Implications of Network Resources and Topologies Over SDN System Performances

Bojan Velevski[1], Valentin Rakovic[2(✉)], and Liljana Gavrilovska[2]

[1] Makedonski Telekom, Skopje, Macedonia
bojan.velevski@telekom.mk
[2] Ss. Cyril and Methodius University in Skopje,
Rugjer Boskovic 18, Skopje, Macedonia
{valenitn,liljana}@feit.ukim.edu.mk

Abstract. The next generation communication networks are envisioned to be flexible, scalable, reliable and secure. Software Defined Networking introduces mechanisms to build and orchestrate efficient, flexible and adaptable networks. This is achieved by the separation of software and hardware functionalities in the network devices. This paper evaluates the overall Software Defined Networking performances with respect to the allocated network resources, i.e. different CPU's and bandwidth's allocations. The performance results clearly show that the overall system performances can be highly influenced by the network setup and chosen topology.

Keywords: SDN · Mininet · Network resources · CPU · Bandwith

1 Introduction

During the last decade, the rapid development of the communication technologies has resulted in significant increase of novel services and user's data demands. This has opened new challenges and issues that cannot be addressed by the conventional communications networks. For example, conventional IP based core networks are experiencing significant difficulties regarding manual configuration, management and optimization of network resources, enforcement of security policies - activities dependable on vendor OS upgrade, patch, release.

Recently, Software Defined Networking (SDN) [1] has drawn significant attention from both academia and industry as an auspicious technology capable to solve these issues. The core concept behind SDN is to separate the control plane from the forwarding plane, i.e. to separate the software from the hardware functionalities of the networking infrastructure, i.e. devices. This aspect facilitates on-the-fly and dynamic reconfiguration of the network and its underlying resources, such as *CPU power*, *link bandwidth*, *routing protocols*, etc. This reconfiguration is orchestrated by the SDN controller, which represents the Networking Operating System in the considered architecture. Currently, OpenFlow [2] is the most promising and exploited protocol that leverages the communication between the SDN controller and the networking devices. SDNs can leverage rapid deployment of new businesses, as well as swift and effective research and development.

O. Fratu et al. (Eds.): FABULOUS 2017, LNICST 241, pp. 25–31, 2018.
https://doi.org/10.1007/978-3-319-92213-3_4

The industry and academia commonly exploit simulators, emulation platforms and prototype testbed, for developing and testing novel SDN related features. Testbeds offer the most accurate analysis for SDN experiments. However, the number of SDN related research activities performed on testbeds is scarce and limited [3–5]. Most of the current research is performed by exploiting the network emulator Mininet [6]. The research works commonly focus on SDN performance analysis based on different types of networking aspects, such as SDN controller algorithm and advantages over conventional networks [7, 8] routing protocols [9, 10], network anomaly detection and recovery [11] resource allocation [12], etc.

However, none of the ongoing research works, have evaluated the impact of the network resources and topology on the overall system performances. This paper specifically analyses different network resource allocations for different network topologies and scrutinizes their effect over the overall SDN performance.

The paper is structured as follows. Section 2 presents the generic scenario setup. It also elaborates on the SDN topologies used for the performance analysis. Section 3 provides the performance evaluation and analysis. Section 4 concludes the paper.

2 Scenario Setup and SDN Topologies

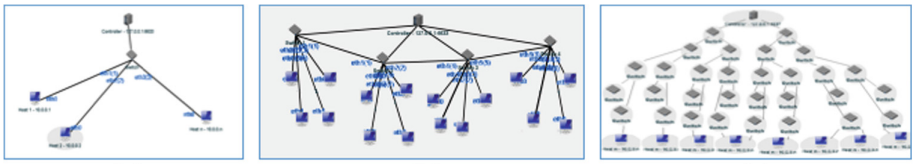
This section presents the scenario setup and the relevant SDN topologies used for the performance analysis. The main goal of the paper is to evaluate different SDN setups with respect to the allocated network resources, like *CPU* and *link bandwidth*, for a set of network topologies and analyze their impact on the overall system performance. The performance metrics of interest are, achieved *network throughput* and the *end-to-end jitter*. These metrics are chosen, since they clearly reflect the network capabilities and are crucial QoS parameters for *non-real-time* and *real-time* applications.

All observed SDN topologies are *Open-Flow* based that are designed in the Mininet emulator [6]. Mininet, by default supports a plethora of built-in network topologies which can be implemented using the available Mininet command-line options. Every SDN topology consists of a SDN controller, SDN switches and hosts. The paper, specifically focuses on three distinct SDN topologies, i.e. *Single*, *Linear* and *Tree* topology.

Single Topology. The topology is consisted of single Open-Flow (OF) switch connected with multiple hosts, Fig. 1a.

Linear Topology. The topology is consisted of predefined number of hosts and switches, where each host is connected to a specific switch. All switches serve the same number of hosts and are interconnected between each other on the same network depth, Fig. 1b.

