OpenFlow-Based Software Defined Networks

Ankith Gunapal
10/24/2012

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment, that my paper contains no plagiarized material, and that I have made every effort to cite sources properly.

Student Signature Date

Abstract
The present day network architecture finds it challenging to meet the ever-growing needs of the end users and services [1]. With the collaborative effort of a large number of experts, software defined networking has been developed which addresses the challenges faced. This paper explains how OpenFlow-Based Software Defined Network architecture helps in building highly dynamic networks that satisfy the ever-growing need for fast, reliable and flexible network infrastructure.

I. Introduction

Software Defined Networking (SDN) is a new network architecture where there is separation between the data layer and the control layer [1]. The control layer is directly programmable and is present in a centralized location. This enables us to treat the entire network infrastructure as a single virtual entity, which can be controlled from a single point and used by the higher levels of abstraction for various kinds of services. OpenFlow is the first widely accepted interface between the control layer and the data layer. OpenFlow, along with a network operating system like NOX, facilitates centralized control of management applications instead of running distributed algorithms on individual network devices [3]. This architecture enables us to design highly dynamic networks.

II. The Need for a New Network Architecture

The current network technologies have a number of limitations. The network architecture we see today has evolved from a set of distinct protocols and algorithms running on individual network devices [1]. Complex protocols have evolved over time to meet the speed and infrastructure requirements of the industry. Thus, when a new device is to be added to a particular network topology, it requires a reconfiguration of a number of network devices (switches, routers) and applications (firewalls, authentication). This process can be very tedious and time consuming and could lead to some unforeseen configuration errors. Because of the complexity involved, the networks are designed to be relatively static to minimize any major disruption in the network

services. The static nature of the network implies that the network cannot vary dynamically according to the traffic and end user needs. On the other hand, an OpenFlow-Based SDN is so dynamic that one can have two different paths between any two nodes depending on the traffic [2]. The dynamic nature of the network results in more efficient use of available resources. Another major issue with the existing networks is the dependency on vendors. When an enterprise sets up a data center with network devices from a particular vendor, the enterprise is totally dependent on the vendor for any future upgrades to the infrastructure. Having an open interface would help the enterprises in coming up with better, faster solution on their own. Thus, their progress would not be dependent on the design cycle of the vendor [1].

III. Software-Defined Networking

This section gives an overview of the Software-Defined Networking (SDN) architecture. The subsequent sections explain the data layer and the control layer. Network intelligence resides in the SDN control layer [1]. The network devices are freed from running any protocols on their own. They just merely accept commands from the SDN controller, which can be programmed according to the user's requirements. Therefore, the entire infrastructure appears as a single virtual entity to the application layer. A number of Application Programming Interfaces (API) are supported that offer common network services like routing, switching and bandwidth management [6]. The SDN architecture is shown in Figure 1.The job of the network manager is simplified because he has to just program in one location instead of having to worry about multiple configurations in each network device within the concerned topology [1]. With SDN, the network manager can monitor the traffic and dynamically change the topology that generates the best throughput. Another advantage of this architecture is that it is vendor-independent. The

network manager can program the network using a common methodology irrespective of which vendor the network device belongs to. He can also implement additional functionality on his own without having to wait for the vendors to get a software update for their equipment.

