Software Defined Networking

Subjects: Engineering, Electrical & Electronic

Contributor: __ Imran

Software Defined Networking(SDN)has recently received much attention to address some of the enduring networking challenges. SDN's concept is based on ideas of generalization network hardware and decoupling the network controls software from the implementation devices . SDN is a new intelligent architecture for network programmability. SDN's main idea is to separate the control plane from the network devices, enabling the data control from a central and external software entity called an SDN controller. The Open Networking Foundation (ONF) is a non-profit consortium dedicated to the development, standardization, and commercialization of SDN for the transport and IP network layer.

SDN machine learning IoT SDN leveraging ML IoT leveraging SDN

1. Introduction

Figure 1 presents SDN layered architecture, consisting of [1] four planes: data plane, management plane, control plane and application plane. The Data plane presents a forwarding table that forwards the incoming packets to a network device. Management plane provides intelligent provisioning and orchestration systems for entire network management. The control plane is also termed as the brain of the networks that control different types of data planes using SBIs and protocols. The application plane defines the layer on different northbound applications that exist, which can help out SDN perform and solve future challenges of 5G, as this plane brings innovation, openness, and flexibility for the network vendors. SDN architecture provides dynamicity, network flexibility, and management capabilities. Research studies suggest SDN is the most reliable and promising for separating strategic network computation and data forwarding.

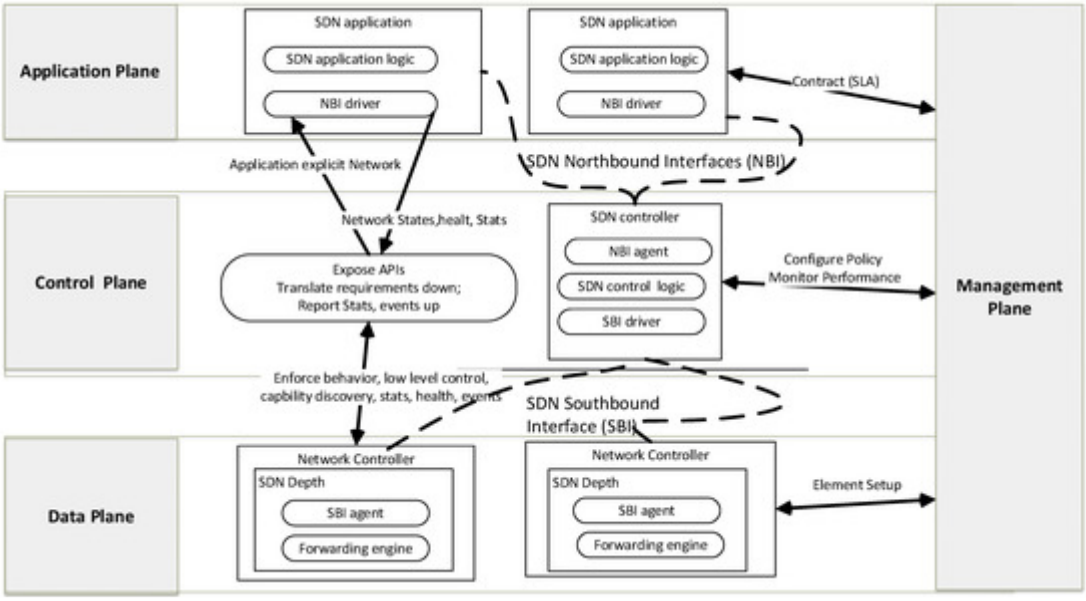

Figure 1. SDN Architecture.

Several northbound interfaces (NBIs) between the control plane and applications were introduced to provide high-level abstractions to the applications that reside on the control plane and in the form of various network-level services. OpenFlow [2] is a significant addition to the southbound interfaces (SBIs) that enable the network to be managed efficiently. Nevertheless, SDN can not be limited to OpenFlow as other less conventional protocols exist such as ForCES [3], NETCONF [4], OVSDB [5], Pollex [6], LISP [7], and OpenState [8]. OpenFlow deports all the intelligence to a centralized entity called SDN controller, which enables the separation of the control plane from the forwarding plane. There are three SDN functions in the panorama of OpenFlow, status reporting for each device connection, slicing, and flow-based forwarding. OpenFlow-based interconnection device matches the packets against a flow table inside the forwarding plane. FlowVisor controller is responsible for handling the flows, decisions, and publishing of the policies.

