International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 11 – No. 7, December 2016 – www.ijais.org

Introduction to Software Defined Networks (SDN)

Ominike Akpovi A.
ICT Department
Petroleum Training Institute
Delta State, Nigeria.

Adebayo A. O
Comp. Sc. Dept.
Babcock University
Ogun State, Nigeria.

Osisanwo F. Y.
Comp. Sc. Dept.
Babcock University
Ogun State, Nigeria.

Ebiesuwa Seun
Comp. Sc. Dept.
Babcock University
Ogun State, Nigeria.

ABSTRACT
Legacy IP networks are complex, difficult to manage and are also vertically integrated i.e. the control and data planes are bundled together. The concept behind Software Defined Networks (SDN) is to break the vertical integration, by separating the network control logic from its underlying hardware, (routers and switches) promoting (logical) centralization of network control, and introducing the ability to program the network. In this paper, we start by introducing the motivation for SDN, its architecture and how it differs from traditional networking. Open flow is discussed next and SDN simulation tools are also discussed. The applications and main ongoing research efforts and challenges of SDN are also mentioned.

General Terms
Software Defined Networks (SDN), Openflow, Controllers

Keywords
Software Defined Networks (SDN), Programmable Networks

1. INTRODUCTION
The definition of Software Defined Networks SDN by the Open Network foundation (ONF) is the most acceptable [1]. In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized and the underlying network infrastructure is abstracted from the application. SDN focuses on separation of the control plane from the data plane, centralized controller and view of the network, open interfaces between the devices in the control plane (controllers) and those in the data plane and programmability of the network by external applications. [2]

Legacy data and communications networks are complex and difficult to manage. They involve different equipment that run complex distributed control software that is closed and proprietary. Traditional IP networks are vertically integrated, that is, they have the control and data plane bundled together. The goal of Software Defined Networks, SDN is to make networks more programmable. [3]. The programmability of the network is achieved by means of software applications which run on top the network operating system (NOS)

SDN separates the control and data plane and it promotes logical centralization of network control and introduces the ability to program the network. SDN also makes it easier to create and introduce new abstractions in networking, simplifying network management and facilitating network evolution. In addition, SDN breaks vertical integration by separating the two planes; The control plane – which decides how to handle network traffic, and the data plane – which forwards traffic according to the decisions made by the control plane. [4]

The control plane sometimes called the controller, has the logic of controlling and forwarding behavior like tracking topology changes, installing forwarding rules, computing routes etc. The control plane can be implemented completely in software and installed on commodity hardware The data plane on the other hand forwards traffic based on rules as dictated by control plane logic like forward, filter, buffer, rate-limit and measure packets. Some of the benefits of separating the data and control plane (SDN) are:

1. Global network view- the controller can see the status of all routes and switches quickly deciding the best route
2. Horizontal Integration- Separate and independent growth of hardware and software and flexibility in choosing hardware and software by customer.
3. Elimination of Middleboxes (Middleboxes are devices that manipulate traffic for purposes other than packet forwarding e.g. firewalls, server load balancers, network address translators etc.)
4. Easier deployment of new network services and protocol [5]

2. RELATED WORKS
Nunes, B.A. a. et al., 2014, discussed early programmable networks that laid the foundations for the SDN of today. Some of the early programmable networks discussed included (i) Open Signaling- which allowed the deployment of new services through a distributed programming environment (ii) Active Networking- that proposed the idea of a network infrastructure that would be programmable for customized services. (iii) The 4D Project- that advocated a clean slate design that emphasized separation between the routing decision logic and the protocols governing the interaction between network elements and (iv) Ethane- which was the immediate predecessor to OpenFlow. Ethane's focus was on using a centralized controller to manage policy and security in a network. Ethane laid the foundation for what would become Software-Defined Networking.

Fei Hu et al, 2014, discussed other SDN Standards besides OpenFlow (the most popular SDN protocol/standard). For instance, ForCES (Forwarding and Control Element Separation) which proposes the models to separate IP control and data forwarding, Transport Mapping layer for the forwarding and control elements, logical function block library for such a separation, etc. and SoftRouter, which

defines clearly the dynamic binding procedure between the network elements located in control plane (software-based) and data plane.

