OpenFlow+: Extension for OpenFlow and its Implementation
Hongyu Hu, Jun Bi, Tao Feng, You Wang, Pingping Lin
Network Research Center, Tsinghua University, China
4-204, FIT Building, Beijing 100084, China
huhongyu@cernet.edu.cn, junbi@tsinghua.edu.cn, fengt09@mails.tsinghua.edu.cn,
{wangyou, linpp}@netarchlab.tsinghua.edu.cn

ABSTRACT
The Internet has made great success and big progress. However, the network-layer of Internet and the network devices in Internet have been relatively stagnant. Few changes or improvements have been made in last forty years, which is a stark contrast to the prosperity of the application-layer of the Internet. OpenFlow aims to enable innovation for the network-layer and network devices, which decouples the traditional network devices and rebuilds each module of the devices in an open and standardized way. But there are still some challenges that need to be overcome before OpenFlow can become more practicable and more usable, and ultimately, achieve larger scale commercial applications. Based on our analysis of OpenFlow, we proposed some extensions for OpenFlow (OpenFlow+) in this paper, including these four aspects: standard hardware extension, control mode extension, communication interface extension, and low-cost FlowTable realization. Next, the implementation details of OpenFlow+ in a commercial router are described, and finally, two new applications based on OpenFlow+ are demonstrated. We believe that OpenFlow+ will help to speed up the business development of OpenFlow and promote its large-scale use and applications.

Categories and Subject Descriptors
C.2.1 [Network Architecture and Design]: Network communications

General Terms
Design, Standardization, Experimentation

Keywords
OpenFlow, Extension, OpenRouter, TLV, Implementation

1. INTRODUCTION
After forty years of development, the Internet has made great success and become one of the critical infrastructures of our businesses, studies and daily lives. The architecture of today’s Internet is relatively stagnant due to the designing principle of “Keeping the simplicity of network while leaving the complex processing tasks to hosts” [1]. The functions of the application-layer have been greatly enriched because the applications on hosts can be flexibly modified and deployed. Meanwhile the network-layer has become like a steel plate and the network devices have become like opaque black-boxes because of the lack of openness in the network-layer. We can list many related limitations: debugging on network/network devices is troublesome; new network services are difficult to deploy; also the network itself lacks self-adaption and self-adjustment, etc. Words such as closed, inflexible, unmodifiable and so on have been increasingly used to describe today’s Internet [2].

OpenFlow is proposed with the aim to enable innovation for the networks [3]. The major changes to the networks brought by OpenFlow include: 1) The control planes of the network devices, which often require changes, are separated from the data planes of the devices, since the data planes are relatively rigid. This change would lay the foundations for future free changes in control logics. 2) The control planes are shifted out of the devices and centralized on a control server called Controller. Based on this change, the control logics could be easily modified. Meanwhile, OpenFlow protocol is designed for the Controller and the data planes to exchange information. 3) A new type of hardware--FlowTable is designed and added to each network device. FlowTable has the basic functions of the data plane, and actually is the abstract of the data plane, we think, and can be managed from outside devices. When outside various control logics need to control the data plane, FlowTable is the only hardware that they need to access. So FlowTable is an open and standardized hardware resource, which will facilitate the change, design and deployment of control logics.

OpenFlow decouples the traditional network devices, and rebuilds each module of the devices in an open and standardized way. Each module can implement innovation and evolution individually. We believe that this is a first step towards an evolving network. The valuable thoughts that OpenFlow brought to us reflect in the following aspects, as shown in Figure 1. 1) The design of FlowTable in the data plane realized the standardization, simplification and openness of network hardware. 2) The design of OpenFlow protocol realized the standardization and openness of network hardware access interfaces. 3) User-defined control logics can be easily added to the Controller as new components. This provided a unified and standardized way to organize the logics in the Controller. 4)
The centralized computing mode designed in OpenFlow makes some functions or services based on global information possible. This made up the weakness that the distributed computing in current networks presents. For example, QoS routing and traffic engineering, which need global analysis, can be easily realized in the centralized computing mode.

![Diagram](image)

Figure 1. The contribution of OpenFlow to network architecture.

However, we believe that some improvements or extensions should be made to OpenFlow:

1. Standard hardware in OpenFlow needs to be extended. OpenFlow only provides a type of standard hardware—FlowTable. FlowTable can only implement some basic forwarding functions of the network devices, but it cannot fully meet all the requirements of users to control and operate the network devices, such as packet sampling functions and rate limit functions of QoS with hardware, etc. One way to provide more standard hardware resources to use is to open and standardize the existing and mature hardware resources, such as ACL&QoS tables, FIB tables and sampling hardware, etc.

