# Modeling and Performance Analysis of a Converged Network

Jihad Qaddour,
Dept. Information Technology
Illinois State University
Normal, IL, USA
Email: jqaddou@ilstu.edu

Yesh Polishetty
Dept. Information Technology
Illinois State University
Normal, IL, USA
Email: ypolish@ilstu.edu

***Abstract*—Moving from the existing independent infrastructure to integrate shared environment (converged network) will reduce the infrastructure maintenance cost, which we use for performance analysis. This paper addresses a converged network performance analysis based on OPNET simulations. Efficiency metrics, response times, transit time, throughput and transfer delay for different transaction routes in a converged network are evaluated. Various rounds of testing have been conducted using Single User Testing and Load Performance Testing to collect performance metrics to identify architecture bottlenecks. Based on performance analysis, we observed some poor performance as well as more delays in the response times to reach TC servers. We concluded that the response times between a client and TC servers violated the Service Level Agreement. This Paper identifies that most delays were contributed mainly by processing times of TC servers in a converged network.**

***Keywords-Preformance Analysis; Convergence; Response times; OPNET; Modeling.***

## I. INTRODUCTION

When a request is sent from the Trusted Zone in an organization, it passes through several layers. One of the problems is to identify where the bottleneck of the performance of the application is. Figure 1 shows the application architecture, which includes vendor, sales client, admin users, different servers at different tiers, and multiple firewalls to secure the network. Moreover, we show the response times, that will be measured, between the users and different server's zones. The server layer's zones include Mule Servers, TC servers, web servers, and database servers. For sending request transactions from a client to different servers, we use OPNET for modeling the architecture of the network and simulation to calculate different performance metrics, such as transit times, response times, throughput, and others. With the help of these performance metrics, we can analyze the performance of each transaction and compare the result to identify any performance issues, delay violations, and the Bottleneck in the architecture to be resolved [1]. According to Liu [3], application problems can be detected in the earlier stages of the life cycle of the application before converging into the shared environment. OPNET accelerates troubleshooting by rapidly pin pointing the root cause. It ensures application Service Level Agreement (SLA) and the compliance. Using OPNET simulation, we can identify problems related to application performance at different tier levels as well the transit time delays between different applications regardless of its web/windows-based infrastructure [2].

For the application to be converged and move from existing infrastructure to the shared infrastructure using OPNET Simulation, the bottlenecks can be identified, and ways to improve the performance can be analyzed and optimized. This helps the organization move the infrastructure easily, maintain the consistency, and provides redundancy within the available space [5]. According to [3], OPNET tools are used to monitor application performance live in production and to test the environments. Consequently, the application can be easily monitored and maintained to achieve the organization requirements and attain the SLA. Using OPNET modeling and simulation tools, we will collect performance metrics (response times, transit times, throughput, and others) to analyze the performance of the network. We will run two different testing methods: Single User Testing and Load Performance Testing, using three different transactions (Dashboard, URL, and submit requests) to identify the architecture's bottleneck.

 ISBN: 978-1-61208-514-2 

The remainder of the paper is organized as follows: In section II, we present a detailed discussion on all the components of the converged network; Section III describes network application testing where we use two different testing methods; Section IV discusses the OPNET, which was the simulation tool we used for performance analysis and modeling; Section V presents our finding, analysis discussion, and solution to enhance the converged network performance; and Section VI is the conclusion, where we identified the architecture bottleneck and proposed solutions.

## II. COMPONENTS INVOLVED

In this section, we will introduce the components that are involved in the testing: users, different servers, and details about application infrastructure.

### *a. Users*

All users are sending various transactions from different computers, which are known as 'clients' to reach different servers for different business applications. For this testing we are using three different transactions from a client to servers, at different tiers using different connections to be able to compare the results between each tier and identify the performance delays of the application.

### *b. TC Server*

TC server provides organization with secure supported and extended Java application server based, which is fully compatible with Apache Tomcat. Many companies are attracted by the performance and the convergence benefits of using the TC server. TC servers allow for easy installation. When there are any changes in the infrastructure of the application, there will be minimal security risks observed. These risks can be mitigated easily after installing into the shared environment [4].

### *c. Mule Server*

A Mule server is an enterprise service bus, which is used to transfer a request from one server to another. It can be served as a security layer, and using a Mule server can improve the performance of the application as well. The time spent on a Mule server is very short compared to the time spent on the application and web servers [5].

### *d. Web Servers*

Dedicated Web servers are used to extend the security and improve reliability. Web servers can be customized based on the infrastructure and different user requirements [3].

### *e. Application Infrastructure*

Figure 1 explains the application architecture. Network Single User Testing and Load Performance Testing will be conducted using this model and then different results for different testing will be recorded.

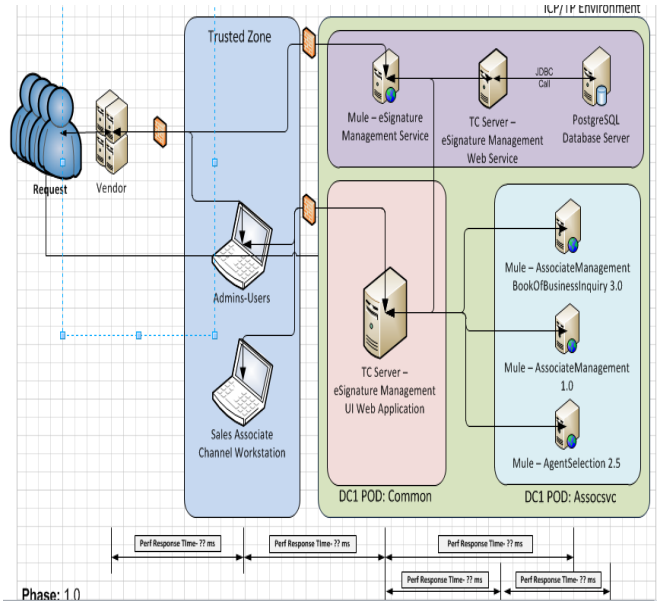



Figure 1. Application Infrastructure

## III. NETWORK APPLICATION TESTING

Two different tests were conducted in this paper: The Load Performance Testing and Single User Testing.

### *a. Load Performance Testing (LPT)*

In this testing, we send multiple requests to the application and we use LPT tools and OPNET to collect different performance metrics (such as response time, transit time, and throughput for performance analysis of the application and network behaviors). The main objective of LPT is to find out where the application breaks, so that the new convergence can be constructed based on the size the application that can stand. Through this testing, we find out the transit times between clients and all servers at different layers of the application. This helps us to calculate the total delays and the break point of the application to stay connected. Based on the testing results, we use the performance metrics as an input to building convergence in the shared environment [2].

 ISBN: 978-1-61208-514-2 

Figure 2 shows object-by-object and different components performance. The red color shows the delay in getting a response. The yellow color indicates the object is processing the request. Finally, the green color indicates the response times. As we can conclude from this test, the red color is very high, which indicates that getting the response takes much longer times. Therefore, this identifies that the delay getting the response is the architecture bottlenecks. A solution for this needs to be investigated, including the replacement of slower devices by faster ones.

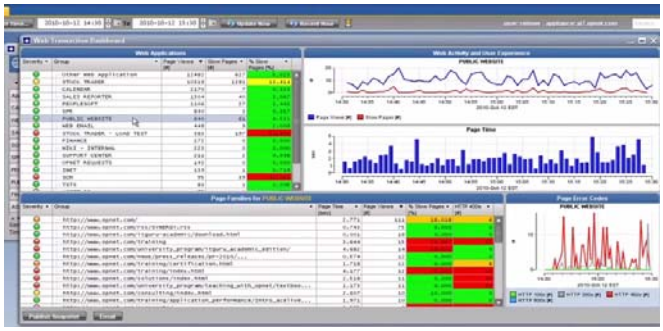


Figure 2. Load Testing Performance

### b. *Single User Testing (SUT)*

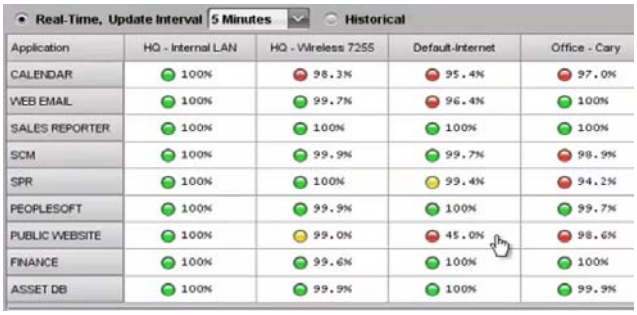
SUT determines the most accurate time for the layers in the application. It also determines the firewall response times and the throughput for the application. This is fast and simple testing compared to the LPT. This testing was used as a smoke testing before conducting the actual LPT [5].