2. OpenFlow

OpenFlow provides an interface-based communication among the infrastructure and control layer based on open networking Foundation (ONF) [9][10]. Moreover, it provisions a way for controlling switches regardless of source code disclosure by the vendors. In summary, Openflow provides a way to directly access and manipulate the forward plan of switches and routers [11]. Furthermore, it grants access towards the flowable and provides instructions to the switches on managing and direct traffic of the network. It allows the network managers to alter the network flow in a short time span [12]. According to studies, there are two main categories of OpenFlow-based switches: OpenFlow-only and OpenFlow-hybrid. OpenFlow support is limited to OpenFlow operations, while the hybrid version supports operations and normal Ethernet switches [13].

The OpenFlow controllers are responsible for managing the OpenFlow switches based on a secure channel protocol called OpenFlow protocol. One or more flow tables are contained within a switch for performing a

forwarding operation and packet tracing. A flow table includes a flow entry; each entry possesses header fields, specific counters, and actions. The purpose of a header field is to match up against packets with information related to VLAN, ID, source and destination ports and IP address, etc. There are counters for keeping information about the number of packets their sizes. Action provides information associated to processing and matching packets; Their forwarding action is also specified, such as being sent to the controller or a port or sometimes dropped [14][15][16]. OpenFlow channel provides an interface for connecting the switches and controllers. Using this interface, the switches are being managed and configured by the controller; additionally, the events are received, and packets are sent through the switches. Various messages are sent using this channel, including asynchronous messages that involve messages for updating the controller regarding the network event and state change. The second type is controller-to-switch messages; these messages are for managing and inspection of switch states. Lastly, symmetric messages are initiated by the switch or the controller and are sent unsolicited [17][18][19]. OpenFlow controller is responsible for managing, allocating, and updating instructions and policies to the networking devices. It can decide how to handle packets with invalid flow entries and control the switch flow table. OpenFlow switch can establish communication with one or more controllers. A multiple-controller architecture can increase network reliability if a switch fails. OpenFlow starts operations when the switch is connected to all its configured controllers simultaneously, whereas related messages are only sent to the next switch [20][21][22][23].

3. Data Plane

The Data plane resides at the bottom layer of SDN architecture, consisting of forwarding devices such as routers and switches, whether virtual or physical. Virtual switches are software-based switches that run on a common operating system (OS), examples of virtual switches are Open vSwitch [24], Pantou [25], and Indigo [26]. Physical switches are hardware-based switches, implemented either on open network hardware such as NetFPGA [27], or on a merchant switch from networking hardware vendors. ServerSwitch [28] and switchBlade [29] are two examples of NetFPGA-based physical switches. Hardware vendors design their's merchant switches with the support of SDN protocols, for example virtual switches are more flexible and have complete feature support for SDN protocols. Physical switches have a high rate of flow forwarding as compared to virtual switches. Both are used to forward, drop, and modify packets using the control plane logic.

4. Control Plane

In SDN, the control plane acts as the brain that performs a set of actions; for example, it applies flow rules to handle the received ethernet frames that decide the traffic destination ports. SDN controller program network resources, control communication between applications and forwarding devices. SDN controller translates application plane requirements into policies and distributes these custom policies into forwarding devices. Control plane functionalities include network topology storage, configuration of devices, notification of state information, and shortest path routing. Some of the SDN controller architectures proposed in literature are NOX [30], Floodlight [31], POX [32], OpenDayLight [33], Ryu [34], and Beacon [35]. SDN controller has three interfaces for communication: southbound, eastbound westbound. Control data plane interfaces (CDPIs) are interfaces between the data and

control plane. CDPIs are also called SBIs, used for forwarding devices to exchange control policies and network state information with the control plane of an SDN controller. SBIs allow programmatic-based control of all device capabilities such as event notifications, advertisements, statistic reports, and forwarding operations. NBIs are exploited by applications to get an abstract view of the network to facilitate automation, analyze specific network behaviors, and analyze network requirements. Eastbound interfaces (EBIs) and westbound interfaces (WBIs) are used in the multi-controller SDN solutions. In multi-controller SDN networks, the exchange of information is important between controllers to provide a global network view to the applications. Examples of distributed control architectures are HyperFlow [36] and Onix [37]. EBIs and WBIs are private and cannot communicate with each other. SDN communication-interfaces [38], distributed-control plane (CIDC) [39], and east-west bridge [40] are some proposals for communication between different SDN controllers.