Feamster, N., Rexford, J. & Zegura, E., 2013. discussed the history of programmable networks. In this paper, they divided the history into three stages, each with its own contributions: (1) active networks (from the mid-1990s to the early 2000s), which introduced programmable functions in the network, leading to greater innovation; (2) control- and data-plane separation (from around 2001 to 2007), which developed open interfaces between the control and data planes; and (3) the OpenFlow API and network operating systems (from 2007 to around 2010), which represented the first widespread adoption of an open interface and developed ways to make control- and data-plane separation scalable and practical.

Kreutz et al., 2014, summarized different instances of SDN-related work prior to SDN, splitting it into five categories-data plane programmability, control and data plane decoupling, network virtualization, network operating system and technology pull initiatives.

3. SDN ARCHITECTURE

As shown in Fig. 1 below, the bottom tier involves the physical network equipment including Ethernet switches and routers and this forms the data plane. The central tier consists of the controllers that facilitate setting up and tearing down flows and paths in the network. The central tier (control layer) links with the bottom tier (infrastructure layer) via an application programming interface (API) referred to as the southbound API. OpenFlow, sFlow and NetFlow are such protocols which are used for traffic analysis and monitoring in a network. Connections between controllers operate with east and westbound APIs. The controller application interface is referred to as the northbound API. [2]

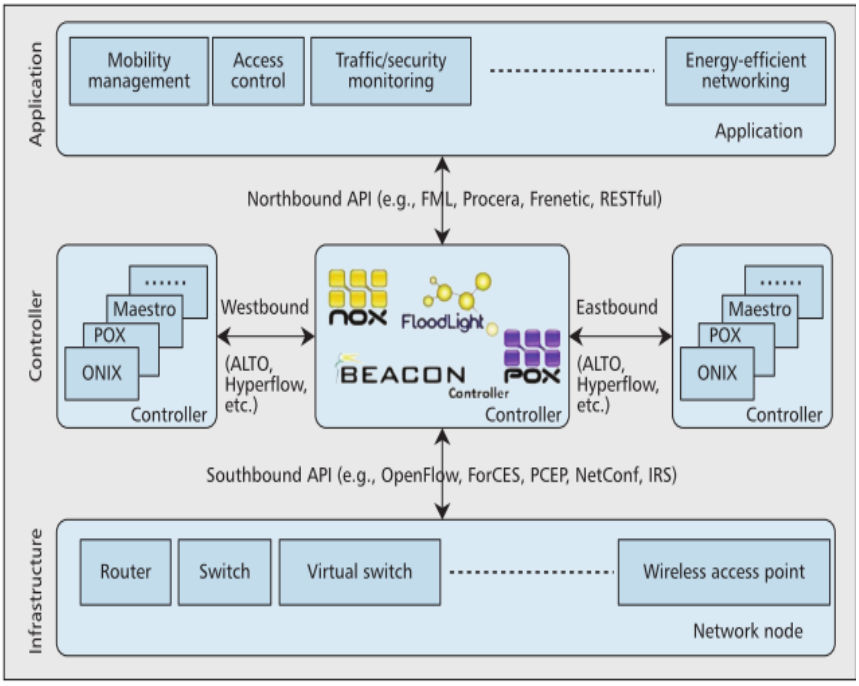

Figure 1: SDN Architecture [2]

The management application defines the policies, which are translated to southbound instructions that program the behavior of the forwarding devices. [4] The northbound API can be used to develop applications for network management, load balancing etc. There is no generally accepted standard protocol for the northbound API. The most common and accepted standard for southbound communication is OpenFlow.

In legacy networks (closed to innovations in the infrastructure) as shown in Fig. 2 below, the system is closed with custom hardware features and applications. i.e. you have specialized applications, operating systems and hardware and this slows down innovations in the network.

