Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 9, September 2014, pg.397 – 403

RESEARCH ARTICLE

Orchestration Of Software-Defined Application Delivery Networks using Open-Flow Architecture

Raghuram.P[1], G. Praveen Babu[2]
[1]M.Tech, Computer Networks Information Security, School of IT, JNTU
[2]Associate Professor, School of IT, JNTU
[1] raghuram369@gmail.com; [2] pravbob@jntuh.ac.in

Abstract: Computer networks are complex and difficult to manage. They involve many kinds of equipment, from routers and switches to middle-boxes such as firewalls, network address translators, server load balancers, and intrusion-detection systems. Creating an application delivery network using such complex and distributed networking approach is very difficult and is often error prone. In this paper we discuss an innovative approach to create application delivery network using software defined networking. With the help of centralized control plane software defined networking (SDN) simplifies the task of creating Switches, routers, firewalls and load balancers etc. and helps in orchestrating the application delivery in a very short time and with least amount of resources.

Keywords: SDN, open flow, Firewall, Switches, Routers, Simulation, Automation

I. Introduction

A network organizing technique that has come to recent prominence is the Software-Defined Network (SDN) [1]. In essence, an SDN separates the data and control functions of networking devices, such as routers, packet switches, and LAN switches, with a well-defined Application Programming Interface (API) between the two. In contrast, in most large enterprise networks, routers and other network devices encompass both data and control functions, making it difficult to adjust the network infrastructure and operation to large-scale addition of end systems, virtual machines, and virtual networks. In this article we examine the characteristics of an SDN, and then describe the OpenFlow specification, which is becoming the standard way of implementing an SDN. Software-defined networking (SDN) is an approach to computer networking that allows network administrators to manage network services through abstraction of lower level functionality. This is done by decoupling the system that makes decisions about where traffic is sent (the control plane) from the underlying systems that forward traffic to the selected destination (the data plane). The inventors and vendors of these systems claim that this simplifies networking. Traditional application delivery networks consist of costly equipment like routers, firewalls, load balancers etc. which are quite difficult to manage and are a nightmare when it comes to the implementation of a change or troubleshooting. We try to solve this using the disruptive SDN technology which converts any open flow enable device into application delivery equipment on demand basis. This there by reduces the CAPex and OPex of application delivery alongside easing the process.

II. SDN Architecture

The SDN architecture departs from legacy solutions by building networks from three abstractions or layers. First, the infrastructure layer acts as the foundation for an SDN architecture. The infrastructure consists of both physical and virtual network devices such as switches and routers. These devices implement the OpenFlow protocol as a standards-based method of implementing traffic forwarding rules. Second, the control layer consists of a centralized control plane for the entire network. The control plane is decoupled from the underlying infrastructure to provide a single centralized view of the entire network. The control layer utilizes OpenFlow to communicate with the infrastructure layer.
Third, the application layer consists of network services, orchestration tools, and business applications that interact with the control layer. These applications leverage open interfaces to communicate with the control layer and the network state.

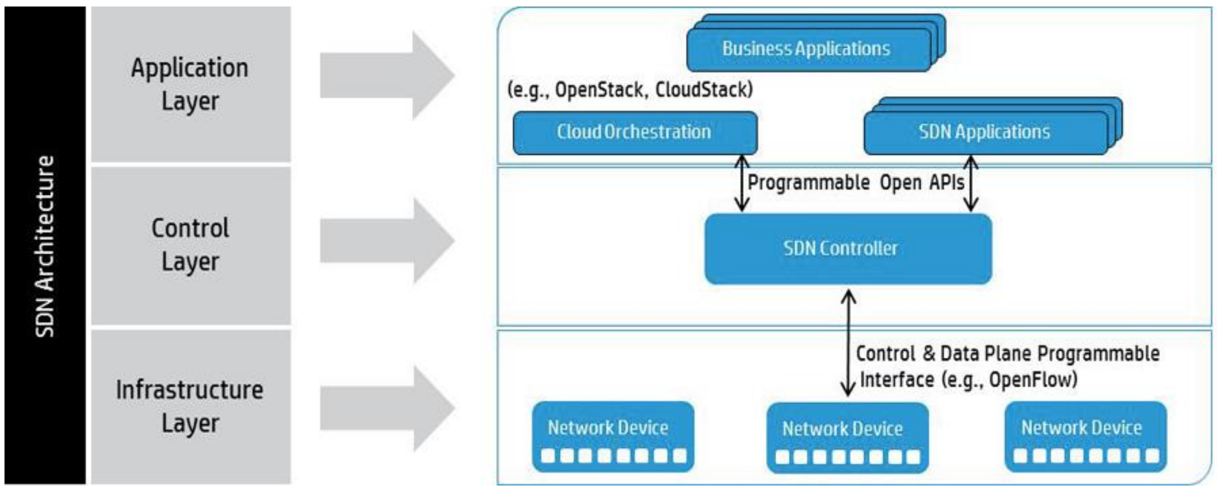


Fig. 1 SDN architecture

III. SDN enabled by OpenFlow

At the foundation of enabling SDN is an emerging open standard called OpenFlow, which ultimately allows the network to be more responsive to business needs.

OpenFlow hides the complexity of the individual pieces of network devices. OpenFlow centralizes the control of those devices in a virtualized manner, simplifying network management for network managers.
The OpenFlow protocol uses a standardized instruction set, which means that any OpenFlow controller can send a common set of instructions to any OpenFlow-enabled switch, regardless of vendor.

OpenFlow is an open-standards way of virtualizing the network. Network managers can specify different policy rules for different groups of devices and users, which create multiple virtualized networks regardless of the physical network connections. This allows network managers to easily customize and manage these virtualized networks to ensure proper policies such as forwarding path, QoS and security.

OpenFlow is designed to be programmable. The OpenFlow instruction set allows network managers to try new ideas or create new protocols to solve problems specific to their organization`s network needs. This allows network architects to experiment with new services and protocols on a real-world network that cannot be simulated in a test lab—or is too risky to undertake in a production network today.

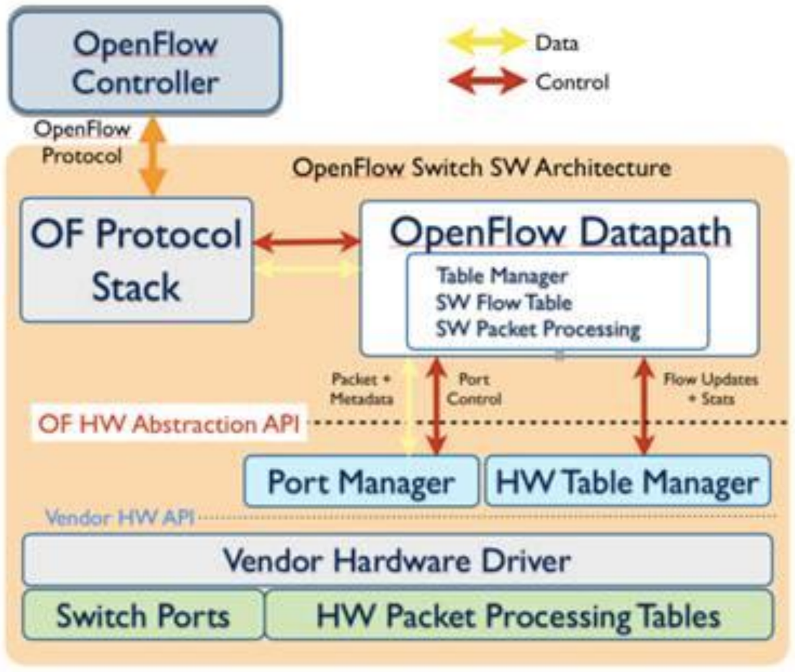


Fig. 2 Openflow protocol

IV. Application delivery blocks

1. <u>Monitoring:</u>

Dashboard is a tool which helps the network administrators and architects to have a real-time health monitoring of controller resources, control, and data channel utilizations, with self-healing capabilities. This is done with the help of open-flow counters. Each open flow enabled device expose specific set of counters which collect the statistics of the device and the network resources. Counters are maintained for each flow table, flow entry, port, queue, group, group bucket, meter and meter band. OpenFlow-compliant counters may be implemented in software and maintained by polling hardware counters with more limited ranges. Duration refers to the amount of time the flow entry, a port, a group, a queue or a meter has been installed in the switch, and must be tracked with second precision. Amongst the available counters in a switch few of the essential fields are used for gathering the appropriate details.

TABLE I

Counters	**Bits**
Reference Count (active entries)	32
Packet Lookups	64

Counters	Bits
Packet Matches	64
Received Packets	64
Received Bytes	64
Duration (seconds)	32
Transmitted Bytes	64
Receive Drops	64
Transmit Drops	64
Receive Errors	64
Transmit Errors	64

Based on the user defined network, aggregate stats are constructed and the result is visually displayed to user using a web interface.

Auto-heal capabilities are employed in the data fabric monitoring. This is achieved by continuously monitoring the receive drops, Transmit drops, receive errors and transmit errors against the set of parameters collected from the user through web interface. Once the threshold is reached, the port of the switch on which the frame rate errors are occurring is restarted

2. Design:

Design module helps the users to identify the whole network topology at once. This is possible with the help of a state table which is constructed from the network map create by the open flow commands. Since the controller communicates with all the open flow enabled devices it can have a map of the entire network. Making use of the counters as discussed in the earlier sections, it is also possible to generate a visual table depicting all the flows which are existing in that particular switch. This is collected with the help of the controller exposing the rest API.

The second phase of design module is called canvas. This helps users in creating vision type of network diagrams. The advantage here is that users can simulate this in a virtual environment. This helps users in eliminating any kind of errors are design mistakes before the implementation/production phase. HTML5 technology helps us in creating such a canvas.

This is achieved using Mininet which creates a realistic virtual network, running real kernel, switch and application code, on a single machine (VM, cloud or native), in seconds, with a single command. Mininet allows creation of multiple nodes(hosts/switches/controllers) which allows a big network to be simulated on a single PC. This is very useful in experimenting with various topologies and different controllers. Mininet makes use of open virtual switches to create a network schematic in virtual environment. The canvas application converts the vision style diagrams into a python code and then can be implemented using the mininet to create custom topologies.

```
sudo mn —topo=mytopo —custom ~/demo.py —controller=remote,ip=<controller ip>
```

The above command creates a custom network topology from the python file which is created from the network map. This simulated environment is useful for testing any kind of changes or implementations prior to implementation.

3. Control:

Openflow Action fields help us in dynamically converting any open flow enabled switch into a fire-wall module upon requirement. The policy base keeps the track of all the policies which user requires to be implemented in their network. The policy base lets users to fetch data from external resources like excel sheets, change request documents, etc. Rule base module keeps track of all the existing rules running live on the network. The policies which are extracted from the user input are constructed into an open flow command are passed into the open flow module of the switch with the help of SDN controller.

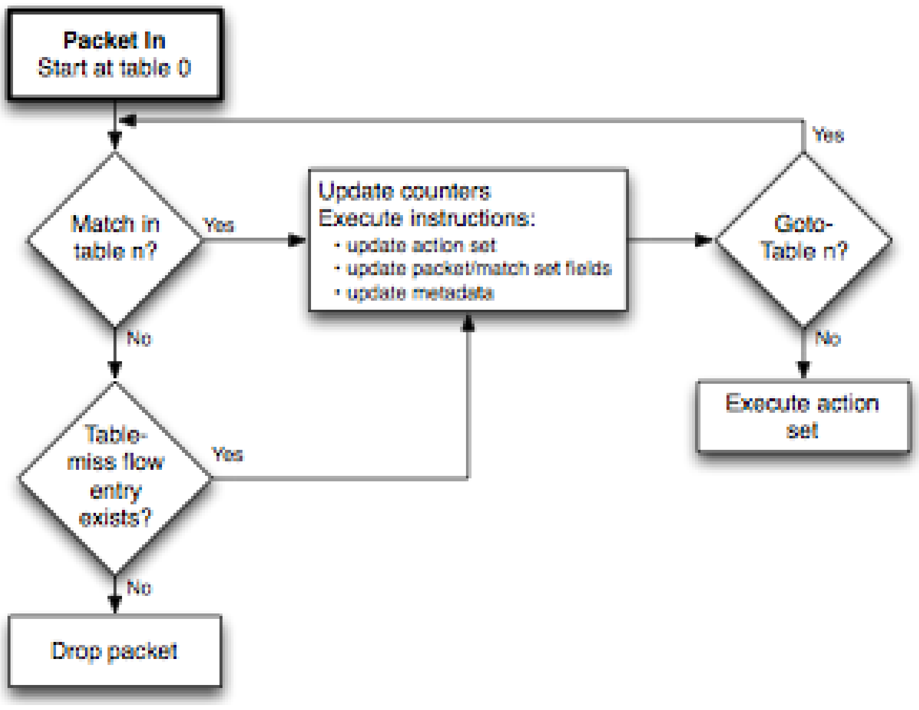


Fig. 3 Control application process flow

We make use of Open flow's match-action characteristics for creating dynamic application delivery devices. Creating a flow entry into the switched does this. The flow entry consists of the match field and action field. Match field guides the switch to match the incoming packet`s header information with the predefined values and once a match is found take the action as specified in the flow entry. In our case the flow entry would be either to allow the traffic normally or block the traffic as decided by the user.

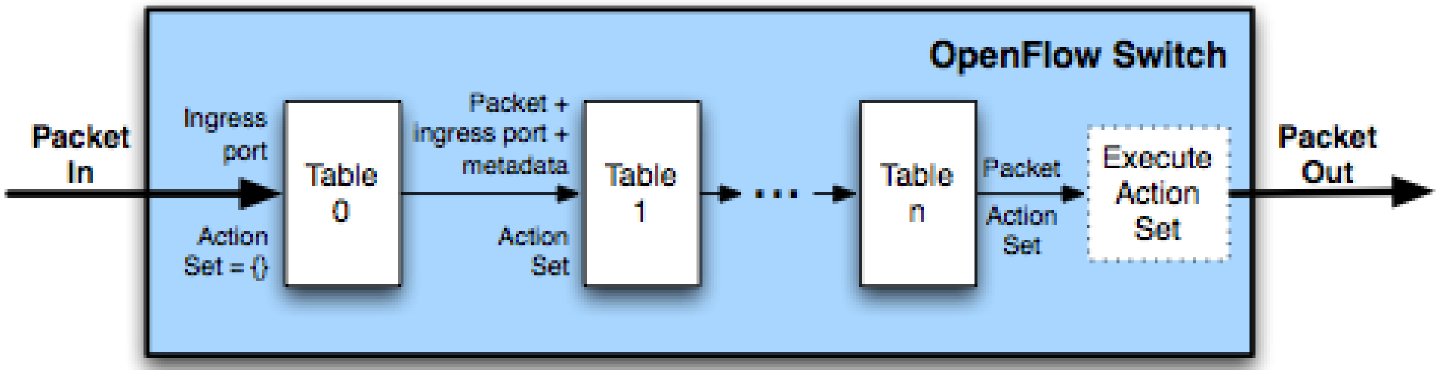


Fig. 4 Control module action

a) User creates a policy using the graphical input form

b) Policy is converted into a open flow command

c) The open flow command is passed to the open flow enabled switch using the controller