2. FlowTable hardware in OpenFlow can be realized low-costly and quickly. Currently, to realize an OpenFlow switch, a new and additional hardware—FlowTable, is required to be added to it. However the cost of OpenFlow implementation in a commercial router is very sensitive to the device vendor and user. We think the implementation of FlowTable can make use of existing mature hardware resources in a commercial device, such as ACL&QoS tables and FIB tables. Opening these hardware resources for external control logic to use not only can realize the forwarding function of FlowTable but also can reduce the cost of FlowTable’s implementation and development. Lowering the cost of FlowTable implementation using existing hardware inside a router is a basic premise of large-scale commercial implementation and deployment of OpenFlow.

3. The control mode needs to be extended. The centralized control mode can implement many new network functions, but pure centralized mode will reduce control efficiency in some circumstances comparing with the distributed control mode, in which the control functions are directly running inside each network device. Meanwhile, the traditional distributed computing mode that brings mature functions and mature productions, such as OSPF routing function, intra-AS topology discovery, etc., should also be utilized by outside control logics. Therefore, we think it is necessary to design a mixed control mode so that the centralized control logics in the OpenFlow controller and the distributed control logics in each device may be integrated to achieve stronger and more effective control functions and effects.

4. The communication interface needs to be extended. OpenFlow protocol is an interface between the control plane and the data plane used to transfer FlowTable, packet and network state information, etc. These information usually has fixed length and fixed format, so OpenFlow protocol currently is designed to describe each information with fixed length and fixed format[4]. While this message type cannot organize or express uncertain and variable-length information effectively and clearly, such as the next-hops information (0 ~ n) in a routing table entry. On the other hand, OpenFlow protocol cannot conveniently support data information modification and add new data type, such as extending current FlowTable from IPv4 to IPv6.

Based on our analysis of OpenFlow, we will propose some extensions for OpenFlow (which we call as OpenFlow+) in this paper, and illustrate the implementation of OpenFlow+ in a real commercial router, and finally demonstrate new applications based on OpenFlow+.

2. OPENFLOW+: OUR EXTENSION FOR OPENFLOW

The extensions and improvements we have made for OpenFlow are shown as follows, including four aspects:

2.1 More standard hardware resources are exposed and extended

We have done deep analysis of many application scenarios of OpenFlow. Through these analyses, we concluded that the simplex packet forwarding function provided by FlowTable hardware is not enough for many practical applications, while other hardware functions within devices, such as QoS and sampling hardware functions, are also very useful and indispensable for some applications. Moreover, we discovered that the ACL&QoS tables, FIB tables, sampling hardware and other hardware functions within current vendor’s devices have become very mature and very transparent. Also, the implementation difference of these hardware functions among different vendor’s devices has become smaller and smaller. These hardware
functions have become the de facto standard mature hardware resources.

Therefore we expose or open the ACL&QoS tables, FIB tables, and sample functions of the current equipments for the external control logic software to use and manage in OpenFlow+, which are serving as independent standard open hardware resources, same as the FlowTable. Control logic software outside of the devices uses and manages these new standard open hardware resources inside of the devices through standard communication messages (the extended OpenFlow protocol) just as the way it uses FlowTable. This extension for OpenFlow is shown in Figure 2, in which the legend is the same as the one in Figure 1. In the left of Figure 2, only FlowTable is exposed; while, in the right of Figure 2, many other standard hardware resources are exposed.

2.2 Coexisting collaborative mode of distributed computing and centralized computing is designed

We deduced that neither the pure external centralized control mode nor the pure internal distributed control mode is enough for complex businesses or applications. The pure external centralized control mode is deficient in control efficiency and function maturity, while the pure internal distributed control mode is deficient in global coordinate computing. Therefore, we designed a coexisting collaborative mode of distributed computing and centralized computing, in which distributed computing control logic inside the devices and centralized computing control logic outside the devices exist simultaneously in the same network. They exchange data information by standard communication messages, and collaboratively manage and control all network hardware resources to realize complex and advanced functions and applications for users. We call this new mode as the Coexisting Collaborative Computing Mode (CCCM), shown in Figure 3.

