Effect of Conflicting Flows on TCP and UDP Data Transfer Rates in OpenFlow Switch for Software-Defined Networks (SDN)

MUTAZ HAMED HUSSIEN, MOHAMED KHALAFALLA HASSAN
Faculty of Engineering,
Future University,
Khartoum,
SUDAN

Abstract: - Software-defined networking (SDN) is a framework that enhances scalability and agility in network simplicity and control. It is characterized by logical centralization, improving control, and data planes. Nevertheless, additional investigation is needed to fully understand the influence of flow conflict on the transfer rate variable. The present research aims to analyze the impact of flow conflict on the efficacy of the two protocols, the transmission control protocol (TCP) and the user datagram protocol (UDP), respectively, by utilizing throughput as a measure of efficiency. The measurements are verified through SDN OpenFlow networking modeling using MININET. The findings reveal a significant average alteration in transfer rate for TCP and UDP when SDN conflict rules are present, ultimately impacting network and operational efficiency.

Key-Words: - Software defines the network, OpenFlow Table, conflict flow, transfer rate, TCP, and UDP.

Received: August 12, 2023. Revised: October 27, 2023. Accepted: November 29, 2023. Published: December 31, 2023.

1 Introduction

SDN is a networking infrastructure technology that makes networks more efficient, adaptable, and programmable, [1], [2]. This approach has resulted in numerous benefits. Administrators and systems engineers can apply modifications through a centralized management interface to address new business demands, [3]. Software-defined networking (SDN) significantly advances network agility by integrating specialized technological paradigms. The strategic decoupling of the network's control and data-forwarding elements is central to the SDN architecture. This separation facilitates the independent configuration of the network's primary controller, commonly referred to as the kernel, thus empowering network architects with comprehensive authority over the foundational structure of the network. SDN stands out in its ability to effectively govern and regulate network behavior, exhibiting remarkable reactivity and cost-efficiency, [4].

OpenFlow (OF) is a critical Software-Defined Networking (SDN) protocol. In the context of SDN, the controller is another crucial component of OpenFlow. The controller's primary role is to integrate platforms and enable the creation of applications using a northbound Application Programming Interface (API). OpenFlow facilitates the establishment of an interface among controllers and the forwarding plane, allowing a more effective and adaptive response to changing company demands, [5], [6]. The OpenFlow architecture of Software-Defined Networking (SDN) is depicted in Figure 1.

The OpenFlow protocol is the cornerstone for communication between OpenFlow switches and controllers. This protocol operates through flow tables embedded within the switching devices, which are instrumental in forging the connections. Utilizing these flow tables as a conduit, the OpenFlow protocol facilitates the establishment of interconnectivity. These tables play a pivotal role in ensuring that data transmissions are accurately conveyed, executed, and dispersed, adhering to the specific flow entries. This mechanism is critical in maintaining efficient and effective network operations, [7], [8].

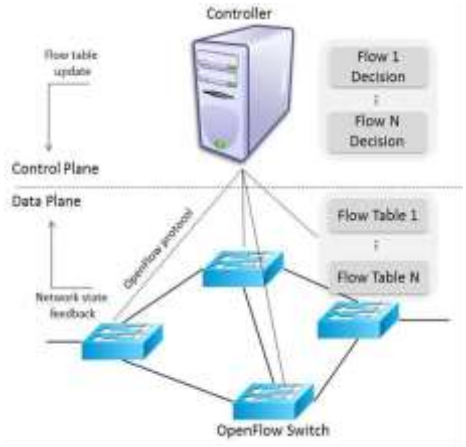

Fig. 1. OpenFlow SDN Architecture, [9]

2 Problem Formulation

Different types of flow conflicts can affect both traditional networks and software-defined networks (SDNs). making it difficult for networks to connect. These conflicts in SDNs can be categorized into two main types based on their adherence to SDN rules and their impact on networks, [10]. In the realm of Software-Defined Networking (SDN), flow entry conflicts can arise under many scenarios, notably during the establishment of an SDN network wherein a switch interfaces with several virtual networks. These conflicts may emerge whether the network is under the purview of a single controller or a group of controllers, leading to potential discrepancies in flow management. Additionally, a distinct category of flow conflict is observed when packets of ambiguous importance correspond with existing flow entries, further complicating the network's operational dynamics. These scenarios underscore the complex interplay of elements within SDN configurations, necessitating meticulous oversight for optimal network functionality, [11], [12], [13].

Flow entries in an open flow table can cause security module malfunctions. Correlating packets with their priorities helps, but interconnected flow entries can cause conflicts, [14], [15].