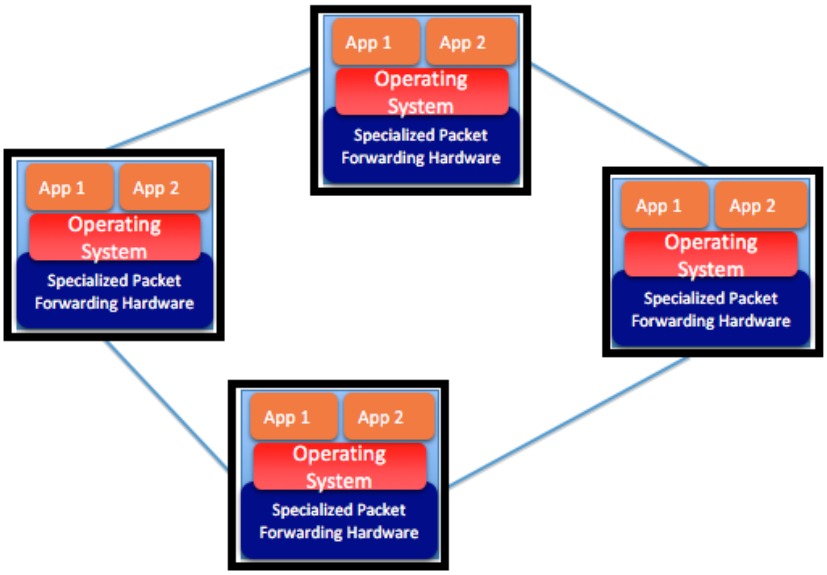

Figure 2: Legacy Network [11]

In a software defined network however, devices are horizontally integrated, we have open interfaces and there are huge opportunities for rapid innovations. Fig. 3 describes the SDN approach to open networks. It shows a bunch of switches connected by network cables. The network operating system is then built (which is essentially software running on servers). The servers talk to other switches using open flow and the servers use the information they get from the switches to build a global network view (i.e. to know the topology).

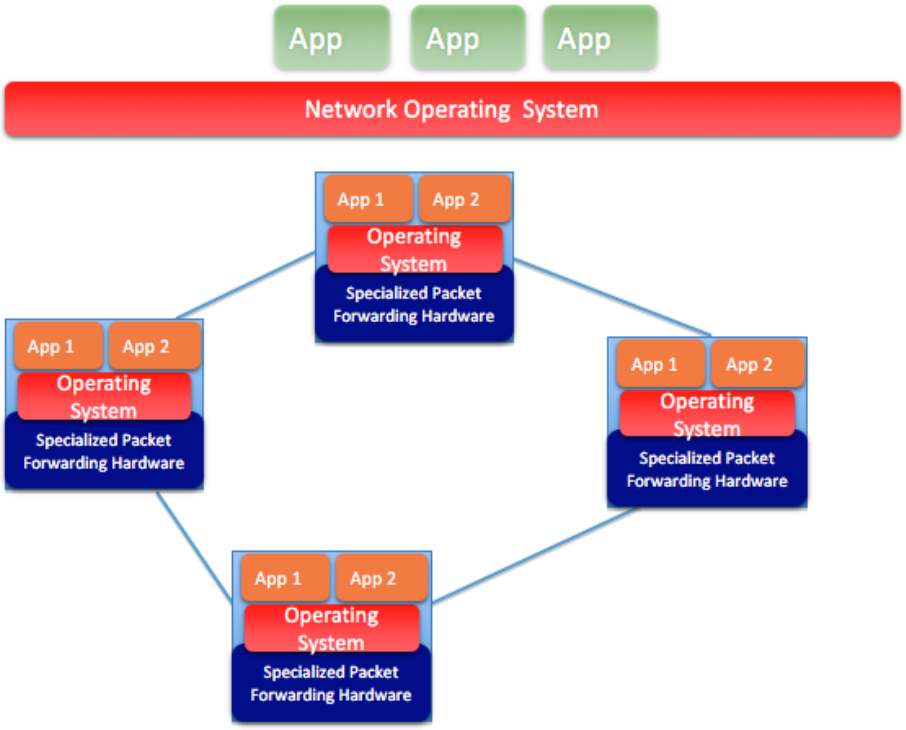

Figure 3: SDN approach to Open Networks [10]

The control programme is then built on top of that topology (routing, access control etc.). The control program expresses the operators' goals e.g. connectivity, isolation etc. implemented on top the global network view. The Network Operating System (NOS) links global view and physical switches i.e. gathers information for the global network view while the switches / routers follow orders from the NOS [12] Fig. 4 shows a software defined network with an open interface to packet forwarding, the network OS and a well defined API.