5. Application Plane

The top layer in the SDN is called the application plane, consisting of business applications. Business applications provide management and optimization of business services. These applications implement the control logic based on the network state information received from controller NBIs to modify the network behavior. In the study, [41] the solutions of SDN for traffic engineering are discussed. In paper [42], a security-based survey study is conducted. Yan et al. [43] proposed analysis of distributed denial of service (DDoS) attacks in SDN networks in a cloud computing environment. A survey on fault management issues in SDN and its solutions is presented in [44]. SDN is deployed in real-time scenarios including transport network, wireless networks [45][46], optical networks [47], wide area networks (WAN) [48], IoT, EC [49], and cloud computing [50].

6. P4 (Programming Protocol-Independent Packet Processors)

P4 is a programming language to access the hardware without the knowledge of the architecture of hardware. P4 is used to modify the packet-forwarding mechanisms of the SDN switches [51]. Initially, P4 was used to write software programs and program hardware switches. Hardware resources including network interface cards, networking appliances, FPGA, and ASIC. P4 is used to set the custom headers and dynamic parsing of headers from the packet [52]. P4 provides the custom match and action tables and other constructs such as counters, registers, etc. This makes the P4 language entirely protocol-free. If a definite protocol is used in the network, it is easy to reconstruct the P4 program for new header field maintenance. Other P4 features include configurations, making new P4 applications reusable if needed rather than purchasing new networking devices. Furthermore, the P4 is free from any target device specifications and characteristics. Nonetheless, P4 is dependent upon the design of a device. The P4 application written for a distinct architecture is deployable beyond all destination devices with the same architectural design [53]. The P4 program is specially created for the data planes layer; however, the destination device may hold both the control and data planes. P4 is also used in literature for defining the interfaces between the control and data plane partially, but it cannot manage the control plane's functionality.

References

1. Haleplidis, E.; Pentikousis, K.; Denazis, S.; Salim, J.H.; Meyer, D.; Koufopavlou, O. Software-defined networking (SDN): Layers and architecture terminology. Internet Res. Task Force (IRTF) RFC 7426 2015, 7426.
2. Alsaeedi, M.; Mohamad, M.M.; Al-Roubaiey, A.A. Toward adaptive and scalable OpenFlow-SDN flow control: A survey. IEEE Access. 2019, 7, 107346–107379.
3. Forwarding and Control Element Separation (ForCES) Protocol Specification. Available online: (accessed on 13 March 2021).
4. Popic, S.; Vuleta, M.; Cvjetkovic, P.; Todorović, B.M. Secure Topology Detection in Software-Defined Networking with Network Configuration Protocol and Link Layer Discovery Protocol. In Proceedings of the 2020 International Symposium on Industrial Electronics and Applications (INDEL), Banja Luka, Bosnia and Herzegovina, 4–6 November 2020.
5. The Open vSwitch Database Management Protocol. Available online: (accessed on 13 March 2021).
6. Asadollahi, S.; Goswami, B.; Sameer, M. Controller's scalability experiment on software defined networks. In Proceedings of the 2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), Bangalore, India, 1–2 February 2018.
7. The locator/ID Separation Protocol (LISP). Available online: (accessed on 13 March 2021).
8. Dargahi, T.; Caponi, A.; Ambrosin, M.; Bianchi, G.; Conti, M. A survey on the security of stateful SDN data planes. IEEE Commun. Surv. Tutor. 2017, 19, 1701–1725.
9. Bakhshi, T. State of the art and recent research advances in software defined networking. Wirel. Commun. Mob. Comput. 2017, 2017, 71916472017.
10. Dong, L.; Chen, L.; He, B.; Wang, W. The research on designs of multiple flow tables in the openflow protocol. In Proceedings of the 27th International Conference on Computer Communication and Networks, Hangzhou, China, 30 July–2 August 2018.
11. OpenFlow Version Roadmap. Tech. Rep. Available online: (accessed on 13 March 2021).
12. Mondal, A.; Misra, S.; Maity, I. AMOPE: Performance analysis of OpenFlow systems in software-defined networks. IEEE Syst. J. 2019, 14, 124–131.
13. Salih, M.A.; Cosmas, J.; Zhang, Y. OpenFlow 1.3 extension for OMNeT++. In Proceedings of the 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Liverpool, UK, 26–28 October 2015.