3 Problem Solution

TCP and UDP are commonly utilized protocols for sending and receiving packets and data between networked devices inside a network, [16]. Assessing network throughput is essential for improving performance. Consequently, numerous notable research has concentrated on analyzing and assessing the data transmission capacities of TCP and UDP protocols in Software-Defined Networks (SDN). These investigations have thoroughly analyzed the efficacy of TCP and UDP protocols in terms of transfer rates. In addition, they have utilized a specialized model to evaluate the effectiveness of these protocols when incorporated into an SDN framework.

This research is crucial in identifying the most efficient network architectures for SDNs, guaranteeing optimal data transmission and overall network performance, [10], [17].

The primary objective of this study is to assess and track the TCP and UDP throughput metrics in software-defined networking (SDN). The transfer rate is evaluated under two circumstances. The normal flows are operational in the network architecture in the first case. In the second case, additional rules are introduced to the OpenFlow switch to create conflicting flows that run parallel to the normal flow. The execution of this project involves the utilization of the Ryu controller, which employs the simple and fat tree topology network that has been built and tested within the Mininet software. Figure 2 depicts the suggested approach.

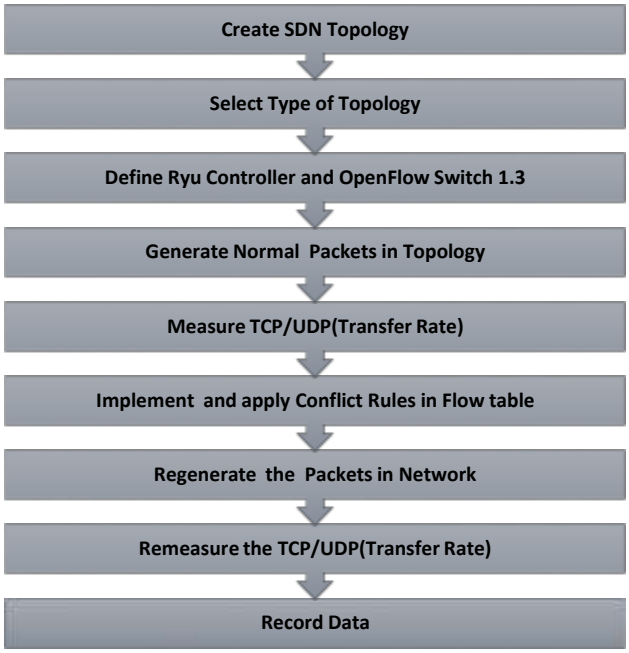

Fig. 2. Proposed Method Process

The initiation of the process begins with the establishment of a Software-Defined Network (SDN) within the Mininet environment. Following this, a specific network topology is selected, with the example here including both simple-tree and fat-tree topologies. The next critical step is the precise configuration of the controller and the OpenFlow switch. This stage involves the transmission of network packets and a detailed examination of the characteristics of TCP and UDP protocols. The process then progresses to the implementation of conflict rules in the OpenFlow switches. This leads to the retransmission of packets within the network, necessitating a reevaluation based on their TCP and UDP properties. After the data capture is completed, it undergoes rigorous analysis and further evaluation. Notably, the conflict rules are systematically applied to the OpenFlow switch across both topologies at varying intervals, ranging from 0 to 3600 seconds. This systematic approach allows for a comprehensive understanding of the network dynamics under different configurations and timeframes.

4 Methodology

This section provides a thorough explanation of configuring SDN platforms and creating simulations for research purposes. It also includes a complete description of the frameworks and methods utilized in the conducted tests.

4.1 Configuration Established for SDN Architecture

The objective of this investigation is to generate, gather, and safeguard contradictory and consistent data streams using a simulation software-defined networking (SDN) architecture. To accomplish this, we are using the Mininet framework, coupled with the Ryu controller and the Mininet simulator. The Mininet framework has VirtualBox installed, which is a virtual communication network. For a detailed overview of the simulation settings in the simulation environment, please refer to Table 1.

