Volume: 08 Issue: 05 Pages: 3175-3180 (2017) ISSN: 0975-0290

Web-Based User Interface for the Floodlight SDN Controller

Hakan Akcay
Department of Computer Engineering, Istanbul University, Istanbul
Email: hknakcay@gmail.com
Derya Yiltas-Kaplan
Department of Computer Engineering, Istanbul University, Istanbul
Email : deryayiltas@gmail.com

ABSTRACT

Software Defined Networking (SDN) was born as a solution for next-generation network design. Due to its flexible architecture, SDN promises to make network devices simpler while giving better centralized control ability over network and improving parameters such as flexibility, resilience, reliability, and security. In this paper, we briefly introduce the SDN architecture and the Floodlight Controller that is one of the popular SDN controllers. We build a web-based user interface for the Floodlight Controller by using REST API. This application is the first program in the Floodlight SDN Controller literature to view the controller upon several properties such as device connections and flow tables.

Keywords – **Floodlight Controller, OpenFlow, Programmable Networks, SDN Web, Software-Defined Network.**

Date of Submission: March 14, 2017 Date of Acceptance: March 22, 2017

INTRODUCTION

The increase in the number of devices connected to the Internet, makes it very difficult to control the network. Problems such as configuration errors, lack of capacity of routing tables, security leaks are more common than ever before. Controlling the network by an administrator has become highly complex because of the inflexible behavior of the network components from switches and routers to firewalls, network address translators, load balancers, and intrusion detection systems.

In this context, Software Defined Networking (SDN) is a new way to design and manage networks [1]. SDN is a network programming framework that allows developers to program the network services with making it more intelligence and enhancing the performance of the network [1]. SDN is an agile, simple to implement and not costly architecture that decouples the data level from the control level [2].

SDN is being contributed by many prominent vendors like Cisco, Google, HP, Big Switch Networks, etc. [1].

The rest of this paper organized as follows: Section 2 introduces traditional networks. Section 3 describes SDN in more detail. Section 4 is about Floodlight Controller and finally Section 5 presents the conclusions.

1. TRADITIONAL NETWORKS

In the traditional networks, all network elements are defined by OSI (Open Systems Interconnection) model. This model has 7 layers that can have multiple protocols individually [3]. Each layer operates depending on its own lower layer and serves the upper layer. There are simple network devices on the lower layers of the OSI model while more complicated devices work on the upper layers. Therefore, operating the devices on the upper layers is more complicated than that on lower layers [3]. Switches are layer 2 elements and also the simplest devices in present networks. Routers are more complex, because they manage network traffic based on the forwarding decisions with using the routing tables which are constructed manually.

According to the traditional network approach, most of the network functionalities are implemented in a dedicated manner such as configuration of routers and switches, delivery of network applications using a hardware such as ASIC (Application Specific Integrated Circuit) [4]. Data flow is managed by routers and switches by using certain protocols.

We represent a traditional network architecture in Fig.1 by including the data, control, and management planes together.

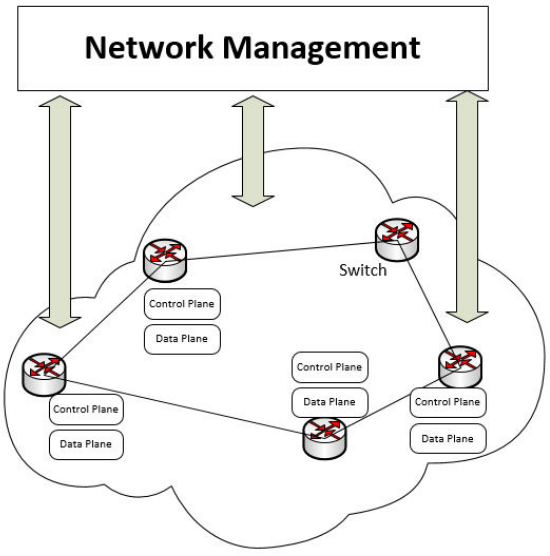

Fig. 1. Traditional network management

Volume: 08 Issue: 05 Pages: 3175-3180 (2017) ISSN: 0975-0290

The limitations of the traditional networks are becoming more important. In addition, they are not agile, slow to implement and takes too long to market [2].