Tree Topology. The topology is consisted of hosts and switches arranged in a tree fashion, where the hosts are always located at the end of the topology, i.e. leafs. The branches in the topology interconnect multiple switches and hosts according to the specific topological design. The network depth parameter is utilized to reflect the number of branches i.e. network levels, Fig. 1c.



(a) (b) (c)

Fig. 1. SDN topologies: Single (a), Linear (b), Tree (c)

For the purpose of the performance analysis and comparison, all three topologies are assumed to incorporate the same number of hosts. Specifically, the network design and topological complexity is presented in Table 1.

Table 1. Parameters of the proposed OF topologies.

	Single topology	Linear topology	Tree topology
Number of OF controllers	1	1	1
Number of OF switches	1	4	31
Number of hosts	32	32	32

All of the SDN topologies utilize the same controller. The implemented controller is the Mininet built-in one, which employs L2 i.e. hub-like operation to all SDN switches.

3 Performance Analysis

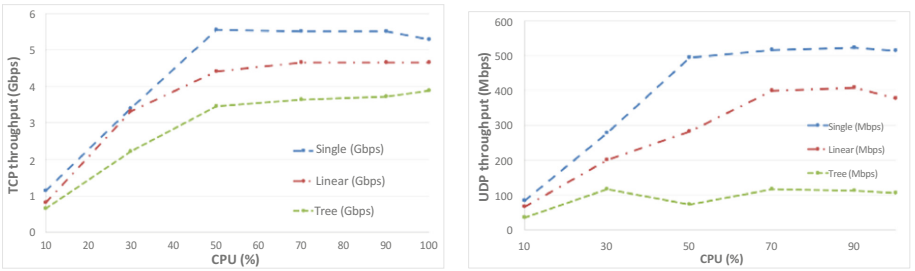
This section, provides the performance analysis of the different network configurations presented in the previous section. The analysis is performed for both TCP and UDP flows, using the "iperf" tool and it is averaged over all of the different hosts in the network. All network topologies incorporate the same number of hosts, Table 1. The network configuration parameters of interest are given in Table 2.

Table 2. Network configuration parameters.

Parameters	Values
CPU (%)	10, 30, 50, 70, 90, 100
Bandwidth (Mbps)	10, 30, 50, 70, 90, 100

Figure 2a depicts the achieved network throughput versus the allocated CPU to network elements when streaming the TCP traffic. It is evident that the achieved network throughput increases when allocating lager portions of the available CPU resources. However, for the CPU allocation higher than 50%, the network throughput converges to a specific value for all three topologies. This is a result of the underlying

transmission capabilities of the network. Different link bandwidths would result in different behavior of the achieved throughput. Specifically, higher link bandwidths will induce a throughput convergence for higher CPU values. It is also evident that the *single* topology achieves the highest throughput, whereas the tree topology achieves the lowest throughput.



(a) (b)

Fig. 2. Achieved network throughput vs CPU load: TCP traffic (a), UDP traffic (b), (Link Bandwidth = 100 Mbps)

Figure 2b depicts the achieved network throughput versus the allocated CPU to the network elements when streaming the UDP traffic. Similar conclusions hold as for Fig. 2a, i.e. the system throughput increases when allocating lager portions of the available CPU resources. It is also evident that the *single* topology achieves the highest throughput, and the three topology achieves the lowest throughput.

Figure 3 depicts the end-to-end jitter between two hosts in the network versus the allocated CPU to the network elements. It is evident that the allocated CPU resources have no impact on jitter for the single and linear topologies. However, for the tree topology the CPU allocation significantly impacts the network performances, due to its underlying complexity.

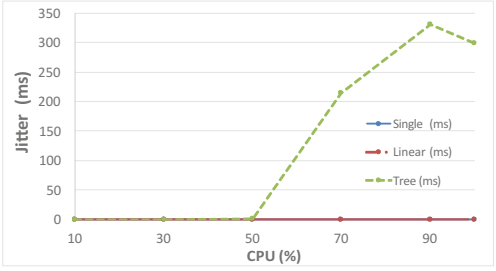
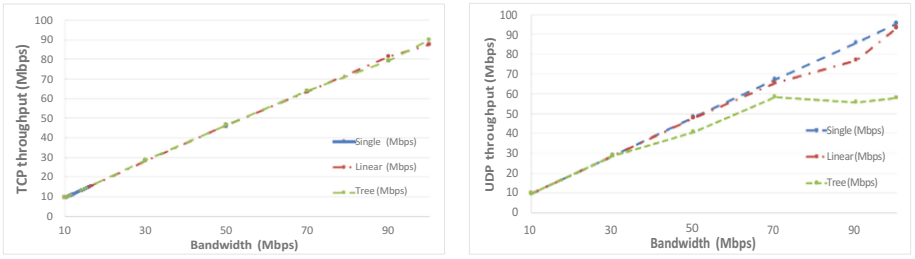


Fig. 3. End-to-end jitter vs CPU load (Link Bandwidth = 100 Mbps)

Figure 4a depicts the achieved network throughput versus the allocated bandwidth when streaming the TCP traffic. It is evident that the network throughput increases proportionally when increasing the allocated bandwidth. All three topologies have almost identical behavior.



