow with an experiment.

KEYWORDS: OpenFlow, NetMagic Platform

I. INTRODUCTION

Switching equipment has become commercial, and network equipment vendors closed up their equipment for their commercial interests, which seriously hindered the development of network innovation. Therefore, how to provide a test platform for network innovation without exposing the internal structure of network device has become the key of network innovation. Professor Nick McKeown from Stanford University proposed OpenFlow [1] to achieve this goal. OpenFlow separates production flow with experiment flow. In addition, OpenFlow enables network experiments without affecting production flow. Therefore, OpenFlow is a suitable way to enable network innovation.

NetMagic platform [2] proposed by NUDT (National University of Defense Technology) is a good platform to implement OpenFlow model. NetMagic is an open network switching platform for the innovative research of next generation Internet architecture. NetMagic has been used by researchers to model and verify new protocols. In addition, NetMagic has also been used by teachers to help students learn how to build Gigabit Ethernet (GigE) switches and Internet Protocol (IP) routers. NetMagic has characteristics of reconfigurable, reprogrammable, high port density, fast processing and so on.

Of course, the implementation on the NetMagic platform is not the only way to implement OpenFlow model. There are many other ways to implement OpenFlow model, such as network equipments with OpenFlow features and NetFPGA platform [3] etc. However, these methods all have some disadvantages to implement OpenFlow model. Network equipment isn’t open and can’t update with the development of the OpenFlow. The implementation with NetFPGA platform is reconfigurable. However, due to the limit of performance and port-density, the implementation on NetFPGA platform can’t be used in large-scale network.

However, the implementation on the NetMagic platform resolves these questions. First, NetMagic platform is reconfigurable and reprogrammable, so the implementation on the NetMagic platform can update with the development of the OpenFlow model. Second, NetMagic platform has novel architecture and advanced performance, so the implementation on the NetMagic platform can be used in the large-scale experimental network.

II. OPENFLOW COMPONENTS

In the traditional routers, the control logic and the data path are co-located on the same device. But in the OpenFlow model, the control logic is removed to the external controller, and the network device is simplified to a part of data path.

OpenFlow model consists of the OpenFlow switch and the external controller. The detailed components of the OpenFlow model are shown as Figure 1.

A. OpenFlow switch

OpenFlow switch is the core components of the OpenFlow model, which is mainly used to manage the packets forwarding.

According to the degree of the support for OpenFlow, OpenFlow switches can be divided into dedicated OpenFlow switch and OpenFlow-enabled switch. According to the development of the OpenFlow, OpenFlow switches can be divided into "Type0" switch and "Type1" switch. "Type0" only supports 10-tuple header fields and only four actions, as shown in Figure 1; "Type1" supports more features to support complex network tests, such as packet filtering, multicast packets forwarding etc. Following, we explain OpenFlow switch in the way of "Type0" OpenFlow switch.

OpenFlow switch consists of flow table, secure channel and OpenFlow protocol.

Flow table consists of a number of flow table entries, and each flow table entry is a forwarding rule. Packets which
arrive in the switch get processing rules by looking up the flow table. Flow table entry consists of header fields, counters and actions, of which the header field is a 10-tuple and the identifier of the entry; counters are used to count the bytes or packets; actions mark how to process the data packets which matches the flow table entry.

Secure channel is an interface that connects OpenFlow switches to the controller. Through the interface, the controller can manages and configures the OpenFlow switch. Furthermore, the controller can receives the events from the OpenFlow switch and send packets to the OpenFlow switch. All messages that OpenFlow switches and the controller communicate over the secure channel must be formatted according to the OpenFlow protocol.

OpenFlow protocol provides an open and standard interface which is used by the controller to communicate with each OpenFlow switch. The core of the OpenFlow protocol is the set of structures used for OpenFlow Protocol messages. OpenFlow protocol supports three types of messages in the OpenFlow model: Controller-to-Switch, Asynchronous and Symmetric, and each type have several sub-types.

B. Controller

OpenFlow model removes the control logic of the traditional router to the external controller, so the controller implements the function of making forwarding decisions. The packet arrives in the OpenFlow switch. If no match is found in the flow table, the packet will be encapsulated to the controller. The controller is responsible for decide how to handle the packet which arrives in the controller. In addition, controller can manage and maintain the flow table by removing, adding and updating the flow table entries.