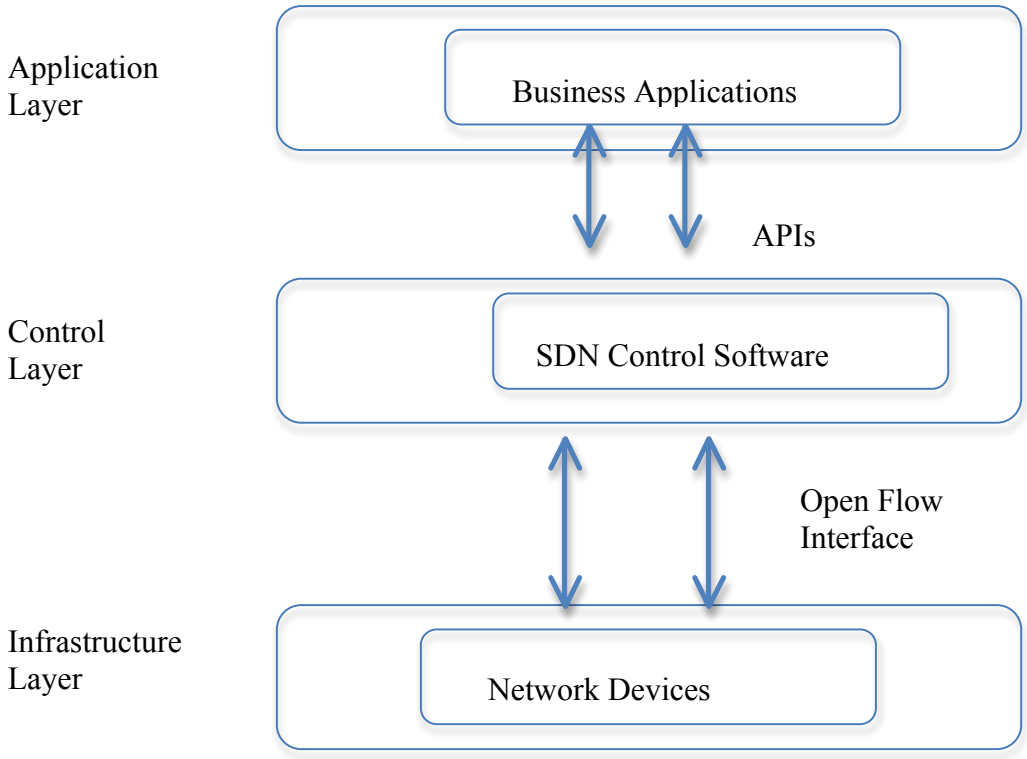

Figure 1: Software Defined Network Architecture [1].

IV. OpenFlow

OpenFlow is the first standard interface that has been defined for communication between the SDN Control Layer and the Infrastructure Layer as shown in Figure 1 [1]. OpenFlow allows access to the physical devices like switch, router and virtual devices (VLAN) from centralized control software. OpenFlow interface is implemented both at the infrastructure end and the control software end. OpenFlow uses the concept of flows to establish connection between the required points. It takes into consideration the current traffic, the required quality of service and

establishes a flow between the two points that can be pre-programmed or can be dynamically altered in real time. This provides a great deal of control over the network and the resources can be efficiently utilized. Since the traffic is sent across on a per flow basis, different paths may be utilized to transfer data between two points. This also implies that if a certain part of the network is being under-utilized, it could be configured to have less bandwidth and thus save power and bandwidth. A network device that supports OpenFlow is implemented such that it can support the conventional protocols as well. This implies that an OpenFlow based network device could be seamlessly integrated with the existing network topology and switched over to OpenFlow-Based SDN gradually [1].

V. OpenFlow Switch Architecture

OpenFlow switch is a part of the infrastructure layer in Figure 1 [1]. The concept of an OpenFlow switch was invented in Stanford University. Many of the professors and researchers wanted to experiment with their own networking protocols [2]. However, under the current network architecture they could not conduct these experiments on the Stanford network without disrupting the traffic. OpenFlow makes use of the flow tables in the switches and routers [4][5]. OpenFlow is implemented by programming these flow tables and dividing the traffic into the regular flow and the research flow. The research flow is programmed such that it only accesses the required network devices for the new protocols experiments [2]. This ensures that regular traffic is not disrupted.

An OpenFlow switch primarily consists of three parts 1) a flow table that specifies what action is to be taken for a particular flow entry, 2) a secure communication channel between the switch

and the SDN controller, 3) OpenFlow protocol, a standard way for the controller to communicate with the switch [2]. The basic concept of an OpenFlow switch is shown in Figure 2 [2].