In the coexisting collaborative computing mode (CCCM), control logics exist not only inside of each device but also outside of the devices. The control logics outside of the devices are centralized in a controller server, which serve as the centralized computing. And the control logics inside of the devices serve as the distributed computing. How the internal distributed computing cooperates with the external centralized computing is an important problem in CCCM. The rules that confirm both types of computing to work together and collaborate smoothly in CCCM are designed as follows:

(1) Control logic inside devices and the one outside devices exchange data information through the standard message interface when they interact (Interface1, as shown in Figure 3). On the left side of Interface1, it is the OpenFlow Agent module, on behalf of the control logic outside; on the right side of Interface1, it is the control logic inside. For example, each routing entry information in routing table generated by internal mature distributed routing protocols, such as OSPF, can be encapsulated and send to the control logics outside devices which need these data information. On the other hand, the user-defined routing protocols can also set their own route table items to routing table via control logics inside devices;

(2) Control logic both inside devices and outside devices have the same abilities to make control operations to the open hardware resources inside devices by the standard message interface (Interface2, as shown in Figure 3). These operations may occur individually, or simultaneously. Both internal control logic and external control logic can access or control FlowTables, ACL&QOS tables and FIB tables. For example, the two can issue new table entries into these hardware tables, or get content information from these hardware tables. Similarly, internal and external logic are both available to configure xFlow[5] module and receive xFlow hardware sampling messages independently;
(3) When hardware resource control conflict occurs due to the controls from the internal and external control logic, we designed the following rules to resolve these conflicts:

(i) Establish an arbitration module to execute the optimal selection for both controls from the inside and outside. For example, the routing table entries generated by both external logic and internal logic will be sent to the route management module (Router Table Management, RTM) to arbitrate, and the optimal routing table entries will be selected by RTM and stored in FIB terminally;

(ii) Or using fixed priority levels to choose the optimal control. For example, we can preset the higher priority level for the routing information generated by external logic and the lower priority level for the routing information generated by internal routing protocol. So when the routing table entries from external logic with the same destination address as the ones from internal logic are issued to the open standard hardware resource – FIB, the external routing table entries will be always selected and written into FIB hardware finally;

(iii) Or both of the control operations will coexist. For example, the internal logic can issue ACL items to the hardware, and external logic can also issue ACL items to the hardware. Both of the ACL items issued will be retained and co-exist in the hardware ACL tables, and the terminal effect will be determined by the hardware of the devices.

2.3 Communication messages are reorganized by TLV format

To support more types of information exchange between the control plane and the data plane, and to support the easy extension of the length of existing information in the OpenFlow protocol, we introduce TLV (Type Length Value, TLV) format to the OpenFlow protocol to reorganize the information in it. TLV format not only can efficiently organize data with variable length, but also can conveniently implement the extension for the length and type of data, and also can increase the scalability and robustness of the communication messages, which all benefit from TLV’s clear structure.

In TLV format, each piece of data is organized by the triple of (Type, Length, Value). Information in OpenFlow protocol is arranged and described in an orderly and explicit way, and can be extended easily in both the length and the types of the data. If there are multiple pieces of data needed to be transported simultaneously in one OpenFlow protocol message, the TLVs of the data should be inputted and closely arrayed one by one in the message. Since the type and length of each piece of data in each TLV can be confirmed easily through TLV’s “Type” field and TLV’s “Length” field, we can encapsulate and arrange these TLVs in random order. This arrangement of TLVs will not create the confusion of information as long as there is no blank between each TLVs. Meanwhile TLV can be used or arranged recursively, which means one TLV can be embedded into other TLV serving as its sub TLV. The general format of one TLV is as follows:

**Table 1. TLV general format.**

| TLV Type = XXX (Fixed length) | TLV Length = XXX (Fixed length) | TLV Value = XXX (“TLV Length” length) |

Example 1 for how to use TLV: The IP address in the FlowTable transported by OpenFlow Protocol now is in IPv4 format. It will become very easy for us to extend IPv4 address format to IPv6 address format if the TLV format is used in the organization of FlowTable. The only change needed is for the sender to modify the length of the IP address TLV from 32 bit to 128 bit, which is shown in Table 2-3. And there is no change needed for the receiver to process.

