A Comprehensive Tutorial on Software Defined Network: The Driving Force for the Future Internet Technology

Conference Paper · August 2016

DOI: 10.1145/2979779.2983928

CITATIONS
29

READS
3,674

5 authors, including:

Kshira Sagar Sahoo
National Institute of Technology Rourkela
54 PUBLICATIONS **291** CITATIONS
SEE PROFILE

Sagarika Mohanty
National Institute of Technology Rourkela
10 PUBLICATIONS **55** CITATIONS
SEE PROFILE

Mayank Tiwary
University of British Columbia - Vancouver
39 PUBLICATIONS **247** CITATIONS
SEE PROFILE

Brojo Kishore Mishra
GIET University
150 PUBLICATIONS **262** CITATIONS
SEE PROFILE

A Comprehensive Tutorial on Software Defined Network: The Driving Force for the Future Internet Technology

Kshira Sagar Sahoo
National Institute of Technology,Rourkela
India,769008
kshirasagar12@gmail.com

Sagarika Mohanty
IGIT, Sarang
Odisha
India,759146
sagarikam_23@yahoo.com

Mayank Tiwary
C.V. Raman College of Engineering, Bhubaneswar
India,752054
mayanktiwari09@gmail.com

Brojo Kishore Mishra
C.V. Raman College of Engineering
India,752054
brojokishoremishra@gmail.com

Bibhudatta Sahoo
National Institute of Technology,Rourkela
India, 769008
bibhudatta.sahoo@gmail.com

ABSTRACT

These days the usage of network is growing at a faster pace, at the same time a lot of challenges is facing by the network administrator, to tackle the frequent network access by the users. The network infrastructure is growing rapidly to meet the business need, but it requires re-policing and reconfiguration of the network. But managing the underlying infrastructure becomes more complicated to handle the unprecedented network demand. The Software Defined Network (SDN), is the next generation Internet technology, which not only solves the ossification of the Internet, but also creates innovations and simplifies the network management. The key idea behind SDN is separation of control plane from the data plane, as a result, devices in the data plane simple becomes the forwarding device and transfer all the decision-making activities in a centralized system called a controller. Among many, OpenFlow is the standard and most popular SDN protocol that interacts between controller and forwarding devices. In this article, we will give an overview of the basic architecture of SDN and OpenFlow, SDN-controller interaction and benefits of SDN.

CCS Concepts

•Computer systems organization → Distributed architectures; •Networks → Network Architectures ;

Keywords

Software Defined Network; OpenFlow; controller; flow-table;virtualization;

AICTC '16, August 12-13, 2016, Bikaner, India

DOI: http://dx.doi.org/10.1145/2979779.2983928

1. INTRODUCTION

In a traditional network packet forwarding, managing hardware devices, monitor the data flow are managed by a single hardware device. Conventionally, when a packet received by the routing device called a router, it applies a set of rules to get the shortest path and the desired destination for the packet. Generally, the data packets which are meant for the same destination follow the same procedure. This operation is carried out by most of the inexpensive routing devices. Some expensive devices can treat different packet according to the priority level of the packet. Some router vendors provide the flexibility to manage the queue size such that congestion control and traffic control can manage in an efficient way. But the underlying devices poses a low performance for the unpredictable traffic demand. Another major limitation of the traditional IP-based network is that the underlying hardware requires a huge expense, in case any new adoption. In other words, it is difficult to reprogram.
This has been made possible by implementing all the routing rules in a centralized software module rather than embedding in a hardware device, so that the administrator has more control over the network traffic hence network performance can be greatly improved. The above principle called Software Defined Network (SDN). The goal of SDN is to decouple the control and data plane. All the routing decision and controlling mechanism installed in a device called controller. The controller sends command to the data plane which consists of router and switch to manage the data packets. Control plane may consist of one or more than one controller in a network depending upon the size and usage of the network [1]. Initially, it was developed and tested as a part of the Ethane project at Stanford University. But later it became more popular when VMware acquired the brainchild of the SDN developer Nicira network. Gradually many open source products developed their framework to allow developers to create and test various SDN products. Open-Daylight, Cisco like most big organizations started their research wing for SDN [3].
The main objective of this paper is to give an overall idea about SDN/OpenFlow technology. Authors in [17] describe the concept of SDN in a very interesting way by asking a

various question to the reader. Similar to this author in [11] present an OpenFlow survey on 2-layer SDN architecture. In [6], Jarraya et al. highlighted numerous research issues related to the different layers of SDN. As compared to the above survey articles, this paper mostly emphasized on SDN/OpenFlow interaction mechanism between the devices and the controller along with the other integral part of the SDN.

After going through various research papers, magazines and visit miscellaneous blogs we prepare this article that will build a clear picture about SDN related technology to the reader. The main contributions of this paper are: we provide a complete tutorial on SDN related technologies like SDN /OpenFlow architecture, controller and switch interaction, benefits of SDN etc. In the next section of this article, the limitation of current network technology has been addressed, then explored how these limitations are overcome by SDN technology. After discussing the basic principle on SDN, we have briefly discussed on OpenFlow architecture in section III. We end this article by discussing the benefit of SDN usage and standardization effort towards SDN in the conclusion part.

2. ROUTING MECHANISM IN LEGACY NETWORK AND SDN NETWORK

A typical IP-based network consists of a set of routers and switches. A set of nodes such as the laptop, mobile devices, desktop, etc. is being served by the router. The routing mechanism between the traditional network and the SDN network is described below.

2.1 Traditional Network

In the traditional network scenario, routers execute a specific algorithm to send the incoming packets to the next hop. For example, to access one website, let the data packets have to move through 10 routers, it means 10 router has to run the routing algorithm to serve one request made by the end user. In practice, there are million web users accessing the same web content simultaneously, which create a lot of processing overhead at the router.

In a typical network scenario router can perform two tasks. The first one is the router regularly updating routing tables to get the status of the network, which is commonly known as control plane operation. The second task is to forward the incoming data packets, to the correct destination based on the locally stored routing tables, this is commonly known as forwarding or data plane operation. A cost effective routing algorithm takes decisions based on the routing table of the local devices, determine the shortest path between the source and destination with a minimum time delay and maximum throughput. If the size of the network topology grows the routing table size also relatively increase, which creates a lot of delay since routing information exchange periodically to keep update the network status. Computational burden on the router considerably increase if one of the nodes fails in the network, finding an alternate route other router carries out the same process.

2.2 SDN Network

Since traditional routing devices facing a lot of challenges, SDN overcome it by separating the control plane and data plane. SDN assigns the control plane to an external device called controller, which manages all the routing decisions. Here routers do not have to do computationally expensive

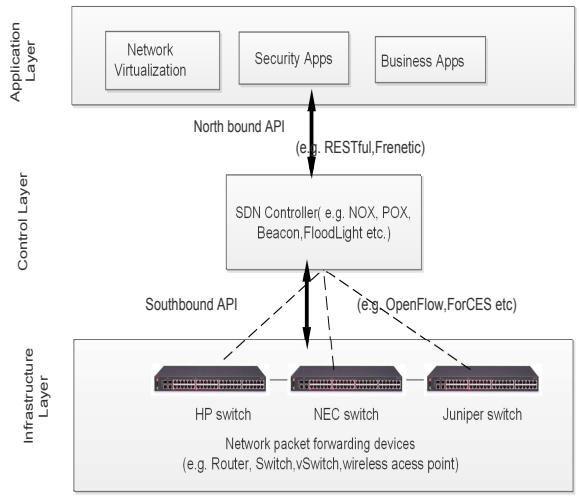

Figure 1: SDN Architecture comprises of application, control, and infrastructure layer.

tasks. All protocols execute in the central controller that takes all the routing decisions. The controller always monitors the global view of the whole network, which can possibly through some well-defined APIs.

3. ARCHITECTURAL DESIGN OF SDN

This section will describe the architecture and principal components of SDN. The fig 1 gives an overall structure of SDN architecture which has describe below.