Controller can communicate with the OpenFlow switch over the secure channel using the OpenFlow protocol. Controller can implement all the functions above by running control software, such as NOX [4]. NOX formats the message that the controller sends to the OpenFlow switches and provides well-defined interfaces to support the development of the applications, so NOX is similar with the operating system of the network. Now, the researchers have explored some applications on the NOX, such as Plug-n-serve, OpenRoads and OpenPipes etc.

III. ABOUT NETMAGIC

NetMagic is an open network switching platform for the innovative research of next generation Internet architecture. NetMagic enables researchers and students to build high-performance networking systems and verify their new ideas and methods for network innovation. The platform is designed with a novel patented architecture, where a common high-density Field Programmable Gate Array (FPGA) device with a combination of commodity Ethernet switch chip can provide the both high-speed Gigabit Ethernet switching capacity and reconfigurable user-defined packet handling function. NetMagic has been used by researchers to model and verify new protocols. Furthermore, NetMagic has also been used by teachers to help students learn how to build Gigabit Ethernet (GigE) switches and Internet Protocol (IP) routers.

NetMagic platform consists of hardware platform and software platform which implements on the remote host.

NetMagic hardware platform directly connects to the end-system. It provides a Fast Ethernet port {P0} to manage the NetMagic platform and 24 Gigabit Ethernet data ports {P1, P2, ..., P24} to process data flow. NetMagic is a combination of commodity Ethernet switch chip and FPGA, of which commodity Ethernet switch chip provide the function of fast data processing and the FPGA make NetMagic platform reconfigurable. NetMagic platform Pre-integrated the management module which supports the remote host to control the hardware platform by the fast Ethernet port in the way of controlling the flow table and overwrite the register etc. The basic structure of the NetMagic hardware platform is shown as Figure 2.

![Figure 2. the architecture of the NetMagic platform](image)

NetMagic software platform is basic on the basic network protocol stack of the remote hosts. It encapsulates the control message described in the NMAC protocol which is the communication protocol between the NetMagic hardware platform and the remote host in the form of standard MAC packet. NetMagic software platform provides well-defined interface to support the development of the applications. In addition, the control message is formatted in the form of standard MAC packet, so the remote host can control the NetMagic hardware platform across networks.

IV. IMPLEMENTATION

As OpenFlow played more and more significant role on the network innovation, there have been many commercial switches which enable OpenFlow. But commercial switches can’t update with the development of OpenFlow. Moreover, commercial switch isn’t open, so researchers can’t improve OpenFlow model with commercial switches. So, in this section, we describe how to implement reconfigurable and customizable "Type1" OpenFlow with NetMagic platform.

NetMagic hardware platform is used to implement OpenFlow switch, and NetMagic software platform is used to implement external controller.

The controller is divided into two parts to implement, which are network topology discovery and making forwarding rules. First, according to the message from the link state module, controller can maintain and update the network topology. Second, with the known network topology...
and special routing algorithm, controller can decide the forwarding rules to the packets which arrive in the controller.

The processing logic of the OpenFlow switch in the User Module is shown as Figure 3. When packets arrives in the UM, they will be buffered in the Packets Memory. Simultaneously, Header parser module will extract key words which will be used in the Look-up module for looking up the flow tables. According to the result produced by the Look-up module, Packets Process modules will implement corresponding actions to the packets. Link state Module is responsible for perceiving status of the adjacent nodes and forwarding these status messages to the controller.

![Figure 3. The processing logic in the User Module](image)

The implementation improves the traditional OpenFlow model in two points.

First, because the NetMagic platform is reconfigurable, the implementation of the OpenFlow model in the paper is customizable. In the implementation, researchers can extract keys according to their requirement, instead of extract 10-tuple in the OpenFlow specification. The implementation improves the adaptability and efficiency of the OpenFlow model.

Second, the implementation improves the forwarding mechanism in the OpenFlow specification. The packet arrives in the OpenFlow switch. If no match is found in the flow table, the key of the packet will be encapsulated to the controller instead of the packet entirely. According to the key, controller can generate the rule how to handle the packet. Obviously, this improvement can reduce the bandwidth consumption of the controller and make it more scalable.

V. EXPERIMENT AND ANALYSIS

In the section, an application of video forwarding will be implemented on our implementation of OpenFlow model to check the effectiveness and show the features of the OpenFlow model.

