Software Defined Networks: What Is It and Why Do We Need It?

A. Michele Parrish

East Carolina University

Author Note

A. Michele Parrish, Department of Technology Systems, East Carolina University

Contact: parrishan15@students.ecu.edu

ABSTRACT

As more devices are connected to the Internet there is an increased demand for more resources that must be provisioned rapidly. In traditional networking, a network was designed for a long period of time; the engineers thought about the clients, the servers and internetworking devices (switches and routers) and how they would all work together. The components were bought and downtime was scheduled to install new systems or upgrade current ones. Today's Internet of Thing (IoT) environment, in which everything from your cell phone to the refrigerator to the car is connected to the Internet, does not allow for significant time for planning and implementation of those new resources. Resources must be able to be allocated and managed as quickly and efficiently as possible. The traditional environment of each switch or router having to be managed independently does not allow network administrators the ability to react swiftly to the new demands nor does it allow the switches or routers to adjust their information quickly enough. Software Defined Networking (SDN) is an architecture that creates a virtualized network that can meet the requirements of today's IoT environment. In this paper the components of SDN will be examined, types of SDNs, and reasons and benefits for using SDN. OpenFlow, a protocol that is used on SDNs, and the connection between network virtualization (NV), network function virtualization (NFV) and SDNs will be will be explained. There are also challenges associated with SDN and this paper will look at them and possible solutions. Lastly the future of SDNs will be addressed.

INTRODUCTION

As more devices are connected to the Internet there is an increased demand for more resources that must be provisioned rapidly. In traditional networking, a network was designed for a long

period of time; the engineers thought about the clients, the servers and internetworking devices (switches and routers) and how they would all work together. The components were bought and downtime was scheduled to install new systems or upgrade current ones. Today's Internet of Thing (IoT) environment, in which everything from your cell phone to the refrigerator to the car is connected to the Internet, does not allow for significant time for planning and implementation of those new resources. Resources must be able to be allocated and managed as quickly and efficiently as possible. The traditional environment of each switch or router having to be managed independently does not allow network administrators the ability to react swiftly to the new demands nor does it allow the switches or routers to adjust their information quickly enough. Software Defined Networking (SDN) is an architecture that creates a virtualized network that can meet the requirements of today's IoT environment. In this paper the components of SDN will be examined, types of SDNs, and reasons and benefits for using SDN. OpenFlow, a protocol that is used on SDNs, and the connection between network virtualization (NV), network function virtualization (NFV) and SDNs will be will be explained. There are also challenges associated with SDN and this paper will look at them and possible solutions. Lastly the future of SDNs will be addressed.

The IoT has grown because of the use of virtualization and cloud computing. Traditional data centers have to change so they can implement these technologies without degrading the performance of switches and routers. These devices consists of application specific hardware chips, the hardware and a software stack that are tightly integrated (Hegde, Koolagudi, Bhattacharya, 2015). A set of specific and predefined linked commands are used on the device's operating system. Errors are more likely to occur in management of a large number of devices

(Masoudi, Ghaffari, 2016). Each switch independently decides which route each frame will take and the decision-making for switching or routing reside on the same device so that network decision-making is spread across different network components (Jammal, Singh, Shami, Asal, Li, 2014). All of this makes it hard for a network to evolve quickly enough to keep up with virtualization, cloud computing and new applications' demands of increased bandwidth, greater accessibility and dynamic management in today's IoT environment. SDN is the solution to overcome these issues.

Traditional network devices, like switches and routers, contain a control plane and a data plane in the same device. The control plane is the "brains of the device" and makes the forwarding decisions. It has Data Link and Network layer forwarding mechanisms like routing protocol neighbor tables and topology tables, IPv4 and IPv6 routing tables, Spanning Tree Protocol (STP) and Address Resolution Protocol (ARP) table (Cisco, 2016). The data plane, also called the forwarding plane, is the fabric connecting the network ports on a device and is used to forward traffic flows (Cisco, 2016). In SDN the control plane is moved from the device to a centralized SDN controller that is "intelligent" and contains policy-making abilities (Cisco, 2016).