In the traditional network approach, network administrators typically configure individual network devices by using the vendor based configuration interfaces. Network operators are responsible for configuring policies and implement many complex protocols to a wide range of network applications [5]. They have to apply these high-level policies into low-level configuration commands manually to configure network elements. They have very limited tools to perform these complex tasks [5].

However, internet applications and services become more complex and difficult to control day by day [5]. The idea of "Programmable Networks" emerged to facilitate network evolution. In particular, SDN is a new network design that promises to simplify network management and enable network innovation and evolution [5].

2. SOFTWARE-DEFINED NETWORKING

SDN is changing the way of design and management of the networks. SDN is a result of long work to make computer networks more programmable and flexible [6]. SDN has two important characteristics. First, SDN separates the control plane and the data plane of the network elements. Second, SDN centralizes the control plane with a single software called SDN Controller that manages the entire network and its components, such as OpenFlow switches, routers, and other middleboxes, via Application Programming Interfaces (API) [6]. An SDN Controller uses a protocol called OpenFlow to control switches and routers from the controller software. We represent a simplified view of the SDN architecture in Fig. 2.

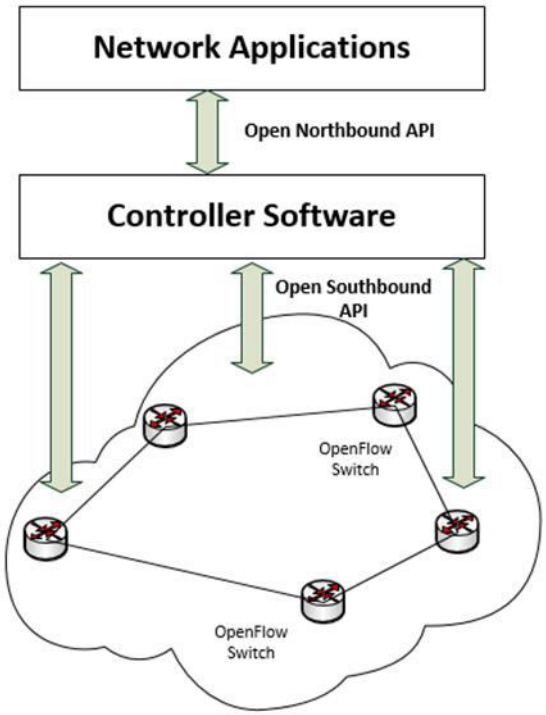

Fig. 2. Simplified view of the SDN architecture.

With the separation of control and data planes, network components return to the simple forwarding devices and the control of these elements is implemented in the logically centralized controller software [7]. The SDN Controller directly controls the forwarding elements via well-defined APIs. The mostly used one of these APIs is OpenFlow. An OpenFlow switch has one or more flow table inside and each table consists of some flow rules. Each rule of table matches a subset of the traffic actions such as dropping, forwarding, and modifying. The controller software has the ability of inserting, modifying, and deleting the rules of a flow table of switch. Therefore, an OpenFlow switch can be instructed by a controller and behaves like a router, firewall or performs other roles like load balancer [7].

2.1. SDN CONTROLLERS

SDN works on five fundamental traits: *plane separation, a simplified network device, centralized control, network automation, and openness* [8].