**Table 2. IPv4 Address TLV.**

| TLV Type = SIP (Fixed length) | TLV Length = 32 (Fixed length) | TLV Value = 1.1.1.1 (32 bit) |
| TLV Type = DIP (Fixed length) | TLV Length = 32 (Fixed length) | TLV Value = 2.2.2.2 (32 bit) |

**Table 3. IPv6 Address TLV.**

| TLV Type = SIP (Fixed length) | TLV Length = 128 (Fixed length) | TLV Value = ::1.1.1.1 (128 bit) |
| TLV Type = DIP (Fixed length) | TLV Length = 128 (Fixed length) | TLV Value = ::2.2.2.2 (128 bit) |

Example 2 for how to use TLV: when we transmit route table entries by OpenFlow protocol, it is hard to describe the next-hop information by the existing fixed data structure because the number of next-hop in a route entry is diverse and not fixed, which means next-hop maybe range from 0 to n. However it is easy to describe the next-hop number information if TLV format is adopted to organize the OpenFlow protocol, which is shown in Table 4-5.

**Table 4. NextHop TLV.**

| TLV Type = NextHop (Fixed length) | TLV Length = 0 - N (Fixed length) | TLV Value = (0 - N) Nexthop Content-Address sub TLVs or Nexthop Content-Interface sub TLVs |
2.4 Using ACL and FIB hardware to implement FlowTable rapidly and low-costly

When implementing FlowTable in existing commercial devices, many mature hardware resources inside devices can be utilized to implement the base functions of FlowTable, such as ACL&QoS hardware tables and FIB hardware tables, etc. So FlowTable, an indispensable element in OpenFlow devices, is implemented in an indirect, low-cost and rapid way, which will promote the large-scale applications and popularization of OpenFlow.

The main functions of ACL&QoS hardware tables are to analyze some of the fields of the packets and perform the “Permit/Deny” actions or QoS actions, such as forwarding the corresponding packet with a certain fixed bandwidth. On the other hand, FIB hardware tables are mainly used to forward packets according to the destination IP address in packets. Therefore, when using the combination of ACL&QoS and FIB hardware to implement FlowTable, the FlowTable sender should encapsulate a FlowTable entry into one of the two different formats: ACL&QoS-type FlowTable or FIB-type FlowTable according to the actual application’s situation. The FlowTable receiver should issue the different types of FlowTable to different hardware tables. So two aspects of the devices need to be extended:

Firstly, the FlowTable needs to be organized and described by two kinds of TLVs. Different values of “Type” field in TLVs identify different types of FlowTable, such as ACL&QoS-type FlowTable and FIB-type FlowTable. Any type of FlowTable must contain at least three sub-TLVs: Header sub-TLV, Action sub-TLV, and Counter sub-TLV, as shown in Figure 4. But in different types of FlowTable, the specific content of these three sub-TLVs may be different. For example, in FIB-type FlowTable, its “Header sub-TLV” would only include the destination “IP address TLV” and its “Action sub-TLV” would only include “Forward action TLV”, as shown in Figure 5.

Secondly, the FlowTable receivers will analyze the “Type” field of the TLVs received, and distinguish ACL&QoS-type and FIB-type FlowTables according to the TLV type and will issue a corresponding item to ACL&QoS or to FIB hardware tables. In the Figure 6, there is no FlowTable hardware in devices, and OpenFlow Agent Module will execute the change from FlowTable TLV to ACL&QoS or FIB hardware resources.
3. OpenRouter: OpenFlow+’s implementation in real commercial devices

We have implemented OpenFlow+ in a commercial router (DCRS 5980/5950), which we call OpenRouter. There is no FlowTable designed and implemented in OpenRouter. FlowTable is replaced by ACL and FIB. Figure 7 is OpenRouter’s architecture.

Firstly, an OpenFlow Agent module is embedded into the control plane of a router whose main function is to run standard OpenFlow communication protocol.

Secondly, information encapsulated in OpenFlow protocol (including FlowTable) is redesigned and reconstructed with Type-Length-Value (TLV) formatting to support more user-defined information transport and easy extension for the existing information, such as FlowTable extension from IPv4 to IPv6.

Thirdly, FlowTable is implemented using existing hardware resources -- ACL&QoS and FIB in OpenRouter. The OpenFlow Agent module will convert a FlowTable entry it receives into a ACL&QoS table or FIB table according to its TLV type.

Fourthly, the OpenFlow Agent module sets up interfaces with Routing Table Management (RTM) module and sFlow module. On one hand, with the interface to RTM, routing change messages generated by routing protocols inside a router can be informed to a controller. On the other hand, a controller can send user-defined routing information generated by customized routing protocol to OpenRouter. With the interface to xFlow, sample packets can send to a controller. These information exchanges show the cooperation and collaboration of control logic inside/outside a device.