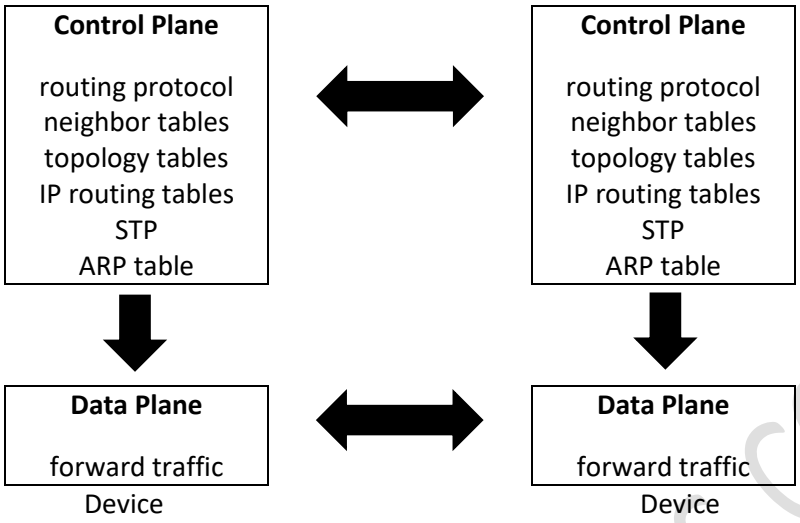

Fig. 1. Traditional network device

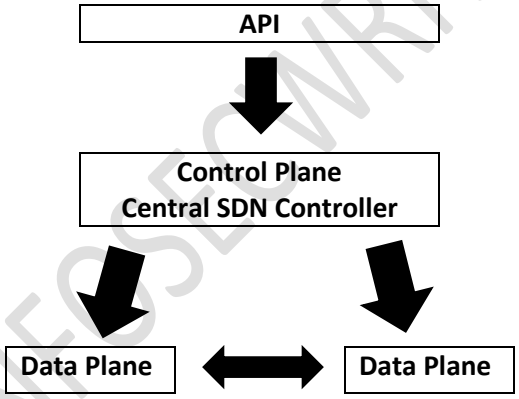

Fig. 2. SDN Architecture

COMPONENTS OF SDN

SDN has three features that set it apart from traditional network architecture. They are that the control plane and data plan are separated, the SDN controller allows centralized control and view of the network and external applications can use the controller to program the network (Hedge et al., 2015). The control plane (controller) allows an abstract view of the entire network infrastructure. This allows the administrator to deploy custom policies and protocols across all

network hardware (Jammal, Singh, Shami, Asal, Li, 2014). The data plane forwards data in the SDN architecture. It contains the Southbound protocols that are used to control and manage the interfaces between the different types of network equipment. The most popular protocol is OpenFlow (Jammal et al., 2014). In addition to these two main components, SDN also has includes a Northbound application interface and East-West protocols. The Northbound application interface is the Application Programming Interface (API) that sit "between the software modules of the controller platform and the SDN applications running atop the network platform" (Jammal et al., 2014). The APIs are currently proprietary but the Open Daylight Project is working to create a standard for controllers' Northbound interface (Masoudi, Ghaffari, 2016). East-West protocols manage the interactions between the different controllers (Jammal et al., 2014).

The ingress switch encapsulates the first packet of a new flow of data in the SDN architecture and sends it to the controller. The required module is run by the controller to discover a path for the flow. The rules for forwarding are installed into the flow tables (FT) on each switch along the path of the flow. Flows that match the entries on the switches are acted on based on the controller's instructions. If a packet doesn't match a flow entry then it is discarded or sent to the controller (Hedge et al., 2015). Traditionally, each routing decision was done locally in a distributed manner. SDNs allow centralized routing and any changes in the network can be dynamically reacted to by the controller. Policy enforcement can also be centrally done by the controller (Hedge et al., 2015).

TYPES OF SDNs

There are three types of SDNs; device-based SDNs, controller-based SDNs and policy-based SDNs. Device-based SDNs are programmable by applications running on the device itself or on a network server. Programmers can build applications using C and Java with Python. Cisco OnePK is a device-based SDN that can be integrated and interact with Cisco devices. Controller-based SDNs use a centralized controller that knows about all devices in the network. Applications can interface with the controller that manages the devices and manipulates the traffic flow. An example of a controller-based SDN is Cisco's Open SDN controller. This is a commercial version of OpenDaylight. Policy-based SDNs are similar to controller-based SDNs and have a view of all network devices but it has an additional Policy layer that operates at a higher abstraction level. It has built-in applications to automate configuration tasks via a guided workflow and GUI so no programming skills are needed. Cisco's APIC-EM is an example of a policy-based SDN (Cisco, 2016).