SDN Controllers fulfill the task of centralized control. The Controller is the main component of the network operation system (NOS) and the SDN networks. Controllers take the responsibility of establishing every flow in the network by modifying flow entries on the network devices [9]. Controllers must perform management of the network state and also the distribution of this state. A controller may involve a database to store information about network elements and the related softwares. A Controller software must provide a modern, commonly RESTful (Representational state transfer) API to an external application [10]. The communication between a controller and all network devices must be provided via a secure TCP control session. Additionally, a

Volume: 08 Issue: 05 Pages: 3175-3180 (2017) ISSN: 0975-0290

device and topology discovery mechanism and a service management system must be provided.

3. FLOODLIGHT CONTROLLER

Floodlight is Java-based, open source and one of the most popular SDN controllers supporting physical and virtual OpenFlow compatible switches. It has a number of packages denoted by *org.openlow* [8]. Floodlight is based on Bacon controller from Standford University [10].

The Floodlight architecture is modular with including device management module, topology module, learning switch, load balancer, Web Graphical User Interface (Web GUI) for web access, counter module for statistics. Floodlight controller presents a RESTful API allowing some applications to learn and get the state of the controller and network [10]. It uses LLDP protocol to discover network topology. Floodlight Provider module called as core module, handles I/O from network devices and turns incoming OpenFlow messages into Floodlight events.

Floodlight controller uses event listeners for receive notifications. Most important listeners are: *SwitchListener, DeviceListener, MessageListener*.

SwitchListener is used to receive notifications whenever a switch is connected or disconnected to the internet or has a change in its port status. DeviceListener is notified whenever a device (generally an end-user node) has been added, removed, moved or has changed its IP address or VLAN address [8].

MessageListener gets notifications whenever a packet has been received by the controller. When a packet is received, the application processes the packet and takes the appropriate action [8].

Floodlight controller provides both reactive and proactive applications: Java APIs for reactive and RESTful APIs for proactive application style. Proactive Floodlight applications can use the RESTful APIs to get information about the network state. Floodlight RESTful API uses the Restlet framework and includes a small web server inside that allows external applications to communicate with the SDN controller. Floodlight Controller Structure is shown in Fig.3.

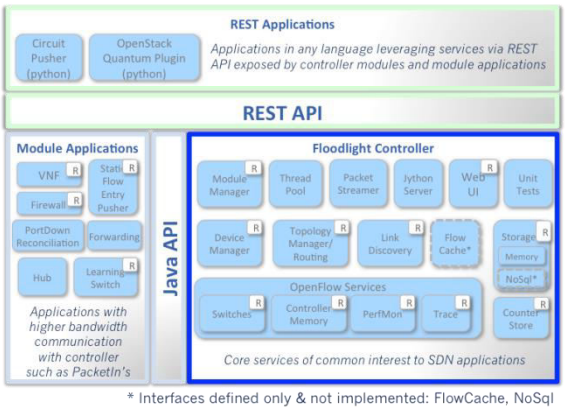

Fig. 3. Floodlight controller structure [11]

3.1. FLOODLIGHT REST API

In the early days of commercial Internet, setting configuration parameters of devices has been possible only using methods such as CLI, TL1, NETCONF, SNMP, TR-069 [10]. These mechanisms are available now and have been rarely used with SDN. But they are slow and difficult to maintain [10]. Furthermore, they are not suitable for today's data centers because that the data centers are required dynamic, frequent, and automated management tasks.

In the last few years, newer methods have been developed to make remote configuration changes on the network devices. The most popular and common one is the REST API. REST has been developed to make API calls across networks and uses Hypertext Transfer Protocol (HTTP) which is commonly used to pass the web traffic [10]. They are simple and extensible and use a standard TCP port and thus do not require any special configuration to allow API calls to pass through firewalls [10]. SDN Controller northbound API is shown in Fig. 4.

