Software Defined Network Adoption Decision Simulator

Qixuan Cao, Stanislav Kalynych, Ke Li, and Jinzhuo Tang
Electrical and Computer Engineering
University of Toronto
Toronto, Canada
damian.li@mail.utoronto.ca

***Abstract*—In this project, we design a simulation tool that facilitates the performance comparison of Software-Defined Networking (SDN) and traditional networks. With a brief background description followed by a more thorough literature review, we identify the current technology gap in SDN application and then describe the scope of our project. We would reach a decision on the necessary tools and knowledge required for an empirical cloud computing implementation and develop a Data Flow Diagram (DFD) Context Diagram for this business proposal.**

I. Background

In the domain of network management, Software-Defined Networking (SDN) offers highly programmable control over network resources, facilitating more flexible and efficient network configurations. The core idea of SDN is to separate the network control layer (decision layer) from the data forwarding layer (execution layer) to make the network management more flexible and centralized [1]. This architecture allows administrators to programmatically dynamically adjust network behavior to changing needs and conditions. However, SDN is not suitable for all scenarios, and its performance compared to traditional network architectures under various use cases remains a topic worth exploring. Enterprises and educational institutions face challenges in deciding whether to migrate to SDN, as they need to evaluate the potential benefits of SDN based on their specific requirements and conditions.

II. Literature Review

Over the past few years, Software Defined Network(SDN) has become an emerging solution to many of the existing network issues [2]. Since our business proposal centers on the performance evaluation regarding SDN in different environments, we would briefly examine existing studies on the performance aspect of SDN.

In [2], Kruetz et al offers a comprehensive survey of the SDN concepts and existing frameworks. The paper notes that traditional IP networks have highly decentralized structure with closely coupled data and control planes, but are quite effective in terms of performance. Though recent efforts have been made by large companies to enable data plane programmability, e.g. Cisco's OpenFlow, most of the SDN implementations trade performance for network consistency(which is critical for integrating distributed switches, e.g. HyperFlow based on OpenFlow, as part of the SDN framework). However, the authors also noted that the performance deterioration could be recovered partially through specialized optimization algorithms and diversified southbound configurations on southbound interfaces.

In [3], the authors present performance as a key challenge for SDN: it is greatly affected by the architecture of the Controller, which serves as a "brain" of the SDN. Although the separation of data and control planes theoretically improves the global network performance, the structure is subject to scalability concerns when maintaining the necessary global visibility of traffic and security vulnerabilities, especially DoS attacks. However, there are existing frameworks aiming for mitigating performance bottlenecks in large-scale networks, for example VeriFlow [4].

In 2013, Gelberger, Yemini and Giladi conducted a detailed experiment on the performance evaluation of two renowned SDN architectures, OpenFlow and ProGFE. [5, 6, 7]. Evaluated on three main aspects, latency, delay jitter and throughput, the authors observed that the complexity of the SDN architecture has no influence on the performance of the SDN: ProGFE, the more complex architecture, has a higher throughput compared to OpenFlow. The SDN flexibility, however, comes at the expense of raw performance regardless of the architecture chosen.

These previous literature help us establish the key factors to be considered when evaluating network performance offered by SDN and traditional IP network, as well as proving the potential gap that could be filled with our tool: there is no previous work for helping the actual developers make decision on the adoption of SDN based on their own tailored scenarios.

III. Project Scope

A. Current Gap

As we have examined in existing literature, the current performance benchmarking done on various SDN architectures lacks the capabilities to fit in a case-by-case scenarios based on users needs. When users set up network environment for their individual use cases, regardless of business or school, they need specific performance data targeted at their specifications rather than generic conclusions provided by existing literature.

This project aims to develop a cloud-based service for simulating the performance comparison between SDN and traditional networks in user specified environments. The service will consider various factors, including the number of users, daily data transmission volume, and network topology, to provide a comprehensive performance assessment. Our goal is to offer users an intuitive comparison to assist them in making informed decisions about migrating to SDN based on their needs and conditions. The following sections details the project scope.

B. Functions

The simulation tool should provide the following for end users:

1) The user should be able to specify parameters to establish a proper topology of their own network environment. The set of parameters exposed to users are: number of hierarchies, number of devices, node fan-outs, estimated regular traffic load, and estimated payload of each packets (subject to change when project implementation details are settled).
2) The simulation tool should be able to simulate network traffic the user specified topology with SDN as the network controlling model.
3) The simulation tool should be able to simulate network traffic the user specified topology with traditional decentralized router as the network controlling model.
4) The user should be able to view performance results from the simulation. The following metrics would be relayed back to user: throughput, latency, average delay, and packet loss rate.

C. Objectives

The simulation tool should provide the following optimizations for end users:

1) The tool should have as low runtime complexity as possible: the user should not wait indefinitely for the performance result, even with large network topology.
2) The tool should require minimal system modification on users' end: the user should not install any libraries locally.
3) The tool should provide fairness across all users: multiple simulation requests from different users would line up in a queue.

D. Limitations

While our service aims to provide as accurate a performance comparison as possible, the complexity of real network environments (such as computing power, storage space, network bandwidth) may result in discrepancies between simulation results and actual performance in some cases. Moreover, the effectiveness of the service may be limited by the availability of cloud computing resources and the accuracy of simulation technologies.

IV. Implementation Details

A. Establishing Simulated Environment

To simulate SDN and measure its performance in a cloud environment, we plan to utilize the Mininet emulation tool [8] and the JPerf network performance measurement utility [9].

1) Network Topology Simulation: Our approach involves leveraging cloud-based servers to simulate a vast number of switches and hosts using Mininet [8]. This setup allows us to create a virtual network environment that mirrors the complexity and scale of a data center's network infrastructure. By conducting our simulations on the cloud, we benefit from scalable computing resources that can accommodate the high number of network elements we wish to emulate.

2) Performance Metrics Extraction: For performance analysis, we will use JPerf to record and visualize the performance metrics of our SDN solutions. JPerf offers valuable insights into various performance aspects, such as bandwidth, latency, and packet loss [9]. The collected data will be crucial for understanding the effectiveness of our SDN configurations and for making informed decisions about real-world deployments of SDN.

B. Design Flow

The SDN Data Flow Diagram as shown in figure 1 depicts a hierarchical structure of an SDN architecture, showing the flow of control and data among the planes:

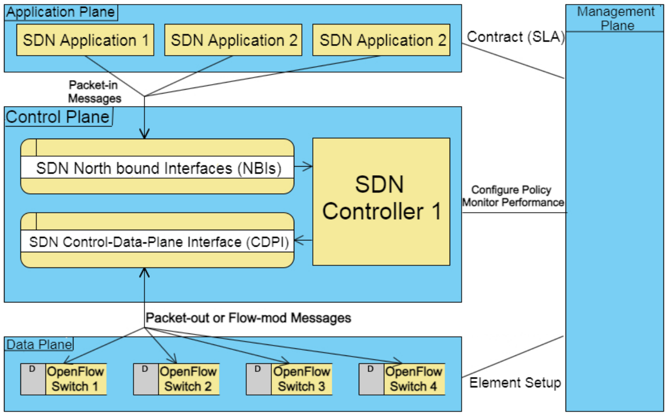

Fig. 1: SDN Data Flow Diagram

- **Arrows**: The arrows between the different planes and components illustrate the direction of the message flow, showing how policies and configurations are set at the management level, then translated into flow rules by the applications and controller, and finally executed by the switches at the data plane.
- **Application Plane**: consisting of three SDN applications, the application plane interacts with the control plane by sending ”Packet-in” messages. These applications can represent various network functions such as load balancing, firewall policies, or network monitoring.
- **Control Plane**: the control plane is responsible for translating the requirements from the application layer into flow rules that can be applied to the data plane, with the SDN Controller (Controller 1) as the brain of the network. The controller communicates with applications via the SDN Northbound Interfaces (NBIs) and with the switches via the SDN Control-Data-Plane Interface (CDPI).
- **Data Plane**: The bottom layer shows four OpenFlow switches labeled Switch 1 to Switch 4, which handle the actual forwarding of packets based on the flow rules provided by the control plane. They communicate back to the controller with ”Packet-out” or ”Flow-mod” messages, indicating the actions taken on the packets or changes to the flow tables.

REFERENCES

[1] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, “Software-defined networking (sdn): a survey,” *Security and Communication Networks*, vol. 9, no. 18, pp. 5803–5833, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1737

[2] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[3] K. Nisar, E. R. Jimson, M. H. A. Hijazi, I. Welch, R. Hassan, A. H. M. Aman, A. H. Sodhro, S. Pirbhulal, and S. Khan, “A survey on the architecture, application, and security of software defined networking: Challenges and open issues,” *Internet of Things*, vol. 12, p. 100289, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S25426605203

[4] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey, “Veriflow: verifying network-wide invariants in real time,” in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN ’12. New York, NY, USA: Association for Computing Machinery, 2012, p. 49–54. [Online]. Available: https://doi.org/10.1145/2342441.2342452

[5] A. Gelberger, N. Yemini, and R. Giladi, “Performance analysis of software-defined networking (sdn),” in *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, 2013, pp. 389–393.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” vol. 38, no. 2, p. 69–74, mar 2008. [Online]. Available: https://doi.org/10.1145/1355734.1355746

[7] R. Giladi and N. Yemini, “A programmable, generic forwarding element approach for dynamic network functionality,” in *Proceedings of the 2nd ACM SIGCOMM Workshop on Programmable Routers for Extensible Services of Tomorrow*, ser. PRESTO ’09. New York, NY, USA: Association for Computing Machinery, 2009, p. 19–24. [Online]. Available: https://doi.org/10.1145/1592631.1592637

[8] “Mininet an instant virtual network on your laptop (or other pc),” https://mininet.org/, accessed: 2024-02-09.

[9] “Jperf 2.0.2,” https://code.google.com/archive/p/xjperf/, accessed: 2024-02-09.