BENEFITS

SDN has many benefits. The first is the ability to program the network from one interface. Each device in the network doesn't have to have policies and protocols implemented on them. This can be done on the external controller. New devices can easily be introduced into the environment. Policies and protocol changes can be implemented at the controller and populated to a certain device or throughout the network on a secure channel (Jammal et al., 2014). Transport Layer Security (TLS) or OpenFlow can be used for communicating the configuration (Cisco, 2016).

Secondly, as the implementation of virtual machines (VMs) continue to increase, networks must be able to accommodate them and their migration. When a VM is moved, the media access control (MAC) table must be updated and this can cause temporary accessibility issues in traditional networks. An SDN application, network virtualization, provides a solution for this. Tunnels that can abstract the MAC address from the physical infrastructure can enable Layer 2 traffic to run over Layer 3 overlays. This simplifies the deployment and migration of VMs (Jammal et al., 2014).

SDN allows a single point on the network for device configuration and troubleshooting. It can "encourage innovation in the networking field by offering a programmable platforms for experiments on novel protocols and policies using production traffic" (Jammal et al., 2014). New protocols and ideas can be adopted because of the separation of data flows and test flows. Routing decisions can be separated from switching hardware (Jammal et al., 2014).

Lastly, SDNs can save energy. The energy efficiency of network links is very low. The amount of energy consumed is not in proportion of the utilization of the links. Since SDN controllers have a view of the entire network, the controller can determine power states and routing decisions. If it uses protocols for power-state control, they can power off elements, like links and switches, that are not being used (Wang, Li, Jim, Hui, Wu, 2016). Wang et al. (2016) proposed a modification of the OSPF protocol to turn off links along with energy-aware routing algorithms.

OPENFLOW

SDN and OpenFlow are not the same thing. Many are confused about this and it may be because the term SDN was coined after OpenFlow was introduced. OpenFlow is just one API that can be used in SDNs (Farhady, Lee, Nakao, 2015). It is the most widely deployed SDN API (W. Li, Meng, Kwok, 2016). OpenFlow is an open standard that allows the control plane (SDN controller) to talk to the data plane (devices). In 2008, the OpenFlow Consortium proposed OpenFlow. The Opening Networking Foundation is now in control of the OpenFlow specifications (Chen, Ma, Kuo,Yang, Hung, 2015). According to Chen et al. (2015) OpenFlow has three components:

1. OpenFlow switch – this can be a router or a switch that packets will traverse.
2. Secure channel – safe channel that allows information to flow between OpenFlow switch and SDN controller.
3. Controller – uses a secure channel to monitor and manage network devices.

NETWORK VIRTALIZATION

Virtualization is the solution to reduce capital expenditures (CapEx), operational expenditures (OpEx), to overcome the reliance on proprietary applications and diverse, purpose-built network devices. Virtualization allows network hardware and their services to be implemented as software (Y. Li, Chen, 2015). There can be confusion between SDNs, network virtualization (NV) and network function virtualization (NFV). NV allow for the controlling of packets' routing and load balancing by using unused wires to share the load (Chen et al., 2015). This allows multiple network instances to travel across a shared physical infrastructure so NV can be used to run a SDN solution. SDN can also be used to implement NVs because it can distinguish and forward flows to different slices (Farhady et al., 2015).

NFV was created by the ETSI Industry Specification Group to allow network functions that were carried out by proprietary dedicated hardware to be virtualized. This provides "flexible provisioning of software-based network functionalities on top of an optimally shared physical infrastructure. It addresses the problems of operational costs of managing and controlling these closed and proprietary appliances by leveraging low sot commodity services" (Y. Li & Chen, 2015, p. 2542). With NFV the SDN controller (a network function) can be virtualized. Dynamic migration of the controllers to optimal locations can then take place. SDN provides programmable networks connectivity between VNFs to optimize traffic engineering and steering (Y. Li & Chen, 2015). As explained by W. Li & Chen (2015) the differences between NFV and SDN are:

- NVF implements network functions in a software manner. SDN achieves centrally controlled and programmable network architecture that provides better connectivity.
- NFV reduces CapEx, OpEx, space requirements and power consumption. SDN provides network abstractions for flexible network control, configuration and innovation.
- NFV decouples the network functions from proprietary hardware so it can provide agile provisioning and deployment. SDN decouples the control plane and the data plane to provide a central control that enables programmability.

CHALLENGES

There are many challenges faced by SDNs. One is scalability. As SDNs become more widely implemented, and the number of devices that each control plan has to control increases, the SDN controller will have to be able to scale to handle a greater number of devices. It is estimated that a large data center with 2 million virtual machines can generate up to 20 million flows per

second (W. Li et al, 2016). Two scalability issues are present; communication between the switch and the controller, and the memory required on the switches to store the forwarding rules (Hegde, 2015). One solution is to design a distributed architecture for the SDN controller. Reducing the overhead of the controllers would also improve scalability. A modification of OpenFlow called DevoFlow reduces the number of interactions between the controllers and switches (Masoudi & Ghaffari, 2016). DIFANE "reduces the number of requests to a controller by proactively pushing the flow entries to switches" (Masoudi & Ghaffari, 2016, p. 9). Another solution is to implement source routing. In source routing the forwarding rules are not stored on the intermediate switches so the controller doesn't have to communicate with the switches along the path (Hedge, 2015).

In the initial design on SDN security was not considered and still has not been fully tested since they are in limited implementation. Because the controller is the logical centralization of the network it is a potential single point of failure and a target for attacks. Since SDNS are programmable through APIs it may be more vulnerable (Akhunzada, Gani, Amuar, Abdelaziz, Khan, Haya, Khan, 2016). If the controller is compromised information can be learned about network devices and used for more attacks, exploitations and reprogramming of the whole network (Akhunzada et al., 2016). "The southbound interface of an SDN can also easily be targeted with diverse denial of service and side channel attacks" (Akhunzada, Ahmed, Gani, Khan, Imran, Guizani, 2015, p. 36). Misconfiguration of SDN can have worse consequences than configuration errors in a traditional networks. SDN agents can be targeted to inject false flows (Akhunzada et al., 2015). Two main proposals have been put forth to secure SDNs. They are FRESCO and FortNox. FRESCO provides a secure programming framework for OpenFlow

networks. FortNox is a security enforcement kernel that is responsible for avoiding conflicts in rules that arise from different security authorizations (Akhunzada et al., 2015). The keys to secure SDN is to secure the SDN controller, protect the data flows, hardened the SDN agents (devices such as routers and switches) and hardening the API and communication between the controller and devices (Akhunzada et al., 2015).

In their paper, Contreras, Doolan, Lonsethagen and Lopez (2015), categorizes the challenges faced by SDNs and NFVs into three main categories; operational, organizational and business. Below is a summary of their findings:

- Operational
    - Network planning
        - Comprehensive network resource planning
        - Multilayer planning tools
    - Network deployment
        - Increased testing and product homologation
        - Risk of overinvestment in data center infrastructures
        - Criticality of data centers and need for highly secure infrastructures
    - Supportive systems
        - Integration with existing OSS/BSS systems
        - Relies on Open Source developments
    - Network operations
        - Standardization of open interfaces that facilitate uniform control and management across technologies and vendors

        - Control plane resiliency
        - Network automation
        - Decoupling of service from transport
    - Service provisioning
        - Mapping between service requirements and network capabilities
        - Common APIs and information and data models
    - Investment protections and migration
        - Coexistence with legacy networks
- Organizational
    - Departmental organization
        - Cross-technical and cross-functional reorganization of departments
    - Skills and know how
        - Multidisciplinary teams
    - Partnership ecosystem
        - Larger number of partners requiring more coordination and management effort
- Business
    - Analytics
        - Correlation of decoupled service and transport indicators
        - Big Data analytics for predictive actions
    - Customization
        - Standard interfaces for network service and resource consumptions
        - Isolation and security

        - Proper billing mechanisms
    - Network and IT equipment lifetime
        - Re-programmability of equipment to extend the service lifetime
    - Procurement
        - Management of a larger number of vendors
        - Definition of new quotation models to compare prices and solutions respect to conventional products
        - Definition of new guarantees
    - Capabilities for sharing network infrastructures
        - New business models for infrastructure and capacity sharing
        - Impacts of regulation
    - Innovation and experimentation
        - Creation and maintenance of innovative teams

(p. 212)