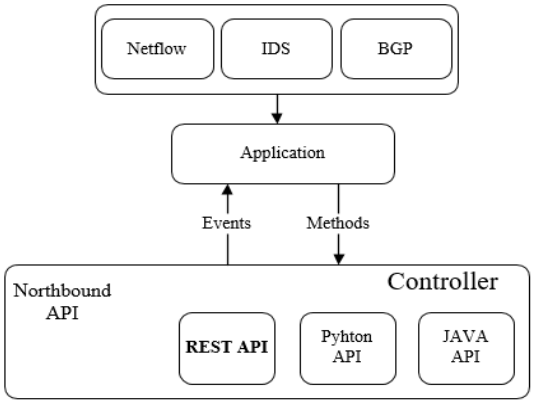

Fig. 4. SDN controller northbound API

The web-based REST mechanism is based on some methods such as HTTP GET, PUT, POST, and DELETE. It is very easy to make these REST calls secure by using HTTPS (Secure HTTP) protocol instead of HTTP. The Floodlight Web GUI is built on the REST API.

Volume: 08 Issue: 05 Pages: 3175-3180 (2017) ISSN: 0975-0290

3.2. FLOODLIGHT WEB GUI

In this section, we mention our Floodlight Web GUI in detail. This GUI provides a way for users to view the controller's state information, to connect switches via inter-switch links and the hosts to the network, to monitor the flow tables of the switches and the network topology [11]. Most of the statistics can be queried and displayed in an easy-to-read and tabular fashion by using this web GUI. Additionally, several modules of the Floodlight Controller can be exposed to the end users via this web GUI. For example, the Static Flow Pusher module has this GUI to insert the flows easily [11].

We develop the Floodlight Web GUI's home page as the controller's dashboard as in Fig. 5. Here, network administrators can monitor the status, uptime, selected role of the controller, connected switches count, connected hosts count, and the links between the switches.

Also, the details about the controller's memory consumption, the storage tables, and all loaded modules can be monitored. All of these parts of the GUI uses the REST API calls of the Floodlight Controller.

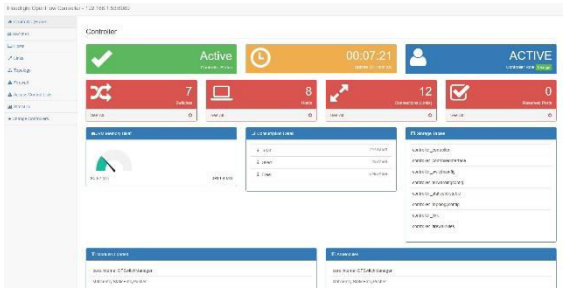

Fig. 5. Home page of the controller

One of the most important module of the GUI is the switches module that gives some detailed information of the connected switches such as flow count, packet count, buffer size, flow table count, vendor, hardware, and software version (see Fig. 6). Additionally, the administrators can change the role of a switch to Master, Slave or Equal by using this module.

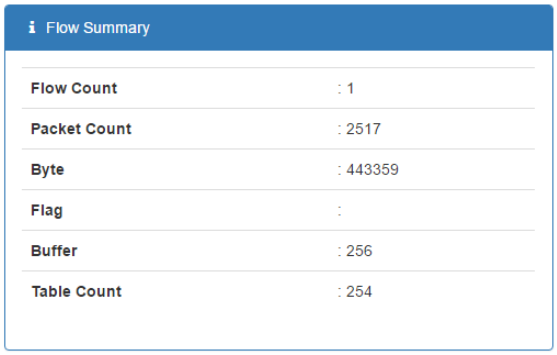

Fig. 6. Flow summary table of a switch

It is possible to add static flow entries to the flow table of a switch (see Fig. 7) and to monitor the port table of this switch.

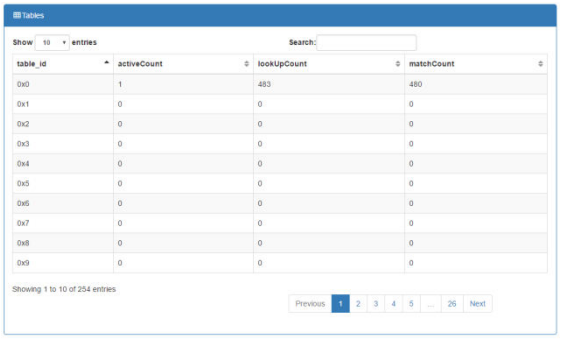

Fig.7. Flow table of a switch

Web GUI presents a list of the hosts connected to the network with the information of the MAC address of a host, IPV4 address, IPV6 address, MAC address of the connected switch, connected port of the switch, and last seen time.