Performing SUT will give the high level information of the application behavior. Using various tools, we can collect metrics during SUT like routing information of requests, response time between two hosts, wait-time, first time to connect, connection time, compression information for each resource, and number and size of requests between each user action [4].

Data collected in SUT can be used to evaluate the application performance during early stages of development, which may greatly resolve most of the performance bottlenecks. Response time and throughput for each component or transaction can be measured and analyzed to find out the possible root cause for delays and bottlenecks in the application to fix the problem [5].

## IV. TOOL USED

OPNET Network tool is used for simulation, network application performance, and to set the application model. OPNET is an event based network simulation tool. Using OPNET, we can conduct different simulation testing to analyze and optimize the performance of hardware and application software of the network. OPNET delivers high definition application performance with the dashboards. Figure 3 shows the Dashboard for an application. This shows various business logics/applications and the related application performance. The green and red colors indicate the status of the application performance when the applications are up and running during the run time environment [4].




| Application | HQ - Internal LAN | HQ - Wireless 7255 | Default-Internet | Office - Cary |
|---|---|---|---|---|
| CALENDAR | 100% | 98.3% | 95.4% | 97.0% |
| WEB EMAIL | 100% | 99.7% | 96.4% | 100% |
| SALES REPORTER | 100% | 100% | 100% | 100% |
| SCM | 100% | 99.9% | 99.7% | 98.9% |
| SPR | 100% | 100% | 99.4% | 94.2% |
| PEOPLESOFT | 100% | 99.9% | 100% | 99.7% |
| PUBLIC WEBSITE | 100% | 99.0% | 45.0% | 98.6% |
| FINANCE | 100% | 99.6% | 100% | 100% |
| ASSET DB | 100% | 99.9% | 100% | 99.9% |

Figure 3. OPNET Dashboard

Using OPNET, the overall application performance can be observed and calculated (such as response times, transit times, and throughput for each tier). In addition, the spikes in the application inside the webserver or database server can be identified as well as the reason for these spikes. OPNET simulation shows that the application is degraded with too many requests simultaneously. We can also measure many performance metrics such as memory performance, throughput, response times, and transit times, which help to optimize them and analyze the performance of different hardware and applications [5].

## V. DISCUSSION AND ANALYSIS

In this section, we present detailed discussion and analysis. We will discuss the application transactions, which observed during different process transactions of testing. Moreover, we will analyze different connections between all components and discuss the testing results to identify the problems.

 ISBN: 978-1-61208-514-2 

*a. Observations*

Figure 1 shows the application architecture, which was used for sending requests and collecting different performance metrics to do performance analyzing and optimization. We sent requests manually, then we observed three different transactions as follow: Transaction 1 was Launch Dashboard without cache. Transaction 2 was Launch Create URL without cache, and Transaction 3 was Submit Request without cache.

TABLE 1. TRANSACTION DETAILS

| Transaction Name | Response Times | Throughput |
|---|---|---|
| Launch Dashboard | 4.2 seconds | 336.3kb |
| Launch URL | 883.9 milliseconds | 107.7kb |
| Submit Request | 3.9 seconds | 136.7kb |

In Table 1, we show the application transactions that have been observed during different transactions of this testing. We also show the response times and throughput information of the three transactions.

*b. Deep Dive Analysis*

Plotting the previous three transactions (Dashboard, URL, and Submit Request launches) versus response times shows that transaction 1 (launch Dashboard without cache) is taking the longest time (4.2 seconds), while the second transaction (launch URL) takes the least time, less than 1s. In Figure 4, we plotted the three transactions versus response times. This result shows that this was due to the increase in connections between different components. The minimum transaction as can be seen in the above table was the URL, which we recommended to all the users to use instead of the other two methods.