Table 1. System and environment specification for Mininet simulation

Software and Hardware	Specifications
Processor	Core i7
RAM	16 GB
Operation System	Ubuntu 18.04
Controller	Ryu
Programming	Python 2.7
OpenFlow Switch	Version 1.3

4.2 Model Development

The SDN module that forwards the simulation comprises two types of constraints. The initial configuration enables regular operation and functionality, while the subsequent configuration enforces conflict rules in the OpenFlow switch. OpenFlow is a protocol that allows the SDN controller to interact with the switch to determine the path of data packets. This SDN network is created within the Mininet framework, a versatile emulator capable of simulating intricate network environments comprising multiple hosts, switches, and controllers. Mininet is widely utilized for testing a variety of network applications, protocols, and topologies under diverse operational conditions. Customization of Mininet topologies is achieved through Python scripts, enabling the design of network structures tailored to specific simulation scenarios. The study in question delves into the topologies deployed within a Mininet network, particularly focusing on their connection to the Ryu controller. Ryu stands out as a component-based SDN controller, offering network administrators the flexibility to devise custom traffic management rules. The simple tree topology explored includes three switches and four hosts connected to the Ryu controller. In contrast, the fat-tree topology comprises a more complex arrangement of seven switches and eight hosts, all integrated with the Ryu controller. This comprehensive exploration provides valuable insights into the functionalities and adaptabilities of different network architectures within an SDN environment.

5 Result and Discussion

The results of two protocols, TCP and UDP, are presented in Figure 3 and Figure 4, respectively. The graphs show the throughputs of the two architectures used in this research. The numbers in the graphs are generated by simulating transmitted data collected over 3600 seconds, with measurements taken at 360-second intervals. The illustrations depict the usual flows in the primary tree topology and fat-tree topology using dark blue and dark brown lines. However, the introduction of conflict rules is represented by white blue, and yellow lines, which show the altered flows. The results indicate that the conflict rules have a negative impact on network forwarding transmission, resulting in a reduced transfer rate for conflict flows as compared to expected flows during all test durations.

Figure 3 compares TCP transfer rates in simple tree and fat-tree networks. The evaluation was done for 0-3600 seconds, and two cases were examined: with and without conflict rules. Without conflict rules, simple tree topology showed varied transfer rates from 600-771 Gb/s, and fat-tree had 564-647 Gb/s. With conflict rules, transfer rates decreased for both topologies. For simple tree topology, the results fluctuated between 478-673 Gb/s, and for the fat-tree topology, the results ranged from 213-488 Gb/s. During the 0-3600 seconds period, the TCP transfer rate decreased by 10% and 31% for simple tree and fat-tree topologies, respectively.

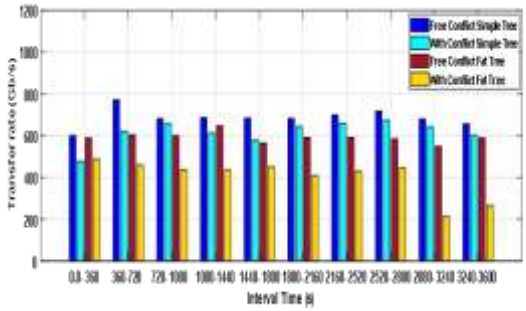

Fig. 3. TCP Transfer Rate for Two Topologies

Figure 4 compares UDP transfer rates for simple tree and fat-tree topologies with and without conflict rules. The bare tree had transfer rates of 19.1-20 Gb/s, while the fat tree had rates of 18.1-21 Gb/s. With conflict rules, both topologies experienced declines, with introductory tree rates at 17.1-18.2 Gb/s and fat-tree rates at 10.1-13 Gb/s. UDP transmission decreased by 11% for simple trees and 44% for fat trees over 0-3600 seconds.

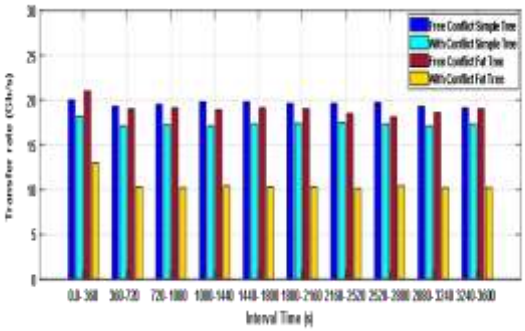

Fig. 4. UDP Ttransfer Rrate for Two Topologies

The evaluation of two scenarios, one employing the TCP protocol and the other utilizing the UDP protocol, is conducted by assessing the quantity of data transmitted. Figure 3 displays a decline in TCP transferred data within time intervals ranging from 0 to 3600 seconds. The simple tree topology shows a loss of 10%, while the fat-tree topology shows a drop of 31%. Similarly, Figure 4 compares UDP data transmission over extended time intervals ranging from 0 to 3600 seconds. The conflicting flows in UDP protocol also lead to a reduction in data transfer, specifically by 11% for a basic tree topology and 44% for a fat-tree topology. Lastly, Table 2 presents a comparison of the mean decrease in TCP and UDP protocols under both implemented conditions.

Table 2. Comparison of the average drop-in transfer rate with flow conflict.