All links between the switches can be monitored via GUI. Properties such as the direction of links, source switch and its port, destination switch and its port, and type of the link (internal, external) are the information provided.

Floodlight controller comes with a firewall module. A network administrator can change the status of the firewall, can enable/disable the firewall and change subnet mask of the firewall by using this module as in Fig. 8. Also, new firewall rules can be added or deleted.

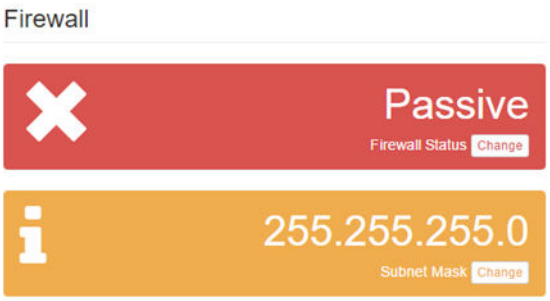

Fig. 8. Firewall module of GUI

While adding a firewall rule, the parameters that must be provided are switch id, source port, source mac address, destination mac address, source IP address, destination IP address, protocol type (TCP, UDP or ICMP), priority, and action (ALLOW, DENY) [11].

Floodlight contains an application that enforces Access Control Lists (ACL) in a reactive way. It is a firewall application that monitors Packet-in messages and then pushes appropriate flow entries [11]. In a proactive way, without being requested by the switch, the controller enforces ACL rules to switches too. Thus, the controller prevents additional delays [11].

Volume: 08 Issue: 05 Pages: 3175-3180 (2017) ISSN: 0975-0290

Fig. 9 represents the record insertion to the ACL and Table 1 gives the properties of an ACL rule.

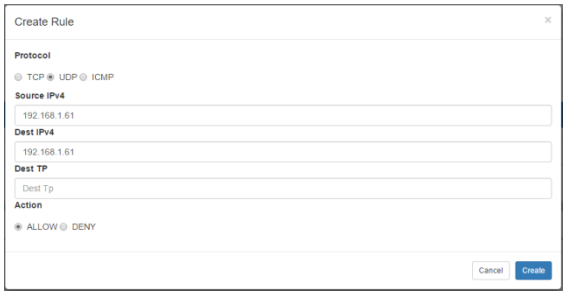

Fig.9. Adding a record to the ACL

Table 1. Properties of an ACL rule

Key	Value	Notes
nw-proto	string	"TCP" or "UDP" or "ICMP" (ignoring case)
src-ip	IPv4 address[/mask]	Either src-ip or dst-ip must be specified.
dst-ip	IPv4 address[/mask]	Either src-ip or dst-ip must be specified.
tp-dst	number	Valid when nw-proto == "TCP" or "UDP".
action	string	"DENY" or "ALLOW" (ignoring case), set to "DENY" if not specified.

Topology discovering module of the Floodlight uses LLDP protocol. And GUI calls the related RESTful API method of this module. Topology view of network is based on DIRECT and TUNNEL links discovered based on LLDP packets [11]. Our sample SDN topology built in the GUI is represented in Fig. 10.