14. Kuźniar, M.; Perešíni, P.; Kostić, D.; Canini, M. Methodology, measurement and analysis of flow table update characteristics in hardware openflow switches. Comput. Networks 2018, 136, 22–36.
15. Zhao, B.; Zhao, J.; Wang, X.; Wolf, T. Ruletailor: Optimizing flow table updates in openflow switches with rule transformations. IEEE Trans. Netw. Serv. Manag. 2019, 16, 1581–1594.
16. Samociuk, D. Secure communication between OpenFlow switches and controllers. AFIN 2015, 39, 2015.
17. González, S.; De la Oliva, A.; Bernardos, C.J.; Contreras, L.M. Towards a resilient OpenFlow channel through MPTCP. In Proceedings of the 2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Valencia, Spain, 6–8 June 2018.
18. Kotani, D.; Okabe, Y. Fast failure detection of OpenFlow channels. In Proceedings of the AINTEC 15: Asian Internet Engineering Conference, Bangkok, Thailand, 18–20 November 2015.
19. Li, W.; Meng, W.; Kwok, L.F. A survey on OpenFlow-based Software Defined Networks: Security challenges and countermeasures. J. Netw. Comput. Appl. 2016, 68, 126–139.
20. Azzouni, A.; Braham, O.; Nguyen, T.M.T.; Pujolle, G.; Boutaba, R. Fingerprinting OpenFlow controllers: The first step to attack an SDN control plane. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016.
21. Gamess, E.; Tovar, D.; Cavadia, A. Design and implementation of a benchmarking tool for OpenFlow controllers. Int. J. Inf. Technol. Comput. Sci. 2018, 10, 1–13.
22. Darianian, M.; Williamson, C.; Haque, I. Experimental evaluation of two openflow controllers. In Proceedings of the 25th International Conference on Network Protocols, Toronto, ON, Canada, 10–13 October 2017.
23. Priya, A.V.; Radhika, N. Performance comparison of SDN OpenFlow controllers. Int. J. Comput. Aided Eng. Technol. 2019, 11, 467–479.
24. Open vSwitch, March. 2021. Available online: (accessed on 27 August 2020).
25. Pantou: OpenFlow 1.3 for OpenWRT,” March 2021. Available online: (accessed on 27 August 2020).
26. Yang, C.; Liu, J.; Chen, W.; Leu, F.; Chu, W.C. Implementation of a virtual switch monitoring system using OpenFlow on cloud. Int. J. Hoc Ubiquitous Comput. 2017, 24, 162–172.
27. Chu, T.-W.; Shen, C.; Wu, C. The hardware and software co-design of a configurable QoS for video streaming based on OpenFlow protocol and NetFPGA platform. Multimed. Tools Appl. 2018, 77, 9071–9091.
28. Zeng, T.; Wang, S.; Liu, S. Research on Intelligent Linkage Server Switch in Case of Power Loss in Computer Room. In Proceedings of the IEEE 11th International Conference on Software

Engineering and Service Science (ICSESS), Beijing, China, 16–18 October 2020; pp. 493–496.