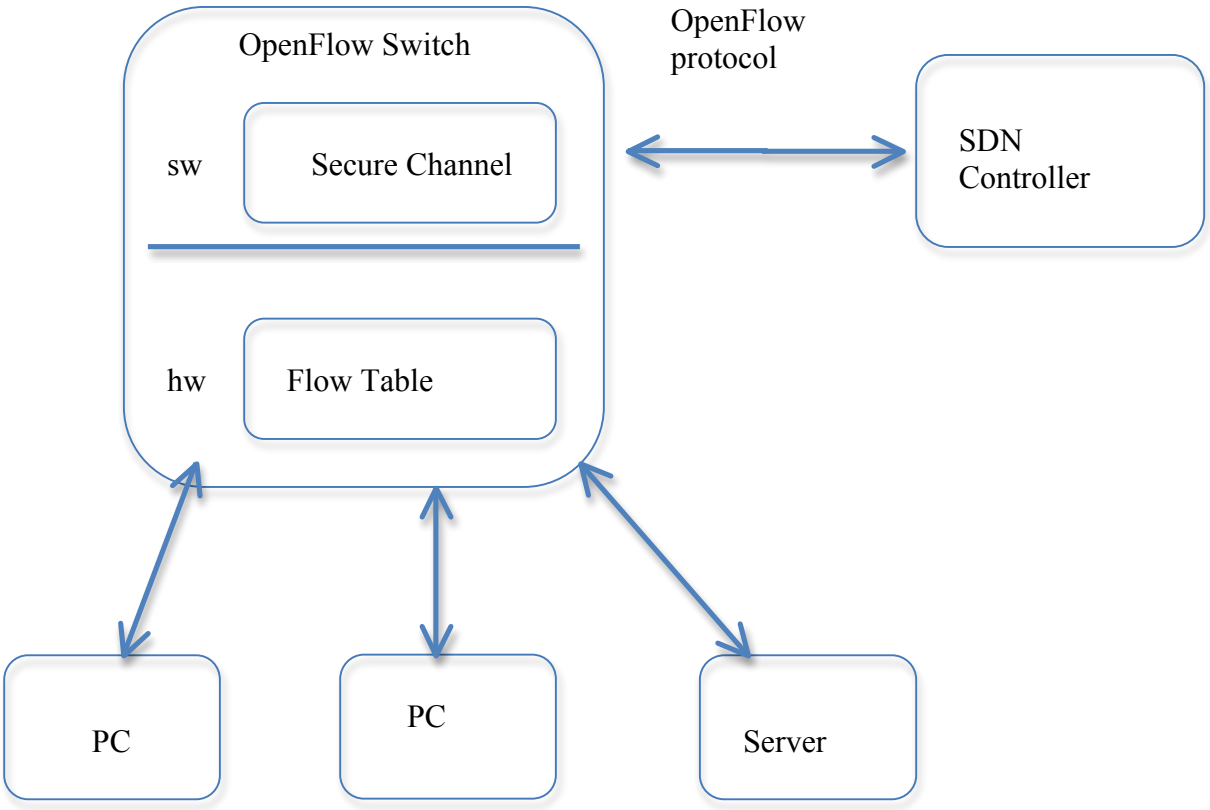

Figure 2: OpenFlow Switch [2].

The Flow Table has three basic components 1) a header that describes the kind of flow, 2) the action associated with the flow, 3) counters that indicate how many packets are transmitted and the frequency of the flow [2]. The Flow Table is constructed by re-using the existing Ternary Content-Addressable Memories (TCAMS) in the network devices [4][5]. The header in the first generation "Type 0" OpenFlow switch has the basic structure as show in Figure 3 [2].

In Port	VLAN ID	Ethernet			IP			TCP	
		Src	Dst	Type	Src	Dst	Type	Src	Dest

Figure 3: “Type 0” Open Flow Switch Header [2].

Each flow can have three different kinds of action. 1) Forward the packets to a particular destination port, 2) drop the packet for security issues, 3) forward the packets to the SDN controller for processing [2]. Flow entries are added or removed from the Flow Table using the SDN controller.

VI. NOX

NOX is one of the SDN control software that is being used with OpenFlow [3]. NOX is basically the Control Layer show in Figure 1 [1]. NOX is a centralized network operating system where the programs are written in high-level abstractions [3]. This greatly simplifies the task of writing programs. NOX does not manage the infrastructure directly but provides a programmable interface for high-level network and control applications to be written. The two primary components of NOX are the NOX Controller and the Network View. The Network View is a database that contains NOX’s network observations. The network observations include the mapping between the high-level abstractions used by the APIs and the low-level addresses of the network devices, the network topology of the switches and routers, the location of the various other network elements. The Network View allows the entire network to be viewed as a single virtual entity. The basic structure of NOX is shown in Figure 4 [3].