![Figure 4. The topology of the experimental network](image)

The topology of the experimental network is shown as Figure 4. In the experimental network, Terminal 1 is the video source; Terminal 2 is the receiver and monitor; Terminal 3 is responsible for send background flow; A, B, C, D is the OpenFlow switch implemented on the NetMagic platform.

The process of the experiment is introduced as follows. Normal video flow is forwarded through A->B->D. Simultaneously, we add background flow to disturb the normal video flow and observe the change of the DF value. Then, we change the forwarding path of the normal video to avoid the background flow and observe the change of the DF value again.

The tests and results are shown as follows:

1) When there is no background flow in the network, the DF values calculated in terminal 2 are shown as below:

![Figure 5. The result of DF value in test 1](image)

Sample variance of DF values:

$$\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2 = 1.557032 \quad (1)$$

Sample expectation of DF values:

$$\frac{1}{n} \sum_{i=1}^{n} x_i = 17.32192 \quad (2)$$

In the Figure 5, except some special points, DF value is smooth overall. So, our implementation has the capability to process the packet at line-rate.

2) Terminal 3 sends normal IP packets to the BC link of the experimental network. Simultaneously, we monitor the LabelCast video flow and calculate the DF value in terminal 2. The DF values are shown as below:

![Figure 6. The result of DF value in test 2](image)

Sample variance of DF values:

$$\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2 = 6.139696 \quad (3)$$

Sample expectation of DF values:

$$\frac{1}{n} \sum_{i=1}^{n} x_i = 16.87368 \quad (4)$$

In the Figure 6, DF value is an important indicator reflected the delay and jitter of the video flow. DF value is bigger, and the delay and jitter of the video flow is more seriously, anti-vice versa.
\[
\sum_{i=1,2,\ldots,n} (x_i - \bar{x})^2 = 2.960085 \quad (3)
\]
Sample expectation of DF values:
\[
\sum_{i=1,2,\ldots,n} x_i = 18.03864 \quad (4)
\]

By comparison, sample variance and sample expectation of DF values in Test 1 and Test 2 don’t have significant differences. Thus, the normal flow hardly impacts the experimental flow in the switching nodes.

3) First, Terminal 3 sends a mass of normal IP packets to congest the BD link of the experimental network. Then, controller changes the transmission path of the video flow to A-> B-> C-> D. Simultaneously, we monitor the LabelCast video flow and calculate the DF value in terminal 2. The DF values are shown as below:

![DF value](image)

Figure 7. the result of DF value in test 3

Sample variance of DF values after changing the transmission path:
\[
\sum_{i=1,2,\ldots,n} (x_i - \bar{x})^2 = 2.682135 \quad (5)
\]
Sample expectation of DF values after changing the transmission path:
\[
\sum_{i=1,2,\ldots,n} x_i = 17.99686 \quad (6)
\]

By comparison, after changing the transmission path, sample variance and sample expectation of DF values in Test 1 and Test 3 don’t have significant differences. Therefore, the video flow recovers normal by changing the forwarding path. Note that, to observe the change of the DF value more clearly, the forwarding path is set to change after the DF value exceeds the threshold for 20s.

We conclude that OpenFlow can implement experimental flow and production traffic flow on the same physical network without interference each other. In addition, researchers can control the behavior of the network in the way of OpenFlow.

VI. CONCLUSION

OpenFlow separates the control logic and the data path, and simplifies the underlying physical device. OpenFlow enables researchers to implement experimental protocol over production network. In this paper, we implement OpenFlow model on NetMagic platform. In the implementation, we improve the traditional OpenFlow model in two points, which are forwarding the key to the controller and making it customizable. By the experiment on the implementation, we conclude that our implementation is capable of handling all the packets at line-rate. Simultaneously, the experiment shows some advantages of OpenFlow model.

While OpenFlow has many advantages, it also has some issues to address. These issues include how to protect the controller from service attack, how to improve its scalability, how to handle the dynamic of network topology. DIFANE [5] represent the direction of OpenFlow, which is a pragmatic compromise of centralized and distributed.

ACKNOWLEDGMENTS

This work was supported by the National Basic Research Program of China (973 Programs) (No.2009CB320503), and the National High Technology Research and Development Programs of China (863 Programs) (No. 2008AA01A325 and No. 2008AA01A323).

REFERENCES