Topology	TCP	UDP
Simple Tree	10%	11%
Fat Tree	31%	44%

6 Conclusion

This investigation focuses on evaluating the message transmission variable in Software-Defined Networking (SDN) for both regular and conflict flows in a standard forwarded SDN environment. To achieve this, we utilized the Mininet software simulator to execute an algorithmic model. Our findings demonstrate that the implementation of conflict rules significantly impacts data transport for both TCP and UDP protocols. Additionally, the data highlights an apparent difficulty with conflict flows in OpenFlow switches, suggesting that further research is necessary to address this issue by resolving the observed losses. One of the key takeaways from this research is that while implementing software-defined networking (SDN), it is essential to consider the issue of conflict flows, especially if the programs or services that are utilized demand an improvement in transfer rate. As such, subsequent investigators ought to emphasize the improvement of safety attributes in the SDN network while maintaining the network's quality of service.

References:

[1] Pisharody, S., et al., Brew: A security policy analysis framework for distributed SDN-based cloud environments. *IEEE Transactions on Dependable and Secure Computing*, 2017. 16(6): p. 1011-1025.

[2] Khairi, M.H.H., et al., Detection and classification of conflict flows in SDN using machine learning algorithms. *IEEE Access*, 2021. 9: p. 76024-76037.

[3] Cui, J., et al. Transaction-based flow rule conflict detection and resolution in SDN. *In 2018 27th International Conference on Computer Communication and Networks (ICCCN)*, 2018, IEEE.

[4] Khairi, M.H.H., et al., A Review of Flow Conflicts and Solutions in Software Defined Networks (SDN). *IIUM Engineering Journal*, 2021. 22(2): p. 178-187.

[5] McKeown, N., et al., OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 2008. 38(2): p. 69-74.

[6] Adeniji Oluwashola David, O.I.O., "Scalable Flow based Management Scheme in Software Define Network (SDN) using sFlow," *WSEAS Transactions on Computers*, vol. 22, pp. 64-69, 2023, https://doi.org/10.37394/23205.2023.22.7.

[7] Lo, C.-C., P.-Y. Wu, and Y.-H. Kuo. Flow entry conflict detection scheme for software-defined network. *In 2015 International Telecommunication Networks and Applications Conference (ITNAC)*. 2015. IEEE.

[8] Hauser, F., et al., P4-MACsec: Dynamic topology monitoring and data layer protection with MACsec in *P4-SDN*. arXiv preprint arXiv:1904.07088, 2019.

[9] Khairi, M.H.H., et al., et al., A Review of Anomaly Detection Techniques and Distributed Denial of Service (DDoS) on Software Defined Network (SDN). *Engineering, Technology & Applied Science Research*, 2018. 8(2), p.2724-2730.

[10] Wang, M.-H., et al., SDUDP: A reliable UDP-Based transmission protocol over SDN. *IEEE Access*, 2017. 5, p.5904-5916.

[11] Khairi, M.H., et al., Generation and collection of data for normal and conflicting flows in software defined network flow table. *Indonesian J. Electr. Eng. Comput. Sci*., 2021. 22(1), p.307.

[12] Zaw, H.T. and A. Maw, Traffic management with elephant flow detection in software defined networks (SDN). *International Journal of Electrical and Computer Engineering*, 2019, 9(4), p.3203.

[13] Wael Hosny Fouad Aly, H.K., Nour Mostafa, Samer Alabed, "Towards Securing OpenFlow Controllers for SDNs using ARMA Models," *International Journal of Applied Mathematics, Computational Science and Systems Engineering*, vol. 4, pp. 21-29, 2022, DOI: 10.37394/232026.2022.4.3.

[14] Ali, T.E., A.H. Morad, and M.A. Abdala, Traffic management inside software-defined data center networking. *Bulletin of Electrical Engineering and Informatics*, 2020, 9(5), p.2045-2054.

[15] Monika, P., R.M. Negara, and D.D. Sanjoyo, Performance analysis of software defined network using intent monitor and reroute method on ONOS controller. *Bulletin of Electrical Engineering and Informatics*, 2020, 9(5), p.2065-2073.

[16] Khairi, H., et al., The impact of firewall on TCP and UDP throughput in an OpenFlow software defined network. *Indonesian Journal of Electrical Engineering and Computer Science*, 2020, 20(1), p.256-263.

[17] Gu, Y. and R.L. Grossman, UDT: UDP-based data transfer for high-speed wide area networks. *Computer Networks*, 2007, 51(7), p.1777-1799.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

Mutaz Hamed Hussien, Mohamed Khalafalla, carried out the simulation, and the optimization Manuscript writing.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare.