Dataplane Software Engineer Intern Experimental Learnings

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirement for the Degree
Bachelor of Science in Computer Science, School of Engineering

Luke Allen Francis

Spring, 2024

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Luke Allen Francis

ADVISOR

Aaron Bloomfield, Department of Computer Science

Dataplane Software Engineer Intern Experimental Learnings

CS4991 Capstone Report, 2024

Luke Francis
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
rgn4su@virginia.edu

ABSTRACT
I was brought into a team at the telecommunications company Ciena in order to test subscriber management software. My teammates and I decided to develop an automated testing suite in order to test this software. There were multiple aspects that needed to be implemented in order to flesh out the testing suite. The primary components included an automated traffic generation tool, as well as multiple different configurations of testing procedures to be run, the routing of the data between the traffic generation tool and the software we were testing to receive it, and code to process the results of each test run and store them as appropriate. The test suite was praised by the senior developers on the team, who provided feedback on modifications and additional features to implement. Future steps for this software involve increasing the rate at which the traffic is generated and sent through the subscriber management system. Additionally, a more robust performance evaluation algorithm can be developed to further analyze the results of each test.

1. INTRODUCTION

I began my internship experience at Ciena not as a software developer, but as an Operations Summer Intern in their Blue Planet division in the summer of 2022. While working on this team, I was responsible for business operations, research, and analytics. I interfaced with numerous members of the team in order to create a comprehensive services process document to outline procedures regarding project entitlement, generation, and tracking. This information was used by software consultants in order to streamline their workflows, and it allowed the onboarding process to be much more efficient. I also interfaced with support, financial planning, and business operations teams to help with reporting and automation for their processes.

After a successful first internship with Ciena, I was selected again to be part of a new project. While the first internship catered more to my business minor, this second internship in the summer of 2023 was fully relevant to my computer science major. I moved to the Boston area in the early summer in order to work in person with their team in Burlington, MA. Previously named
Benu Networks, this team had actually been acquired by Ciena within the past year and was because Benu's software complemented the existing portfolio of telecommunications services that Ciena was able to provide to their customers. This was the team's first time having interns, so it was a learning experience for everyone involved.

2. RELATED WORKS

Software testing is an extremely important yet often overlooked part of the development process. Software testing can be tedious, but it is work that needs to be done in order to

protect the customer by providing quality assurance.

As Manns and Coleman (1987) argue: “the nature of the product is different”. Testing software is not like testing other mechanical systems. The testing takes no physical form, so care must be taken to “promote visibility at all stages”. Additionally, due to the fact that software does not degrade means failures may happen repeatedly in quick sequence if the problem is not detected. Finally, “the quality criteria for acceptance are thus debatable”, so thorough work must be done to assure adequate coverage (p. 494).

Another aspect of software testing that is extremely important is discussed by Banker, et al. (1998). They posit that “software comprehension plays a major role in software maintenance and performance” (p. 435). Thoroughly understanding the software being tested is an important step in developing proper testing for the software. Sometimes it can take longer to fully understand the software being tested than to conduct the testing itself. This was an important step I had to take before designing my test suite.

3. PROJECT DESIGN

The design of our project came in multiple steps. The first step in completing the project was to familiarize ourselves with all the tools and software we would be using in order to implement the test suite, including understanding the software that we were testing. This was a multi-week process where some of the team leaders would demonstrate concepts and teach them to us in meetings.

After learning about each aspect of the software we were using, the next step was to begin to get the traffic generation working. This involved using the system TRex, which is described in the next step. After getting the traffic working on TRex, we needed to launch and automate the TRex server. After getting all the TRex infrastructure set up, we needed to build out the testing suite. This involved running different types of packet tests, and configuring these tests. After enabling the tests we wanted to run, the TRex generator would send the traffic to the TRex server, through the subscriber management software, for the corresponding tests. This data was then captured by our software for storage and evaluation.

3.1 Cisco TRex

TRex is a hardware tool developed by Cisco which is able to simulate large amounts of real-world traffic to a specific address. It is primarily used to benchmark networking devices and applications. By simulating large amounts of real-world data, testers can evaluate the performance, scalability, and reliability of network infrastructure. TRex supports various protocols like TCP, UDP, IPv4, and IPv6, all of which were applicable to our testing. It offers features such as stateful generation, traffic analysis, and supports automation which was perfect for our uses.

The first step was learning how to use the TRex system. I practiced launching a TRex server using command prompts and generating large amounts of traffic to the server for varying durations and at various speeds. This testing allowed me to understand how the TRex worked and fit into our testing suite. Once this was understood, we had to learn how to automate all of the TRex aspects within our personal testing suite. This was the much more difficult and time consuming part. We had to learn how to pass all the required parameters and information we needed, launch the TRex server, and execute the run command. We initially started by launching the TRex server manually, separate from the rest of the automation of the testing suite, and then running the testing suite procedure. We

did this in order to begin building out the rest of the testing suite. After some time, I came back to the server automation and was able to successfully get it to automatically run within the execution of the testing suite command. This meant that everything had been fully automated, so only one command had to be run to complete all of the testing (which could be scheduled).

3.2 Routing and Performance Metrics

After finishing work on the TRex system, we had to move on to the testing suite. The first step when working with the testing suite was making sure that all the traffic was being routed correctly. This meant matching the ports of the TRex traffic generation with the ports of our node that the testing suite was running on, in order to allow the traffic to go to the subscriber management software. This is something that needed to be updated and maintained a few times throughout the summer, as ports needed to be changed in the lab.

After all these other steps were done, it was finally time to evaluate the performance of the testing. We captured a large number of metrics about each test, such as packets sent and received, the speed at which the packets were processed, the number of packets dropped (if any), and if the contents of the outgoing packets matched the contents of the incoming packets. We devised an automated algorithm that took these various data points from the software and evaluated their ratios. If these values were within a certain tolerance level, then the test passed.

The last step was to store the data for each run. We developed an automated script to take the configurations, logs, and results from each run and store them within the testing suite. This was done for each individual test run, and the output was marked by the time that the test was executed. These test runs were then organized into directories by days and months

4. RESULTS

At the end of my summer internship, the other interns and I held a demo to show the rest of the team what we had been working on. The automated testing suite was very popular with the full-time developers on the team. They highlighted how this was a long-time want for them, but they did not have the bandwidth to create it themselves since they were constantly working on maintaining or improving the subscriber management software. They detailed how this would save time for them and get rid of existing pain points. They also provided advice and suggestions on how to improve the testing suite, and they outlined a few desired features. Due to the success of the project, I accepted the team's offer to continue working part time over the course of this current school year in order to further improve the testing suite.

5. CONCLUSION

This project was long desired by the senior developers on the Ciena team in Burlington, MA. The test suite that the other interns and I created will allow them to monitor the continued performance of the software that they have created on a daily basis. It will ensure consistent quality and provide benchmark performance assessment numbers for future interactions of the subscriber management software to build upon. This will, in turn, allow them to provide better quality assurance to their customers that the software is reliable. Due to the automated nature of the testing suite, it will also allow them to spend more of their time devoted to other tasks that require their attention.

6. FUTURE WORK

Future work on this project will likely involve increasing the rate of traffic that the testing

suite sends through the subscriber management software. It will be important to learn from and demonstrate how the subscriber management software behaves when it is operating at or near capacity. This will also allow the senior developers to determine what exact level of traffic the software can handle before the performance declines. Another item of future work for this project will be to increase the complexity of the performance algorithm. This will allow the developers to gain further insight into how the subscriber management software is performing in different areas. They also may be able to discern trends or potential problems that aren't readily apparent.

REFERENCES

Banker, R. D., Davis, G. B., & Slaughter, S. A. (1998). Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Study. *Management Science*, *44*(4), 433–450. http://www.jstor.org/stable/2634607

Manns, T. S., & Coleman, M. J. (1987). An Approach to Software Quality Assurance Training. *Journal of the Royal Statistical Society. Series D (The Statistician)*, *36*(5), 493–498. https://doi.org/10.2307/2348660