3.1 Layers of SDN

In [13], ONF describes a high level architecture of SDN, which functionally and vertically split into three layers.

- Infrastructure layer: This layer consists of forwarding devices like the physical switch, router, etc. Software switches which can be accessible via open interfaces, also part of this layer. This layer is considered as forwarding layer since it allows packet switching and forwarding.
- Control Layer: The control layer is also referred as control plane that comprises a set of software-enabled SDN controllers. This layer allows the network administrator to apply custom policies to the physical layer devices. About the controller functionalities will be briefly discussed next.
- Application layer: Application layer deals with end-user business applications that utilizes the SDN services. Business application such as energy efficient networking, security monitoring, network virtualization etc.

3.2 INTERFACES OF THE CONTROLLER

The SDN controller can interact with these three layers, through some standard open interfaces which have discussed in the below section.

- ***Southbound interface***: Southbound interface creates a channel to interact with the controller and underlying forwarding elements. OpenFlow is the standardized protocol supported by ONF, is the widely used southbound interface [13], which establishes a secured link between the controller and forwarding devices.

The remote updating of the switchs′ flow table by the controller is called programming in the SDN perspective. Later we have briefly discussed the OpenFlow protocol and architecture. Another protocol called ForCES is used for a long time for communication between controller and data plane components [2]. Though ForCES and SDN have some common goal, but they differ in many aspects. A nice comparison between ForCES and OpenFlow have been discussed in [21].

- ***Northbound interface***: The north bound APIs represents are interfaces between the controller and the applications application layer.This interface helps the application developers to manage the network through the program. Since the network policies are dynamic in nature in an SDN environment, hence the traditional languages fail to achieve this. To program the network devices Frenetic is used [4], that provides standard libraries and support modular programming that help to design a high-level packet-forwarding policies for switches. Similar to Frenetic, Pyretic is another programming platform, which provides modular programming and has increased abstraction layer that allow the application developer to develop a more challenging application [14]. A control architecture for SDN called Procera, which uses abstraction layers to hide the details that are unimportant and reveal only the relevant information to upper layers [22].
- ***East-west Bridge***: In practice, a large scale enterprise network is partitioned into sub-networks, where each sub-network is handled by a different controller which has shown in Fig 2. In this multi-controller based architecture, each network has the global network view to communicate with the other domain and exchange topology among the sub-domains. Directly intra domain communicate cannot possible without the help of a proper interface. East-west protocols solve this problem by exchanging network view among the controllers. Authors in [12] have proposed an east-west bridge mechanism for intra-domain communication within SDN, that follows the peer- to-peer exchange mechanism of the network information.

4. THE OPENFLOW PROTOCOL

OpenFlow is todays best explored and is a standard protocol that exchange information between the control plane and OpenFlow devices present in the data plane in an SDN environment. This protocol is a multivendor standard and maintained by the Open Networking Foundation (ONF) [**?**]. The ONF is a joint initiative by major software giants like Google, Microsoft, and Microsoft etc. In the year 2011, the first version V 1.1 came out to the market. Many vendors, including Bell, IBM, CISCO, Juniper, etc., are using the OpenFlow standard and add this functionality to their network devices to make the device SDN compatible. Most of the commodity networking hardware devices have started using SDN enabling technology including HP, IBM, NEC, Pronto,Juniper, Extreme, etc.

4.1 OpenFlow enabled Switch Architecture

To define a path from source to destination OpenFlow offers a programmatic control of flows to the network administrators. It allows a way to remove the overhead of routers

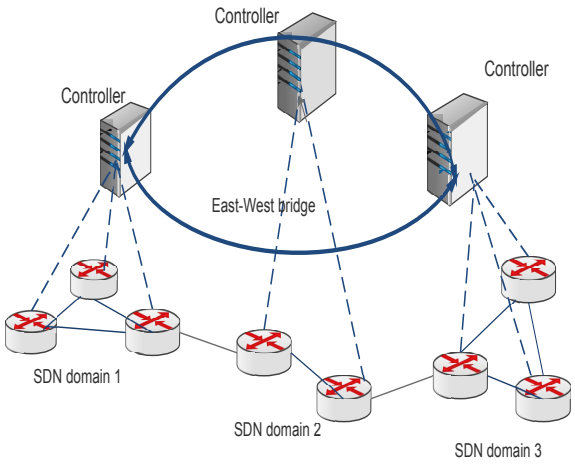

Figure 2: Multiple SDN domains interact each other through East-West interface.

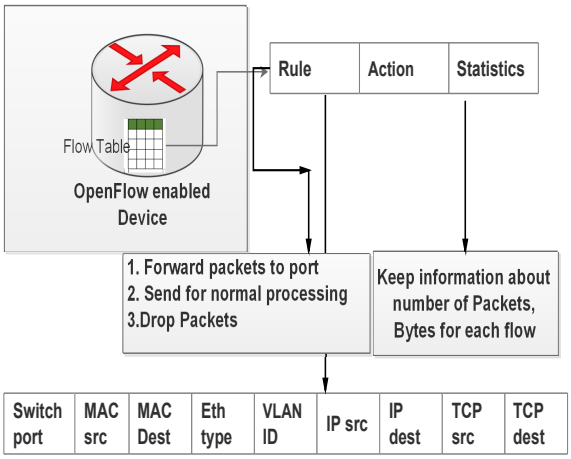

Figure 3: Different matching fields in a flow table.

packet processing for defining the path, and network management costs while escalating a network [8]. In an OpenFLow architecture, the entire system consists of many OpenFlow enabled devices, which are managed by one or more OpenFlow controllers. Again the OpenFlow devices use one or more flow tables. A flow could a set of packets with same MAC IDs or IP addresss, packets with the same VLAN tag or packets are arriving from the port [19]. An abstract model of OpenFlow switch is depicted in Fig 4. Flow table consists of flow entries, which decides how the packet will be processed or forwarded of a particular flow. Each flow entries in the table have three fields: match (header) field, action field and statistics field [10].
Within an OpenFlow switch, basically the packets have gone through a series of flow tables to find its exact match. For every incoming packet, the switch make the forwarding decision by looking into the flow table entries, starts from the first table and ends either with a match in one of the tables or with a miss if no rule present for the packet. A flow rule can be defined by combining different matching fields, such as port number, source or the destination IPv4 address, source or destination MAC address etc. which is illustrated

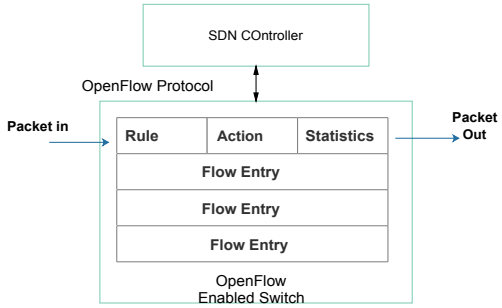

Figure 4: An abstract model of an OpenFlow switch.

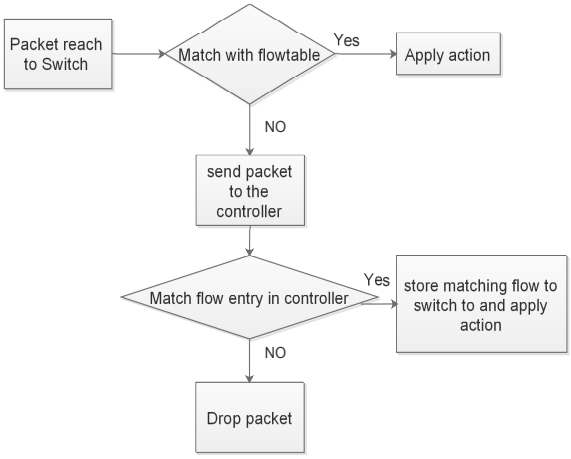

Figure 5: Flowchart of execution sequence of a packet in SDN.

in Fig 3. If there is no such entry in the flow tables for a flow, the packet will be discarded. However, in case of unmatched the switch sends the corresponding packet to the controller. Sometimes the switch sends the unmatched packet to the non-OpenFlow pipeline of the switch [23].

The associated actions pertain to each flow entry might be packet forwarding to the outgoing port, forward to the controller, packet forward to the next flow table for further processing, drop the packet or send the packet for normal processing. The packet might be forward to a physical port, logical or reserved port of a specific flow entry. The third field, i.e. the statistics field of the flow table, keep information about the number of packets, number of bytes for each flow and elapse time since flow initiation.

4.2 Working Principle of OpenFlow Switch

In OpenFlow switch, the flow rules in the forwarding tables are decided by the controller. The controller installs each flow rules to the flow tables. For each incoming packet, the flow tables are looked up and simultaneously the header fields of the incoming packets are matched. If a match is found, the corresponding decision will follow and if no match is found, the packets are forwarded to the controller for additional processing. The processing of packets in OpenFlow protocol can be seen in a flowchart in fig 5.

4.2.1 *Categorization of messages exchanged between switch and a controller*

There is three kinds of messages supported by the OpenFlow

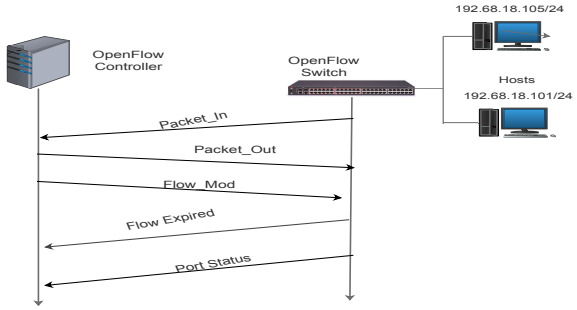

Figure 6: Connection establishment between hosts and the OpenFlow network.

protocol. These messages are: asynchronous, controller-to-switch, and symmetric messages [20]. The controller-to-switch messages are initiated by the controller and used to examine the status and state of the flow table of the switches. The asynchronous messages are sent from the switch to the controller which is referred to as an event and designate a change in the switch state or network state. Among many events, Packet-in event has a noteworthy importance. This event occurs when a packet does not have a matching entry in the flow table, Packet-in message is sent to the controller which decide about the flow establishment for the packet. Lastly, the symmetric messages are sent in either direction. These messages are used to check the liveliness of the controller. Checking of liveliness between controller and switches can be done with the help of Hello and Echo message.

4.2.2 *Establishment of connection between switch and controller*

At the first time, when an Openflow enabled device tries to configure, it first sends a TCP sync message to the controller at the default TCP port i.e. 6633 [7]. A TCP handshake takes place between the controller and switch by sending an acknowledgment message from either end. The connection establishment process is shown in the Fig.6. In the following, the TCP handshake messages are concisely discussed [20].

- Hello : The controller sends its version number to the OpenFlow switch through Hello message in turn the switch replies its version number to the controller.
- Features Request: After getting the version, controller seeks the available ports from the switch.
- Features Reply: In turn, the switch response with a list options such as set of available ports, the port speeds etc.
- Set Config: Set Config message signifies that the controller wishes, the switch has to send flow expirations.

4.2.3 *Connection between hosts on OpenFlow environment*

After a connection has established between the switch and the controller, now the host can communicate with the network. How a host connects to an OpenFlow network with certain message exchange is depicted in Fig 6. Apart from Packet_in we discuss some other messages which are listed below.

- Packet_In: This message is sent by the switch when any packet does not have any flow entry in any one of the flow table.
- Packet_Out: This message is sent out by the controller when a packet has to send through a port of a switch.

- Flow Mod: This is a controller initiated message to the switch. Through this control message, a switch adds a particular flow to its flow table.
- Flow-Expired: This message is initiated by the switch to the controller, when the flows got expired.
- Port Status: Port status such as addition, removal, and modification are being notified to the controller by the Switch.

5. CONTROLLER

Among the three layers of SDN architecture, controller resides in the control layer. It is the main part of SDN that works between network devices and various applications which provide a programmatic interface to the network. An SDN controller has the sole responsibility to manage the network protocols, policies and establish the network path of a flow by installing flow rules on the network devices.

5.1 Implementation Structure

An OpenFlow switch can establish a secured connection between a single controller and multiple controllers. The usage of multiple controllers in SDN, basically addresses two major issues: system failover and load balancing. In a typical SDN network scenario the controller runs on a network-attached server. The implementation structure of SDN controller follows either centralized or distributed structure. Single point of failure is a problem to the centralized structure; hence to allow for the backup process, OpenFLow introduced multiple controller connections for a single network device. Thus the idea of Onix is to maintain a logically centralized but physically distributed control plane [9]. This mechanism reduces the overhead of the local controllers but the centralized view of the entire network can be achieved through specific applications. Another type of hypervisor called FlowVisor allow multiple logical controllers to SDN to achieve network virtualization on an OpenFlow network [18].

5.2 Flow-setup Mode

There is two flow set up modes i.e. proactive and reactive modes are followed by the controller. In proactive flow setup mode, the flow rules are installed in the flow table before the processing of the packet of a flow start. The flow entries of the table installed before the first packet of a flow arrive to the switch. In proactive mode, since the flow rules are pre-installed in the device, the flow setup delay and the controller to the device is minimized. But the disadvantage is, it may overflow the flow table since it does not require the prior knowledge of the flow table size. On the other hand in reactive flow setup, as soon as the first packet reaches to the switch the flow rules are established in the flow table by the controller. In a regular time interval, the flow rules become inactive, hence drain out the previous rules from the flow table is essential. Upon getting a flow setup request, the controller checks the policies of the flow and takes the decision what action has to be taken. Among many commercially available controller, NOX was the first OpenFlow based single threaded controller [5].
The controllers′ performance can be measured by testing the number of flow request handled per second and time taken to respond to these request. The Ethane controller was handled upto 11,000 flow request and the response time was 1.5 ms. More advanced controller like NOX-MT can handle 1.6 million flow request with an average response time 1.5 ms on a 8 core machine. Some available controllers and implementation languages are given in the below Table 1 [6].

6. MAJOR BENEFITS OF SDN

The benefits of SDN are numerous to the organizations. Here a list of benefits is being highlighted.

- ***Efficient use of resources:*** Resources in an NVE can be efficiently utilized with the help of SDN. SDN has the capability to distribute the workload among the controller, which increase the speed and efficiency.
- ***Efficient network administration:*** SDN allows the network administrator to change the characteristics of the network remotely. An easy and efficient network management possible by changing the network characteristics based on the arrival of the workload in the network.
- ***Cross Tenant data center optimization:*** Existing cloud architecture does not support cross tenant functionality. But the SDN can implement in a multitenant environment such as data center and data clouds. The coupled architecture and resource virtualization is well suited for multitenant data center optimization.
- ***Programmability of the network:*** SDN has the ability to control the entire network programmatically. SDN facilitate not to deploy custom policies and protocols on each device separately in a network. Programmability is possible on the control plane itself, through which the behavior of the specific device or whole network can be changed. The controller can enhance the traffic engineering capabilities and reduces the congestion in the network [15].
- ***Virtual and physical network management:*** Network virtualization is the process of providing network resources to end users through virtual network collection resources from multiple Infrastructure Providers. Virtual network is the collection of virtual link and nodes on the underlying physical network. SDN has the ability to manage both physical and virtual network by using a central management tool.
- ***Reduced cost:*** Most of the SDN products are open source. Some products like VMwares NSX, Microsofts Hyper V network virtualization required to pay only the license fee for the SDN solution. On the other hand since SDN support upto layer-3 , no need to purchase expensive hardware by the enterprise which reduces the CAPEX.
- ***Centralized network management:*** SDN allows a centralized view of the entire network status that makes easier to the network administrator to manage the network device efficiently.
- ***Enhanced security:*** In a virtualized environment, providing security to the VM is a very difficult task. But SDN provides a fine-grained security to the end devices [16].

7. CONCLUSIONS

This paper provides an overview of Software Defined Network for researchers. We have provided a detailed architecture of OpenFlow switch, controller, and Switch-controller interaction along with discussed the benefits of SDN to the industry. Several organizations have started working on

Table 1: Controller Classification.

Controller	Developer	Implementation	Architecture
Beacon	Stanford	Java	Centralized multi-threaded
Floodlight	BigSwitch	Java	Centralized multi-threaded
Maestro	Rice University	Java	Centralized multi-threaded
NOX	Nicira	Python/C++	Centralized
OpenDayLight	Opendaylight	Java	Distributed
POX	Nicira	Python	Centralized
RouteFlow	CPqD	C++	Distributed
Maestro	Rice University	Java	Centralized multi-threaded

standardized the protocols of SDN aiming to provide better SDN solutions. IETF, ONF, and ITU-T have been trying to standardize OpenFLow. Despite of its popularity, it has some issues which need to be explored. In our future work we will focus and survey on various research challenges pertain to controller layer of SDN such as security challenges of the controller, traffic engineering, and controller placement problem.

8. REFERENCES

[1] K. Bakshi. Considerations for software defined networking (sdn): approaches and use cases. In *Aerospace Conference, 2013 IEEE*, pages 1–9. IEEE, 2013.

[2] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern. Forwarding and control element separation (forces) protocol specification. Technical report, 2010.

[3] S. Fang, Y. Yu, C. H. Foh, and K. M. M. Aung. A loss-free multipathing solution for data center network using software-defined networking approach. In *APMRC, 2012 Digest*, pages 1–8. IEEE, 2012.

[4] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. Frenetic: A network programming language. In *ACM Sigplan Notices*, volume 46, pages 279–291. ACM, 2011.

[5] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, 2008.

[6] Y. Jarraya, T. Madi, and M. Debbabi. A survey and a layered taxonomy of software-defined networking. *IEEE Communications Surveys & Tutorials*, 16(4):1955–1980, 2014.

[7] V. Khatri. Analysis of openflow protocol in local area networks. 2013.

[8] H. Kim and N. Feamster. Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119, 2013.

[9] T. Koponen, M. Casado, N. Gude, and J. Stribling. Distributed control platform for large-scale production networks, Sept. 9 2014. US Patent 8,830,823.

[10] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[11] A. Lara, A. Kolasani, and B. Ramamurthy. Network innovation using openflow: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):493–512, 2014.

[12] P. Lin, J. Bi, and Y. Wang. East-west bridge for sdn network peering. In *Frontiers in Internet Technologies*, pages 170–181. Springer, 2013.

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[14] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker. Modular sdn programming with pyretic. *Technical Reprot of USENIX*, 2013.

[15] K. S. Sahoo and B. Sahoo. Sdn architecture on fog devices for realtime traffic management: A case study. In *International Conference on Signal, Networks, Computing, and Systems (ICSNCS), New Delhi, India*. Springer, 2016.

[16] K. S. Sahoo, B. Sahoo, and A. Panda. A secured sdn framework for iot. In *2015 International Conference on Man and Machine Interfacing (MAMI)*, pages 1–4. IEEE, 2015.

[17] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao. Are we ready for sdn? implementation challenges for software-defined networks. *IEEE Communications Magazine*, 51(7):36–43, 2013.

[18] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, pages 1–13, 2009.

[19] M.-K. Shin, K.-H. Nam, and H.-J. Kim. Software-defined networking (sdn): A reference architecture and open apis. In *2012 International Conference on ICT Convergence (ICTC)*, pages 360–361. IEEE, 2012.

[20] O. S. Specification. Version 1.4. 0, october 14, 2013.

[21] T. Tsou, X. Shi, J. Huang, Z. Wang, and X. Yin. Analysis of comparisons between openflow and forces. *Analysis*, 2012.

[22] A. Voellmy, H. Kim, and N. Feamster. Procera: a language for high-level reactive network control. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 43–48. ACM, 2012.

[23] L. Yang, R. Dantu, T. Anderson, and R. Gopal. Forwarding and control element separation (forces) framework. Technical report, 2004.