Software Defined Network: Future of Networking

Abhishek S Shetty (ENG20CS0007), Aravind Bhat (ENG20CS0038)

Department of Computer Science and engineering
Dayananda sagar university

Abstract—**The evolution of the Internet has sparked a digital revolution, ushering in an era where connectivity is ubiquitous and accessibility is paramount. However, the conventional IP network infrastructure remains fraught with complexity, posing challenges in management and reconfiguration. Enter Software Defined Networking (SDN), an innovative paradigm that decouples network control from underlying hardware, enabling centralized control and programmability. SDN, embodied by protocols like OpenFlow, abstracts network functionalities to simplify management and enhance flexibility. Leveraging REST API communication, SDN facilitates seamless interaction between controllers and applications, marking a significant advancement in information and communication technology.**

Index Terms—**Software defined networking, network virtualization, network operating systems, OpenFlow, REST API**

I. INTRODUCTION

In today's rapidly evolving technological landscape, the expansion of networks in both size and complexity presents a formidable challenge. As the demands on these networks grow, the traditional methods of managing hardware switches have become cumbersome and inefficient. The manual configuration of individual software switches is prone to errors, particularly within heavily virtualized environments and expansive networks. Additionally, the lack of automatic reconfiguration and response mechanisms in current IP networks further compounds the difficulty of enforcing necessary policies in such dynamic settings.

Compounding these challenges is the vertical integration prevalent in current network infrastructures, where the control plane and data plane are tightly integrated within networking devices. This integration limits flexibility and stifles innovation, hindering the evolution of networking infrastructure.

Enter Software-Defined Networking (SDN), a transformative approach poised to revolutionize the field of networking. SDN offers a solution to the limitations inherent in traditional network architectures by decoupling the control logic (control plane) from the traffic forwarders (data plane). This separation allows network switches to function purely as forwarding devices, with the control logic centralized in a logically centralized controller, often referred to as a network operating system. This paradigm shift simplifies policy enforcement, network reconfiguration, and evolution.

In this paper, we delve into the comparative analysis between traditional networking approaches and SDN in Section II, followed by a comprehensive literature survey in Section III. We provide an overview of SDN architecture and its components, shedding light on its potential to address the shortcomings of current network infrastructures. Additionally, we examine various simulation tools for SDN implementation, offering insights into the practical considerations of deploying SDN solutions.

II.TRADITIONAL v/s SDN

Traditional networking operates on a distributed model where each network device independently runs protocols like ARP, STP, OSPF, EIGRP, and BGP, leading to fragmented control without a centralized overview. Conversely, SDN adopts a centralized control plane model, with a dedicated SDN controller orchestrating communication between switches and feeding data plane information. SDN protocols such as OpenFlow, BGP, OVSDB, XMPP, NETCONF, and MPLS-TP facilitate this centralized control. This division between the control plane (decision-making) and data plane (packet forwarding) in SDN, as depicted in Figure 1, minimizes failures and enhances network flexibility, making it scriptable and configurable. Additionally, by segregating these processes and employing a dedicated server, SDN reduces traffic load, CPU burden, and memory usage, leading to more efficient routing decisions and network configuration.

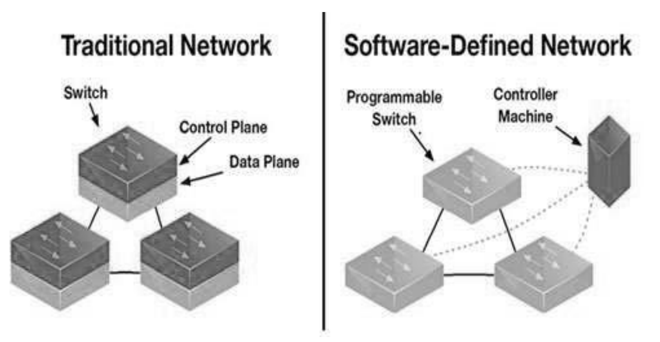

Figure 1: Traditional network v/s SDN

III. LITERATURE SURVEY

McKeown, N. (2011). How SDN will Shape Networking. In this seminal work, McKeown explores the transformative potential of Software-Defined Networking (SDN) in reshaping traditional networking paradigms. Through an insightful analysis, he elucidates the fundamental principles and implications of SDN, paving the way for further research and development in the field. The provided video presentation offers valuable insights into the conceptual framework and practical applications of SDN, serving as a cornerstone for understanding its profound impact on networking.

Schenker, S. (2011). The Future of Networking, and the Past of Protocols. Schenker's presentation delves into the evolution of networking technologies, tracing the trajectory from historical protocols to the emergence of SDN. By contextualizing SDN within the broader landscape of networking history, Schenker offers valuable perspectives on the challenges and opportunities inherent in this paradigm shift. The provided video presentation serves as a thought-provoking exploration of the past, present, and future of networking architectures.

Kim, H., & Feamster, N. (2013). Improving network management with software-defined networking. This article published in the IEEE Communications Magazine presents a comprehensive overview of how SDN can enhance network management practices. Kim and Feamster elucidate the key benefits and challenges of deploying SDN for network management tasks, providing valuable insights for researchers and practitioners alike. The scholarly analysis presented in this article contributes to the growing body of literature on SDN's practical implications for network management.

Molenaar, R. (2013-2017). Introduction to SDN (Software Defined Networking). Molenaar's online resource provides a comprehensive introduction to SDN, offering a blend of theoretical insights and practical examples. Through clear and concise explanations, Molenaar demystifies the complexities of SDN for a broad audience, making it an invaluable resource for individuals seeking to deepen their understanding of this transformative technology.

III. SOFTWARE-DEFINED NETWORKING ARCHITECTURE

In this section, we review two well-known SDN architectures, namely ForCES [1] and Openflow [2]. Both OpenFlow and ForCES follow the basic SDN principle of separation between the control and data planes; and both standardize information exchange between planes. However, they are technically very different in terms of design, architecture, forwarding model, and protocol interface. 1) ForCES: The approach proposed by the IETF ForCES (Forwarding and Control Element Separation) Working Group, redefines the network device's internal architecture having the control element separated from the forwarding element.

However, the network device is still represented as a single entity. The driving use case provided by the working group considers the desire to combine new forwarding hardware with third-party control within a single network device. Thus, the control and data planes are kept within close proximity (e.g., same box or room). In contrast, the control plane is ripped entirely from the network device in "OpenFlow-like" SDN systems. ForCES defines two logic entities called the Forwarding Element (FE) and the Control Element (CE), both of which implement the ForCES protocol to communicate. The FE is responsible for using the underlying hardware to provide per-packet handling.

The CE executes control and signaling functions and employs the ForCES protocol to instruct FEs on how to handle packets. The protocol works based on a masterslave model, where FEs are slaves and CEs are masters. An important building block of the ForCES architecture is the LFB (Logical Function Block). The LFB is a well-defined functional block residing on the FEs that is controlled by CEs via the ForCES protocol. The LFB enables the CEs to control the FEs' configuration and how FEs process packets.

ForCES has been undergoing standardization since 2003, and the working group has published a variety of documents including: an applicability statement, an architectural framework defining the entities and their interactions, a modeling language defining the logical functions within a forwarding element, and the protocol for communication between the control and forwarding elements within a network element. The working group is currently active.

2) OpenFlow: Driven by the SDN principle of decoupling the control and data forwarding planes, OpenFlow [2], like ForCES, standardizes information exchange between the two planes. In the OpenFlow architecture, illustrated in Figure 2, the forwarding device, or OpenFlow switch, contains one or more flow tables and an abstraction layer that securely communicates with a controller via OpenFlow protocol. Flow tables consist of flow entries, each of which determines how packets belonging to a flow will be processed and forwarded. Flow entries typically consist of: (1) match fields, or matching rules, used to match incoming packets; match fields may contain information found in the packet header, ingress port, and metadata;
(2) counters, used to collect statistics for the particular flow, such as number of received packets, number of bytes and duration of the flow;
(3) a set of instructions, or actions, to be applied upon a match; they dictate how to handle matching packets. Upon a packet arrival at an OpenFlow switch, packet header fields are extracted and matched against the matching fields portion of the flow table entries. If a matching entry is found, the switch applies the appropriate set of instructions, or actions, associated with the matched flow entry. If the flow table look-up procedure does not result on a match, the action taken by the switch will depend on the instructions

defined by the table-miss flow entry. Every flow table must contain a table-miss entry in order to handle table misses. This particular entry specifies a set of actions to be performed when no match is found for an incoming packet, such as dropping the packet, continue the matching process on the next flow table, or forward the packet to the controller over the OpenFlow channel. It is worth noting that from version 1.1 OpenFlow supports multiple tables and pipeline processing. Another possibility, in the case of hybrid switches, i.e., switches that have both OpenFlow– and non-OpenFlow ports, is to forward non-matching packets using regular IP

IV. TOOLS FOR IMPLEMENTATION

Several simulation tools such as the OMNET++ and Mininet have been developed to evaluate the performance of SDN. The other tools for simulation are Ns-3 and Estinet. These tools have their own features. The comparison between different simulations tools have been shown in Table 2 [13].

OMNET++ and Mininet are suitable for designing, building, and testing and provide practical feedback when developing real world systems. OMNET++ offers useful facilities for prototyping and simulating SDN application. Mininet is an emulation platform for the functional testing of OpenFlow protocol and SDN application. OMNET++ and NS-3 cannot support real controller. Mininet is less scalable compare to NS-3, Estinet and OMNET++. Estinet combine all the functionality and overcome the problem of real controller and emulation. Estinet is also capable for simulation to large number of switches. Estinet generates the accurate result and results are repeatable.

V. CONCLUSION AND FUTURE SCOPE

In conclusion, Software-Defined Networking (SDN) offers significant advantages over traditional networks, particularly in terms of scalability, flexibility, and ease of configuration. Its centralized architecture streamlines switching and routing operations, making it highly suitable for managing large networks with heavy traffic loads. However, security remains a major concern due to the centralized nature of SDN. To address this challenge, future research should focus on developing stronger security architectures to ensure the integrity and confidentiality of network data. Overall, while SDN holds immense promise for enhancing network performance, mitigating security risks will be crucial for its widespread adoption and continued success.

VI. REFERENCES

[1] N. Mckeown, “How SDN will Shape Networking”, October 2011. [Online]. Available: http://www.youtube.com/watch?v=c9-K5OqYgA.

[2] S. Schenker, “The Future of Networking, and the Past of Protocols”, October 2011. [Online]. Available : http://www.youtube.com/watch?v=YHeyuD89n1Y4

[3] H. Kim and N. Feamster, “Improving network management with software defined networking”, Communications Magazine, IEEE, vol. 51, no. 2, pp.114–119, 2013.

[4] Rene Molenaar, “Introduction to SDN (Software Defined Networking), 2013 - 2017 . [Online] . Available: https://networklessons.com/cisco/ccna routing switching-icnd2-200 105/introduction-to-sdn-software-defined-networking.