(a) (b)

Fig. 4. Network throughput vs allocated bandwidth: TCP traffic (a), UDP traffic (b), (CPU Load = 50%)

Figure 4b depicts the achieved network throughput versus the allocated bandwidth when streaming the UDP traffic. It is also evident that for UDP traffic that increasing the bandwidth increases the overall system throughput. Similarly, to the previous figures, the single topology achieves the best performances, whereas the tree topology achieves the worst performances. It is also evident form the figure that the underlying transport protocol influences the system performances. Specifically, UDP results in lower achieved throughputs compared to TCP.

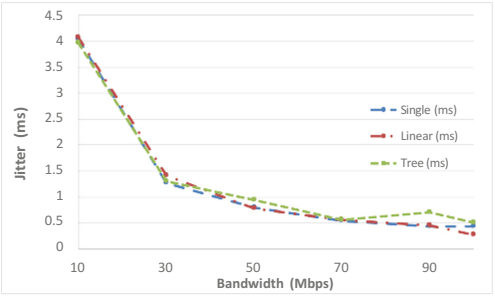


Fig. 5. End-to-end jitter vs allocated bandwidth (CPU Load = 70%)

Figure 5 depicts the end-to-end jitter between two hosts in the network versus allocated bandwidth on network links. It is evident that higher bandwidth allocations result in decreased end-to-end jitter in the network. The tree topology introduces the highest levels of jitter of all evaluated topologies, due to the high complexity. Specifically, the three topology introduces the highest number of hops, of all presented topologies, resulting in increased routing and processing delay. This delay significantly impacts the system performance, specially the end-to-end delay.

This section analyzes the SDN system performances for different network resource configurations, i.e. for differed CPU and bandwidth allocations for three different network topologies. The results clearly show that the network resources such as the CPU and the bandwidth as well as the network topology can play crucial role for achieving the optimal system performances.

4 Conclusions

Software Defined Networking represents a novel and auspicious technology that will pave the road for the next generation of communication networks. Most of the ongoing SDN related research is performed by exploiting the network emulator Mininet, as an accurate and low-cost option for evaluating and analyzing variety of networking related aspects.

This paper analyses the system performance of SDN networks with respect to the underlying network resources, i.e. for different CPU and bandwidth allocations at the networking elements. The Mininet based evaluation is performed for three different network topologies, (i.e. simple, linear and tree topology). The results show that the underlying network resources can have significant impact on the attained system performances. Specifically, higher bandwidth allocations facilitate improved system throughput and decreased end-to-end jitter for all types of networking topologies. Higher CPU allocation also results in significantly improved system throughput for all network topologies. However, the CPU allocation impacts the jitter performances only in very complex network topologies, i.e. network topologies with multiple depth layers. Future work will focus on evaluating the impact of additional network resources and topologies, as well as different types of SDN controllers.

References

1. Jackson, C.R., et al.: Demonstration of the benefits of sdn technology for all-optical data centre virtualisation. In: Optical Fiber Communication Conference 2017, Los Angeles, California United States (2017)
2. Robertazzi, T.: Introduction to Computer Networking: Software-Defined Networking. Springer, Germany (2017). https://doi.org/10.1007/978-3-319-53103-8
3. Chen, Q., et al.: An integrated framework for software defined networking, caching and computing. In: IEEE Network (2016)
4. Huo, R., et al.: Software defined networking, caching, and computing for green wireless networks. IEEE Commun. Mag. **11**, 185–193 (2016)
5. Huang, T., et al.: A survey on large-scale software defined networking (SDN) testbeds: approaches and challenges. IEEE Commun. Surv. Tutorials **12**(4), 531–550 (2017)
6. Mininet network emulator. http://mininet.org
7. Bholebawa, I., et al.: Performance analysis of proposed openflow-based network architecture using mininet. Wirel. Pers. Commun. **86**, 943–958 (2016)
8. Shivayogimath, C., et al.: Performance analysis of a software defined network using mininet. In: Artificial Intelligence and Evolutionary Computations in Engineering Systems, pp. 391–398 (2016)

9. Barrett, R., et al.: Dynamic traffic diversion in SDN: testbed vs mininet. In: Proceedings of 2017 International Conference on Computing, Networking and Communications (ICNC): Network Algorithms and Performance Evaluation. Silicon Valley, USA (2017)
10. Chai, R., et al.: Energy consumption optimization-based joint route selection and flow allocation algorithm for software-defined networking. Sci. China Inf. Sci. **60**(4), 040306 (2017)
11. Mauthe, A., et al.: Disaster-resilient communication networks: principles and best practices. In: Proceedings of 8th International Workshop on Resilient Networks Design and Modeling (2016)
12. Cao, B., et al.: Resource allocation in software defined wireless networks. IEEE Netw. **31**, 44–51 (2017)