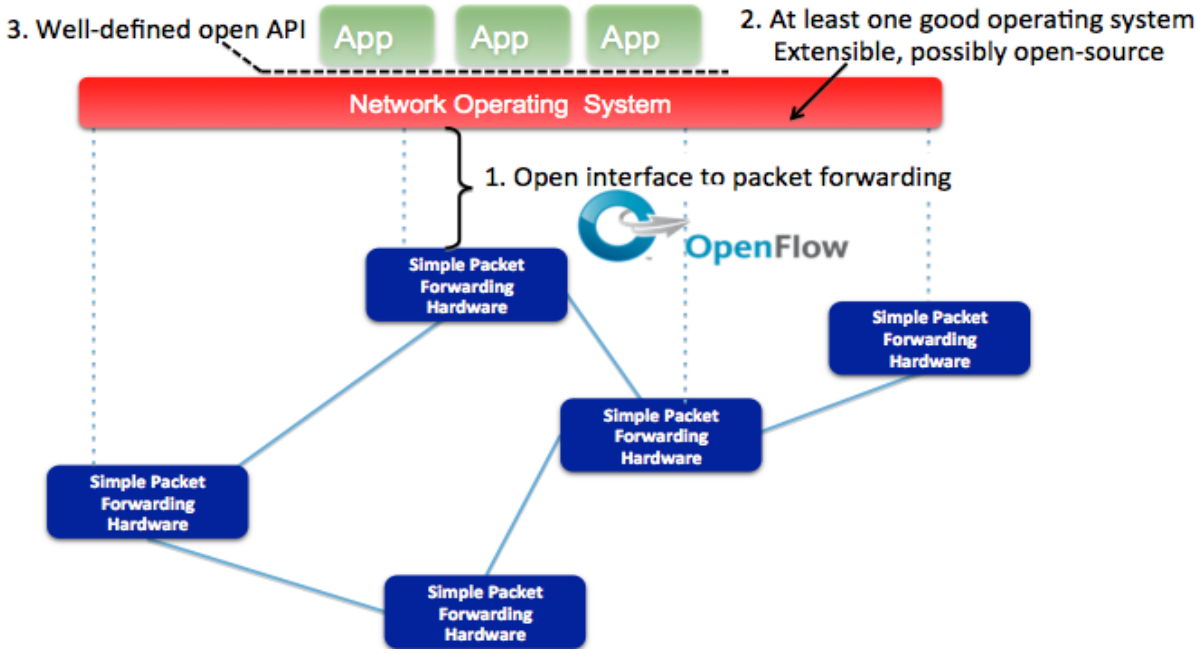

Figure 4: The Software Defined Network [10]

3.1 SDN and Openflow

The Openflow architecture consists of three principal elements: An Openflow switch, an external controller and the Openflow Protocol which establishes the communication between switch and controller through a secure channel. The data plane and the control plane communicate over a secure channel over a standard protocol called OpenFlow [5]. The OpenFlow switch has flow tables and an abstraction layer that securely communicates with a controller via OpenFlow protocol. The flow table contains flow entries that determine how packets belonging to a flow will be processed and forwarded. Flow entries consist of

- Matching rules – to match incoming packets
- Counters – to collect statistics of the flow
- Set of instructions / actions - to be applied upon a match

When a packet arrives at an OpenFlow switch, the packet header fields are extracted and matched against the matching rules. If a match is found, the switch applies the appropriate action. If there is no match, the action taken by the switch depends on the instruction defined by the table-miss flow entry. Every flow table must have a table-miss entry in order to handle table misses. Examples of actions that can be carried out when no match is found are – drop packet, continue matching process on next flow table, forward packet to controller etc [6] Table miss – if a packet does not match a flow entry in a flow table. The implication is that the packet could be dropped, passed on to another table or sent to a controller. [7]