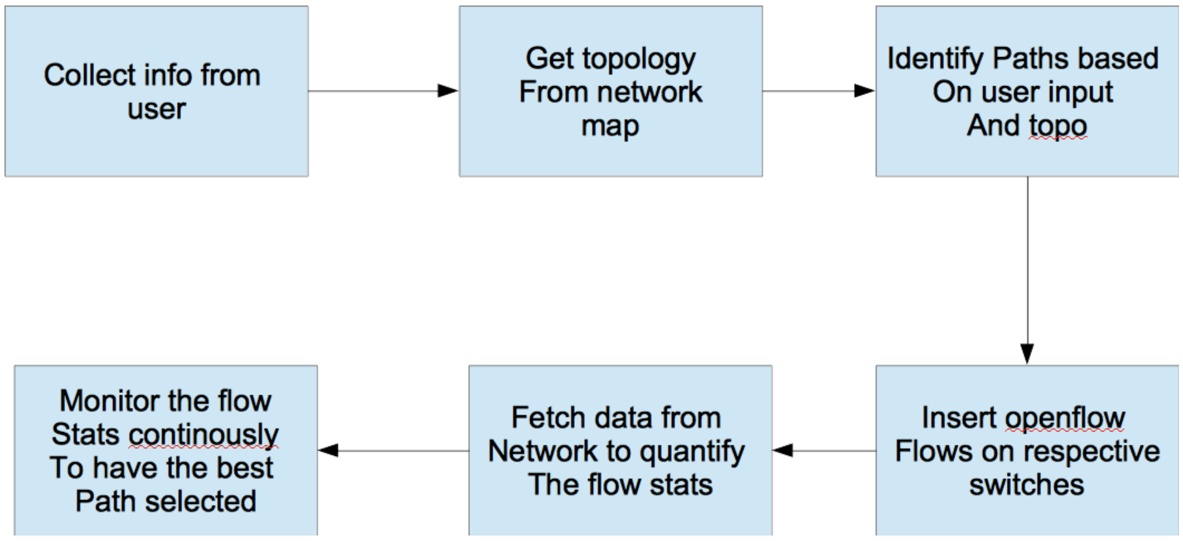
d) Once the switch receives a packet is parses through the header or the packet data. Compares it with the flow match parameters. Once a match is found the switch then takes a predefined action.

e) The action parameters would be either to drop the packets or forward it as usual. This is decided by the user

Another Important of control is the simulation. In traditional networking environment any firewall change or policy change has to be implemented directly on the production network. This is often pretty risky as a single erroneous change can bring down whole network. With the help of mininet as discussed in the design module can be used to created virtual network. We take a snapshot of the live production network and convert it into a state table with the help of controller. The state table is then converted into a python program, which is again implemented using mininet. This helps users in testing all the changes on virtual network before implementing it on the production network.

4. <u>Qos:</u>

Quality of service plays a key role when it comes to network resource management. open flow stats field gives out aggregate statistics of a switch, port or a flow. This information can be hence used to predict the network performance. In our application we read these statistics and dynamically compare them with the predefined threshold values. Any time the aggregate stats exceed certain amount we can change the direction of flow or in our case change the respective flows in individual switches automatically.


Collect info from user
Get topology From network map
Identify Paths based On user input And topo
Insert openflow Flows on respective switches
Fetch data from Network to quantify The flow stats
Monitor the flow Stats continously To have the best Path selected

V. Conclusion

SDN may sound abstract, but the technology is on target to provide concrete advantages to network infrastructure from increased global connectivity to better content delivery.

The concept of the software-defined network (SDN) initially caused some confusion in IT, but the mere outstanding advantages it offers are really placing it amongst the top networking technologies Similar to network virtualization, SDN allows for the direct network-related abstraction of services. This abstraction of services can be accomplished on both logical as well as physical infrastructures, but is not actually defined by specific physical devices or logical components. Software-defined technologies are already creating new ways to abstract resources at the enterprise level, and products are coming onto the market that allows IT departments to benefit from SDN. SDN orchestration techniques discussed above leverage the power of programmable networks to facilitate all the networking functionalities at the fingertips of networking administrators and architects.

REFERENCES

[1] http://tools.ietf.org/html/rfc7149
[2] https://www.cs.uaf.edu/2011/spring/cs641/proj1/dpkline/
[3] http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA3-8562ENW.pdf
[4] http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_16-1/161_sdn.html
[5] http://pronetworkingh17007.external.hp.com/nz/en/solutions/technology/openflow/index.aspx