FUTURE

While the control plane can be programmed using API, this is no ability to program the data plane in most SDN iterations. FLARE is a Deeply Programmable Network (DPN) that strives to allow programming of the data and control plane and guarantees high performance and scalability (Farhady et al., 2015). This would allow for the enrichment of SDN applications. Currently SDN implementations like OpenFlow are tied to certain hardware. If SDN didn't have to depend on specific hardware or protocols it could be more productive and allow for more innovations. Moving forward developing SDN frameworks that rely on commodity hardware

would increase the use of SDN (Farhady et al., 2015). Network administrators are the ones that are using SDN APIs. In the future APIs should be developed so they can be used by end-users for on-demand services. An example would be an intrusion detection (ID) application that a user can configure to filter traffic from a specific source (Farhady et al., 2015). So far SDNs have been implemented in data centers and campus networks but researchers are starting to look at deploying SDNs on optical, home, wireless and cellular networks (Farhady et al., 2015). As SDNs are deployed into this environments new challenges will be introduced but also the world of new possibilities open up as well.

CONCLUSION

SDNs are becoming more popular and deployed into many different environments. They allow the connection between the packet-forwarding decisions and the packet-forwarding devices to be separated. The SDN controller knows the entire network topology and can make the decision about how to move the data through the network and then allow the devices to more the data. There are numerous benefits and challenges for SDN and has it continues to develop and be implemented, many more will arise. We have just begun to see the impact of SDNs on networks.

References

Akhunzada, A., Ahmed, E., Gani, A., Khan, M., Imran, M., & Guizani, S. (2015). Securing software defined networks: Taxonomy, requirements, and open issues. *IEEE Communications Magazine, 53*(4), 36-44. doi:10.1109/MCOM.2015.7081073.

Akhunzada, A., Gani, A., Anuar, N. B., Abdelaziz, A., Khan, M. K., Hayat, A., & Khan, S. U. (2016). Secure and dependable software defined networks. Journal of Network and Computer Applications, 61,199-221. doi:10.1016/j.jnca.2015.11.012.

Chen, J., Ma, Y., Kuo, H., Yang, C., & Hung, W. (2016). Software-defined network virtualization platform for enterprise network resource management. IEEE Transactions on Emerging Topics in Computing, 4(2), 179-186. doi:10.1109/TETC.2015.2478757.

Cisco. (2016). *CCNA_R&S_6.0_bridging_instructor_supplemental_material_mod4_cn* [PowerPoint slides]. Retrieved from https://www.netacad.com/group/resources/ccna-rs-bridge/6.0.

Contreras, L. M., Doolan, P., Lønsethagen, H., & López, D. R. (2015). Operational, organizational and business challenges for network operators in the context of SDN and NFV. Computer Networks, 92, 211-217. doi:10.1016/j.comnet.2015.07.016.

Farhady, H., Lee, H., & Nakao, A. (2015). Software-defined networking: A survey. Computer Networks, 81, 79-95. doi:10.1016/j.comnet.2015.02.014.

Hegde, S., Koolagudi, S. G., & Bhattacharya, S. (2015). Scalable and fair forwarding of elephant and mice traffic in software defined networks. Computer Networks, 92, 330-340. doi:10.1016/j.comnet.2015.09.014.

Jammal, M., Singh, T., Shami, A., Asal, R., & Li, Y. (2014). Software defined networking: State of the art and research challenges. Computer Networks, 72, 74-98. doi:10.1016/j.comnet.2014.07.004.

Li, W., Meng, W., & Kwok, L. F. (2016). A survey on OpenFlow-based software defined networks: Security challenges and countermeasures. Journal of Network and Compute Applications, 68, doi:10.1016/j.jnca.2016.04.011.

Li, Y., & Chen, M. (2015). Software-defined network function virtualization: A survey. IEEE Access, 3, 2542-2553. doi:10.1109/ACCESS.2015.2499271.

Masoudi, R., & Ghaffari, A. (2016). Software defined networks: A survey. Journal of Network and Computer Applications, 67, 1-25. doi:10.1016/j.jnca.2016.03.016.

Mukundha, C., Prabha, I. S., & Sreenu, K. (2016). Providing security in cloud based networks Through software defined networks. Journal of Theoretical and Applied Information Technology, 84(2), 287.

Wang, H., Li, Y., Jin, D., Hui, P., & Wu, J. (2016). Saving energy in partially deployed software defined networks. IEEE Transactions on Computers, 65(5), 1578-1592. doi:10.1109/TC.2015.2451662.