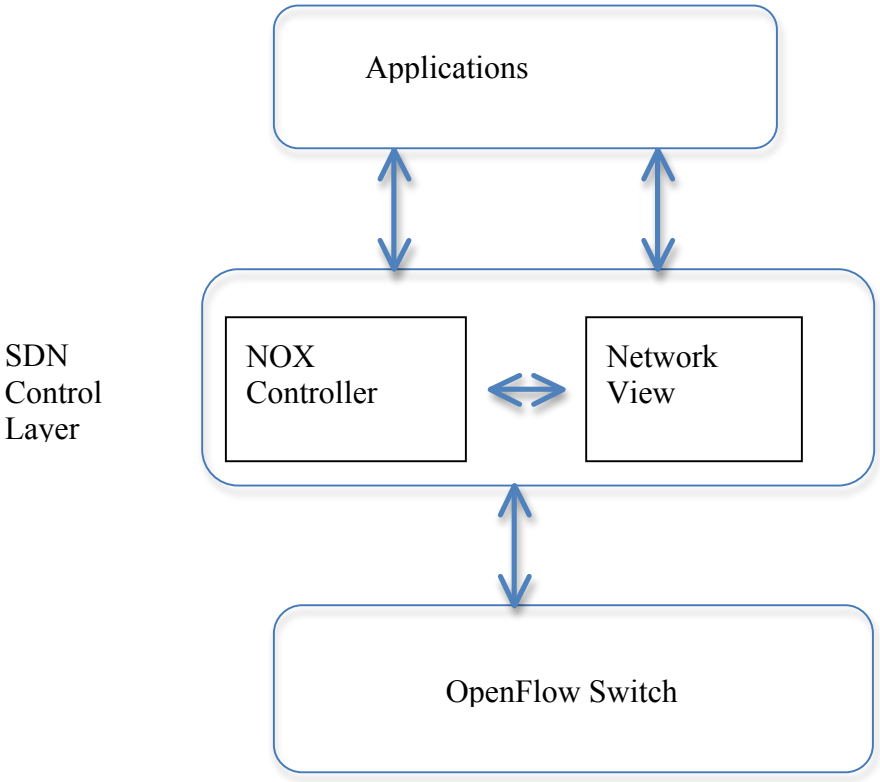

Figure 4: NOX Structure [3].

When an incoming packet arrives at an OpenFlow switch, the packet is looked up in the flow table [2]. If an entry exists, the action mentioned in the flow entry is taken. If there is no entry, the packet is forwarded to the NOX controller [3]. This happens with every first packet of a flow. This concept is used by NOX to form the Network View. The NOX Controller performs the decision-making for the packets. The NOX control software is a collection of NOX control processes running on a server. A particular application in conjunction with a control process can drop the packet or add a flow entry in the switch and forward the packet to the switch. The APIs (Figure 1) in NOX are built around events and the Network View. There are some basic applications in NOX that construct the Network View according to high-level names, keeping

the mapping with the low-level addresses and devices [3]. Other applications are written on top of this base application using the high-level names. These applications perform various network services like authenticating the addition of a new switch. Furthermore, when an event like adding a new switch occurs, a particular event handler is executed. This would specify the course of action to be taken. Lastly, the return value of the handler indicates whether to stop executing the current event or to execute the next logical step [3].

VII. Conclusion

An OpenFlow-Based SDN is able to transform a static network to a highly dynamic, virtualized network by separating the control layer from the data layer [1]. Current day business needs increase the demand for an easily reconfigurable, scalable network and OpenFlow-Based SDN provides a solution for these concerns. Supported by the Open Networking Foundation and OpenFlow Consortium, OpenFlow-Based SDN is on course to become the new standard for networks [1][2].

VIII. References

[1] Open Networking Foundation. “Software-Defined Networking: The New Norm for Networks”. Internet: https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf , Apr. 13 , 2012 [Oct. 15, 2012]

[2] Nick McKeown et al. “OpenFlow: Enabling Innovation in Campus Networks.” *ACM SIGCOMM Computer Communication Review*, vol. 38, issue 2, pp. 69-74, Apr. 2008

[3] Natasha Gude et al. “NOX: Towards an Operating System for Networks.” *ACM SIGCOMM Computer Communication Review*, vol. 38, issue 3, pp. 105-110, Jul. 2008

[4] K. Pagiamtzis and A. Sheikholeslami. “Content-addressable memory (CAM) circuits and architectures: A tutorial and survey,” *IEEE Journal of Solid-State Circuits*, vol. 41, issue 3, pp. 712–727, Mar. 2006

[5] Aaron Balchunas. “The Switching Tables”. Internet: http://www.routeralley.com/ra/docs/switching_tables.pdf , Feb. 2007 [Oct. 15, 2012]

[6] Jit Biswas et al. “The IEEE P1520 standards initiative for programmable network interfaces” *Communications Magazine, IEEE,* vol. 36, issue 10, pp 64-70, Oct. 1998