29. Yan, J.; Jia, C.; Tang, L.; Li, T.; Lv, G.; Quan, W.; Yang, H. Network Programming Interface in General-Purpose Multi-core Processor: A Survey. In Proceedings of the 15th International Conference on Computer Science & Education (ICCSE), Delft, The Netherlands, 18–22 August 2020; pp. 675–680.
30. Badotra, S.; Panda, S.N. Evaluation and comparison of OpenDayLight and open networking operating system in software-defined networking. Clust. Comput. 2019, 23, 1–11.
31. Badotra, S.; Singh, J. Open Daylight as a Controller for Software Defined Networking. Int. J. Adv. Res. Comput. Sci. 2017, 8.
32. Pox. Available online: (accessed on 27 August 2020).
33. Chandramouli, M.; Clemm, A. Model-driven analytics in SDN networks. In Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, 8–12 July 2017; pp. 668–673.
34. Islam, M.T.; Islam, N.; Al Refat, M. Node to node performance evaluation through RYU SDN controller. Wirel. Pers. Commun. 2020, 112, 1–16.
35. Mamushiane, L.; Lysko, A.; Dlamini, S. A comparative evaluation of the performance of popular SDN controllers. In Proceedings of the Wireless Days, Dubai, United Arab Emirates, 3–5 April 2018; pp. 54–59.
36. Aly, W.H.F.; Al-anazi, A.M.A. Enhanced CONTROLLER Fault Tolerant (ECFT) model for software defined networking. In Proceedings of the Fifth International Conference on Software Defined Systems (SDS), Barcelona, Spain, 23–26 April 2018; pp. 217–222.
37. Paliwal, M.; Shrimankar, D.; Tembhurne, O. Controllers in SDN: A review report. IEEE Access. 2018, 6, 36256–36270.
38. Latif, Z.; Sharif, K.; Li, F.; Karim, M.M.; Biswas, S.; Wang, Y. A comprehensive survey of interface protocols for software defined networks. J. Netw. Comput. Appl. 2020, 156, 102563.
39. Benamrane, F.; Benaini, R. An East-West interface for distributed SDN control plane: Implementation and evaluation. Comput. Electr. Eng. 2017, 57, 162–175.
40. Sarmiento, D.E.; Lebre, A.; Nussbaum, L.; Chari, A. Decentralized SDN Control Plane for a Distributed Cloud-Edge Infrastructure: A Survey. IEEE Commun. Surv. Tutorials 2021, 23, 256–281.
41. Mendiola, A.; Astorga, J.; Jacob, E.; Higuero, M. A survey on the contributions of software-defined networking to traffic engineering. IEEE Commun. Surv. Tutor. 2016, 25, 918–953.

42. Ahmad, I.; Namal, S.; Ylianttila, M.; Gurtov, A. Security in software defined networks: A survey. IEEE Commun. Surv. Tutor. 2015, 18, 2317–2346.
43. Yan, Q.; Yu, F.R.; Gong, Q.; Li, J. Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges. IEEE Commun. Surv. Tutor. 2015, 18, 602–622.
44. Fonseca, P.C.; Mota, E.S. A survey on fault management in software-defined networks. IEEE Commun. Surv. Tutor. 2017, 19, 2284–2321.
45. Haque, I.T.; Abu-Ghazaleh, N. Wireless software defined networking: A survey and taxonomy. IEEE Commun. Surv. Tutor. 2016, 18, 2713–2737.
46. Chen, T.; Matinmikko, M.; Chen, X.; Zhou, X.; Ahokangas, P. Software defined mobile networks: Concept, survey, and research directions. IEEE Commun. Mag. 2015, 53, 126–133.
47. Thyagaturu, A.S.; Mercian, A.; McGarry, M.P.; Reisslein, M.; Kellerer, W. Software defined optical networks (SDONs): A comprehensive survey. IEEE Commun. Surv. Tutor. 2016, 18, 2738–2786.
48. Michel, O.; Keller, E. SDN in wide-area networks: A survey. In Proceedings of the 2017 Fourth International Conference on Software Defined Systems (SDS), Valencia, Spain, 8–11 May 2017; pp. 37–42.
49. Baktir, A.C.; Ozgovde, A.; Ersoy, C. How can edge computing benefit from software-defined networking: A survey, use cases, and future directions. IEEE Commun. Surv. Tutor. 2017, 19, 2359–2391.
50. Rafique, W.; Qi, L.; Yaqoob, I.; Imran, M.; Rasool, R.U.; Dou, W. Complementing IoT services through software defined networking and edge computing: A comprehensive survey. IEEE Commun. Surv. Tutorials. 2020, 22, 1761–1804.
51. Zolfaghari, H.; Rossi, D.; Nurmi, J. A custom processor for protocol-independent packet parsing. Microprocess. Microsys. 2020, 72, 102910.
52. Han, S.; Jang, S.; Choi, H.; Lee, H.; Pack, S. Virtualization in Programmable Data Plane: A Survey and Open Challenges. IEEE Open J. Commun. Soc. 2020, 1, 527–534.
53. Dang, H.T.; Wang, H.; Jepsen, T.; Brebner, G.; Kim, C.; Rexford, J.; Soulé, R.; Weatherspoon, H. Whippersnapper: A p4 language benchmark suite. In Proceedings of the Symposium on SDN Research, Santa Clara, CA, USA, 3–4 April 2017; pp. 95–101.