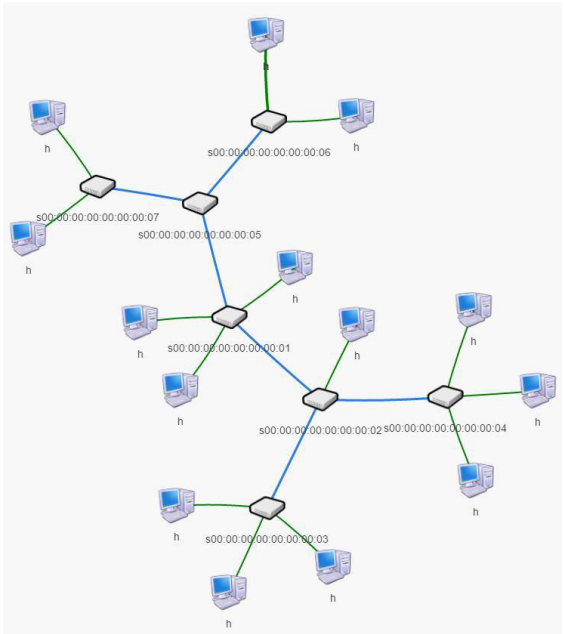

Fig.10. SDN topology view in the GUI

4. CONCLUSION

The traditional network architecture cannot fulfill today's network requirements efficiently and the popularity of SDN is increasing day by day.

SDN is the new network paradigm and can be easily implemented to the classical networks. The Floodlight controller and GUI provide many useful tools and a programmable network framework. SDN is just a new tool for developing new applications to solve network-management problems more easily. To support more widely ranged network services, SDN applications would require much more complex methods to analyze and control the network traffic, and programmable hardware.

Before developing an SDN application, the developer must decide between two general styles of SDN applications: reactive or proactive. In case of the reactive SDN applications, the output of the communication between the switch and the controller is usually a new flow entry in the switch's flow table. Reactive applications can program multiple switches parallelly. On the other hand, the proactive SDN applications require less communication from switch to controller. The proactive SDN applications program switches connected to the network with the flow entries that are appropriate to control and manage the incoming traffic before it arrives at the switch.

The reactive type is more useful when the connectivity to the controller is lost but less additional latency is provided with the proactive programming.

REFERENCES

[1] A. Shahid, J. Fiaidhi and S. Mohammed, Implementing Innovative Routing Using Software

Volume: 08 Issue: 05 Pages: 3175-3180 (2017) ISSN: 0975-0290

Defined Networking (SDN), *International Journal of Multimedia and Ubiquitous Engineering, 11*(2), 2016, 159-172.

[2] G. Wiley, *Software networks, virtualization, sdn, 5g and security* (Great Britain: ISTE Ltd and John Wiley & Sons, 2015).

[3] K. Ahokas, Software-defined networking, *Aalto University CSE-E4430 Methods and Tools for Network Systems*, Finland, Autumn 2014.

[4] Pooja, M. Sood, SDN and Mininet: Some Basic Concepts, *Int. J. Advanced Networking and Applications, 07*(02), 2015, 2690-2693.

[5] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka and T. Turletti, A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks, *IEEE Communications Surveys & Tutorials*, *16*(3), 1617-1634.

[6] N. Feamster, J. Rexford, and E. Zegura, The road to SDN: an intellectual history of programmable networks, *SIGCOMM Comput. Commun*, *44*(2), 2014, 87-98.

[7] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, Software-Defined Networking: A Comprehensive Survey, *in Proceedings of the IEEE*, *103*(1), 2015, 14-76.

[8] P. Göransson, C. Black, *Software defined networks* (USA: Morgan Kaufmann, 2014).

[9] Y. Jarraya, T. Madi and M. Debbabi, A Survey and a Layered Taxonomy of Software-Defined Networking, *IEEE Communications Surveys & Tutorials, 16*(4), 2014, 1955-1980.

[10] T. D. Nadeau, K. Gray, *SDN: software defined networks* (USA: O'Reilly, 2013).

[11] R. Izard (administrator), H. Akcay (GUI developer), (2017). Floodlight WEB GUI. [online] Floodlight.atlassian.net. Available at: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/40403023/Web+GUI [Accessed 17 Mar. 2017].