ON USING MININET TO EVALUATE QOS FOR NODE.JS BACK ENDS

Juan Flores, Oscar Sandford, and Ben Wunderlich

University of Victoria

ABSTRACT

Write this last.

***Index Terms*—** Quality of Service, Software Defined Network, Network emulation, Mininet, Node.js

1. INTRODUCTION

The design of computer networks is driven by a drive towards a target quality of service (QoS), a set of performance measurements. Further, the implementation of network architectures demands time and money, and implementations that fail to meet quality of service standards waste time for both companies and customers. Software-defined networks (SDN) addresses the project of static architectures by enabling programming and measurement in a dynamic setting. It is important to understand and harness the strengths of SDN in a methodical way, so developers can focus more on the design goals and less on tedious implementation tasks.

There does not yet exist a qualitative evaluation of such measurement techniques for modern web servers such as Node.js. Providing a survey of various QoS measurement metrics for this domain will enable web developers to better understand and optimize the QoS metrics that matter most to their application domain.

2. RELATED WORK

Previous studies have defined quality of services measurements [1] and used network emulation tools to program their own experiments [2]. The metrics presented in [1] are throughput, delay, packet loss, and jitter. The authors used these metrics in a generic setting (i.e. no specific domain) in order to compare TCP and UDP protocols. While this domain-agnostic approach makes for a good start, there are several flaws in their results. Firstly, no packet loss was experienced, and therefore the measurement was ignored. Secondly, previous studies used round trip time (RTT) as a QoS metric, which [1] did not consider.

Another study that investigated load balancing algorithms looked at throughput, response time, and memory utilization [3]. Measuring memory utilization is effective for load balancing, but we are considering quality of service, with the focus on the end user. The weight on the server only matters if it affects the client. This paper enforces throughput as a necessary metric, and that response time (or RTT) must be considered.

Regencia and Yu's paper develops a completely configurable QoS testing framework [4]. Their framework allows for topology, switch, load, host, and other such configuration, but this configurablility add unnecessary overhead to our specific problem.

3. APPROACH

Mininet is utilized as a network emulator, for it has seen significant use in related works [5, 1, 2] and is well-documented. The Iperf tool was used in [2] for generating network traffic, and its use is considered for the same purposes in this study. We reused the quality of service metrics outlined in [1]:

- Throughput (total transmitted data in bits)/(total time taken in seconds)
- Delay (time required to transmit the data from sender to receiver)
- Packet loss (the number of packets not delivered to their destination)
- Jitter (the variance in latency)

Mininet's toolkit is used to create a server around a simple Node.js application, and artificially adjust flow control, packet drop rate, and jitter from the server-side. Mininet measures packet return trip time (RTT), delay, and throughput from the server to clients. The individual results of these measurements determine how the overall quality of service changes as various server-side features are tweaked dynamically. Additionally, we show how to use Mininet to simulate DoS attacks by a large number of clients in order to observe how attacks affect quality of service to benign users.

The qualitative effects of the QoS metrics we used are analyzed to see which metrics are best for the domain of this problem. As a result of this study, we provide a quality of service measurement outline for future developers to consider

Thank you to Dr. Jianping Pan at the University of Victoria for his teaching and guidance throughout this project and the course.

when testing their Node.js applications in a pseudo-live setting using Mininet.

Our project is extensible and configurable. All network simulation is done on Mininet's pre-built Ubuntu virtual machine image [6], and our source code for running Mininet in Python with a Node.js server is provided at the end of the work.

3.1. Timeline

1. Jan 31st - Feb 13th: Create project outline, gather resources, outline program and create git repository.

2. Feb 14th - Feb 27th: Build testing environment and prepare test.

3. Feb 28th - Mar 13th: Begin tests, create midterm presentation and present.

4. Mar 14th - Mar 27th: Gather results and prepare final presentation. Begin final report.

5. Mar 28th - Apr 7th: Final Presentation. Continue to work on final report.

6. Apr 8th - Apr 16th: Turn in final report.

3.2. Project Website

Please see the associated website for progress reports and results at https://oscarsandford.github.io/qoemf/.

4. EXPERIMENTS

Text here.

5. DISCUSSION

Text here.

6. CONCLUSION

Text here.

7. REFERENCES

[1] Akhilesh Sharma and Aakanksha Sharma, "Qos parameter analysis of tcp and udp traffic over open flow enabled software defined network," in *Proceedings of International Conference on Data Science and Applications*, Mukesh Saraswat, Sarbani Roy, Chandreyee Chowdhury, and Amir H. Gandomi, Eds., Singapore, 2022, pp. 13–25, Springer Singapore.

[2] Pinkey Chauhan and Mithilesh Atulkar, "Achieving enhanced network performance in udp and tcp traffic of software defined networking by selecting java based controllers decisively," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 7, pp. 268–274, May 2020.

[3] Himanshi Babbar, S. Parthiban, G. Radhakrishnan, and Shalli Rani, "A genetic load balancing algorithm to improve the qos metrics for software defined networking for multimedia applications," *Multimedia Tools and Applications*, 2022.

[4] Josiah Eleazar Regencia and William Emmanuel Yu, "Introducing a test framework for quality of service mechanisms in the context of software-defined networking," *Proceedings of Sixth International Congress on Information and Communication Technology*, pp. 687–699, 2021.

[5] Rogério Leão Santos de Oliveira, Christiane Marie Schweitzer, Ailton Akira Shinoda, and Ligia Rodrigues Prete, "Using mininet for emulation and prototyping software-defined networks," in *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2014, pp. 1–6.

[6] "Mininet 2.3.0," 2021.