3.2 SDN Simulation Tools

Mininet is the first open source simulator for Openflow networks. It is an inexpensive and quickly configurable network testbed. It is the most well known tool for SDN OpenFlow network research. The virtual switches in Mininet are a kind of software OpenFlow switches called "Open vSwitch" [8] With mininet the entire network can be packaged as a virtual machine so that others can download, run, examine and modify it. The major strengths of mininet are – it is good for prototyping, it is easily deployed and easily shared. [9]. Other prominent SDN platforms are estinet, NS-3 and Trema.

3.3 Applications of SDN

SDN can be used to improve network management by eliminating middle boxes and carry out flexible network configuration changes in in enterprise and data centre networks, telecommunications and Internet service provider (ISP) networks. SDN can also be deployed in backbone networks.

4. CHALLENGES

Despite the numerous benefits derivable form SDNs, a few challenges also exist:

1. Controller Design / Performance- SDN control plane may have multiple controllers depending on the network design and topology. If the controller is not properly designed, there could be failure in the network and this can adversely affect the resilience of the network.

2. Integration of SDN with Legacy Networks- Traditional network equipment could be enhanced or out rightly replaced to support SDN and OpenFlow. It is importance that efforts are made to efficiently integrate SDN devices with legacy networks for optimal network performance.

3. Migration to SDN- Asides the technical and cost implications involved, there is also the challenge of convincing users to switch from legacy networks to SDN especially if the networks are performing optimally. In some quarters, SDN could also be viewed as a threat to their jobs.

4. Security- The controller is the key device in the SDN model and thus securing the controller is important because if the controller is breached, then there is the risk of a total network outage. It is important to note however that security concerns are

not peculiar to SDN alone.

5. CONCLUSION

SDN has gained significant momentum in both the research community and in the industry. It is going to become the new approach for networking. Although SDN has its own limitations and challenges, it offers other significant benefits and cost savings such as its programmability, providing a global view of the whole network, providing more flexibility & control to researchers & network administrators, network equipment vendor independency and eliminating middleboxes. Future work can involve improving security of SDN and enhancing the controller design for scalability, resilience and robustness.

6. REFERENCES

[1] W. Braun and M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices," *Futur. Internet*, vol. 6, no. 2, pp. 302–336, May 2014.

[2] S.Sakir, S.Sandra, C. Kaur, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Marc, and N.Rao "Are We Ready for SDN ? Implementation Challenges for Software-Defined Networks," no. July, pp. 36–43, 2013.

[3] N. Feamster, J. Rexford, and E. Zegura, "The Road to SDN," *Queue*, vol. 11, no. 12, pp. 20–40, Dec. 2013.

[4] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," p. 49, Jun. 2014.

[5] P. Ranjan, "A Survey of Past , Present and Future of Software Defined Networking," vol. 7782, pp. 238–248, 2014.

[6] 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 Commun. Surv. Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[7] K. Bakshi, "Considerations for Software Defined Networking (SDN): Approaches and use cases," *2013 IEEE Aerosp. Conf.*, pp. 1–9, Mar. 2013.

[8] S.-Y. Wang, "Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet," *2014 IEEE Symp. Comput. Commun.*, pp. 1–6, Jun. 2014.

[9] B. Lantz, B. Heller, and N. Mckeown, "A Network in a Laptop : Rapid Prototyping for Software-Defined Networks," pp. 1–6, 2010.

[10] Munakami, M (2014), Software Defined Networking [Power Point Slides] Presented at Boise State University.

[11] Fan, S (n.d), Software Defined Networking [Power Point Slides] Presented at Duke University.

[12] Stanford Seminar. (2013, May 9). Software-Defined Networking at the Crossroads. [YouTube video]. Available: http://www.youtube.com/watch?v=WabdXYzCAOU. Accessed July 27, 2016.