Fifthly, two asynchronous messages and a synchronous message are added in OpenFlow protocol to transmit network state information and sampling packets.

4. THE BENEFITS OF OPENFLOW+

The advantages of OpenFlow+ are as follows:

More openness for network devices. OpenFlow+ exposes not only FlowTable hardware but also ACL & QoS, FIB, and xFlow hardware to outside control logic for use. So the hardware of OpenFlow devices becomes more and more transparent and standard.

More efficient control for the network. OpenFlow+ integrates control functions inside/outside a router to realize more efficient and more advanced control functions for the network.

More flexible organization for data in OpenFlow Protocol. OpenFlow+ use TLV format to encode and encapsulate data information. TLV format can not only support the transmission of variable-length and optional data, but also help to achieve the goal of flexible extension of FlowTable structure in future.

More low-cost implementation for OpenFlow hardware. OpenFlow+ enables the use of existing ACL and FIB hardware for a low-cost and rapid implementation of FlowTable. Neither additional hardware nor hardware structure modification are needed for a commercial router in OpenFlow+.

5. TWO APPLICATIONS OF OPENFLOW+

We will show two applications of OpenFlow+ and OpenRouter to illustrate OpenFlow+’s powerful functions and advantages in an intra-AS network.

5.1 Intra-AS Source Address Validation (SAV)

In this application, an intra-AS network with 10 OpenRouters is set up, as shown in Figure 8. There is an intra-AS SAV[6] controller running as a new user-defined component in the NOX (OpenFlow Controller). In this experiment, factual routing information generated by routing protocols inside OpenRouter is reported to the intra-AS SAV controller by extended OpenFlow protocol as soon as the routing table changes. This is similar to the process when the device port state changes and is reported. With the real routing information, the intra-AS SAV
controller can draw out the right topology of the current AS and calculate the right forwarding paths for each IP prefix. Sample packets generated by the sFlow hardware module inside devices are also encapsulated and sent to the intra-AS SAV controller. If a sample packet is analyzed as a spoofing packet according to the forwarding path calculated previously, ACL filter rules will be issued to some OpenRouters to reject these kinds of packets. This application shows that it is necessary and significant to transport information generated by inside control logic to outside one in order to realize the cooperation of both control logics.

5.2 QoS routing in intra-AS

In this application, as shown in Figure 9, we will set up an intra-AS network with 3 OpenRouters. This experiment shows a user-defined QoS routing protocol, which is running in the NOX controller, how to implement assigning different bandwidths to different hosts. A benefit to the openness and available use of ACL&QoS tables for outside control logic is that QoS routing Controller can directly issue QoS entries to OpenRouters’ QoS tables rather than FlowTable. And a benefit to the centralized computing mode of OpenFlow which control the global information of an intra-AS, QoS routing Controller can assign different bandwidths for different hosts (on behalf of different users) to achieve QoS routing.

6. CONCLUSIONS AND FUTURE WORK

The Internet needs innovation. But we still don’t know exactly what functions and features the future Internet should include. So we think it is improper to build a concrete and fixed network for the future Internet now. “Giving him a fish is not better than teaching him how to fish”. OpenFlow is a novel thinking to reconstruct the Internet with many outstanding features, such as its openness and standardization. These features give the Internet more powerful abilities to reform and innovate. However we think there are still some limitations of OpenFlow.

In this paper, we discussed the possible aspects of OpenFlow which need to be extended: (1) More hardware resources, which are open, standardized and controllable, are added to devices to provide diversiform and powerful hardware functions; (2) Distributed and centralized computing mode are combined and integrated together to work to obtain both benefits of the two modes; (3) OpenFlow protocol format is extended to support the transmission of diverse and complex data; (4) A rapid and low-cost implementation method of OpenFlow FlowTable is introduced, in which ACL and FIB hardware is used to serve as FlowTable hardware. We believe that these extensions for OpenFlow will help to accelerate the business development of OpenFlow and promote its large-scale use and popularization.

For the future work, we think more hardware resources in devices should be exposed and standardized for use, and the methods for inside or outside control logics to use these open and standard hardware resources should be unified.

7. ACKNOWLEDGEMENTS

This work is supported by “Future Internet Research Cooperation Agreement between Tsinghua University and Stanford University”, National Science Foundation of China under Grant 61073172, and National Basic Research Program ("973") of China under Grant 2009CB320501.

8. REFERENCES