As a result, the application performance using Dashboard and Submit Request were the most degraded and took much longer times. Moreover, the (SLA) was defined by the organization to be less than 4 seconds, which means all the transactions should be returned in less than 4 seconds. The shorter the response times, the more effective the result. If we compare the result in Figure 4 to SLA, we conclude that the first transaction (Dashboard) does not meet the requirement (4.2 s > 4.0 s), which is 200 milliseconds more than the business requirement. In the third transaction (sending request) the transaction time barley met the SLA requirement. Therefore, the application performances in these two transactions were degraded. Consequently, a solution is needed, such as replacing some existing hardware with faster alternatives, and improved software.

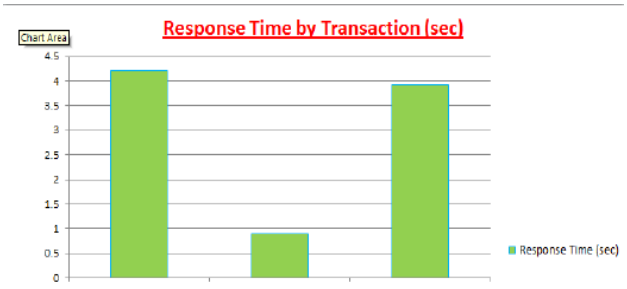



Figure 4. Response Time Graph

TABLE 2. TIER LEVEL DETAILST

| Tier for Launch Dashboard | Latency Observed (milliseconds) |
|---|---|
| Client to Application Central Server | 13.2 milliseconds |
| Client to Open Am Server | 13.2 milliseconds |
| Application central server to TC server | 3.9 seconds |
| Application central server to Environment Server | 273 milliseconds |
| Application central server to Application east server (Bloomington) | 0 |

In Table 2, we show the latency between a client and different servers at different tiers. Using OPNET, we calculated the times between a client and different servers at different tiers. We observed that the request from Bloomington, Illinois for the central server has no latency delay. On the other hand, when the request is sent from a central server to any other area servers, the network latency delay is around 13.2 milliseconds. But the key bottleneck is identified when a request sent from a central server to a TC server: the response time is taking around 3.9 seconds. This delay time is very long, which means the request spent most of the time processing the TC server.

 ISBN: 978-1-61208-514-2 

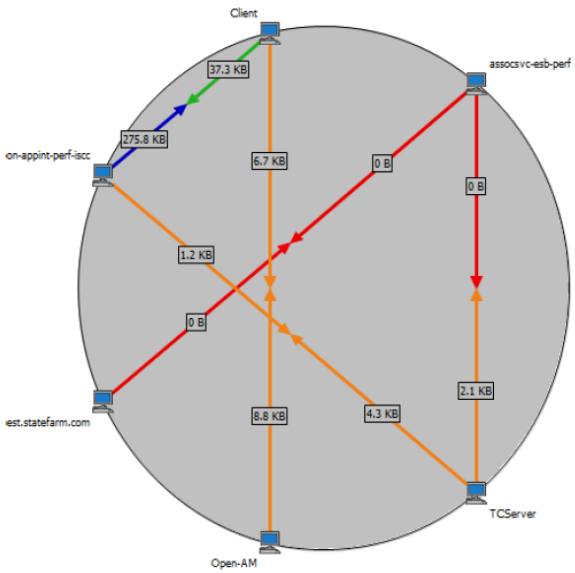



Figure 5. Multi Transaction Report

In Figure 5, we show the transaction report response times and other performance metrics during the testing process of the application. We run the above three transactions (Dashboard, URL, and Submit Request) again for several testing periods. The testing would initially start as SUT to make sure that the application is properly designed. Then the LPT would run several times by sending a sequence of requests at a time to check the application performance, which means more load is put in the architecture. LPT was performed to replicate the exact scenario of the application in the new environment, which will be used by many other services. This is called the shared environment. The outcome of this testing was similar to the previous testing, which result that the most time spent was in the TC server layer. This confirms that, while running a full load in the architecture, we have the same result. Most time was spent in processing, within the TC server. As a solution, changing TC server with a faster one will shorten the process time and delay would be shorter.

## VI. Conclusion

Using OPNET modeling and simulation, we collected performance metrics (response times, transit times, throughput, and others) to analyze the performance of the network. We run two different testing: SUT and LPT, using three different transactions (Dashboard, URL, and Submit Request). After running these tests several times, we conclude that the architecture bottleneck was identified to be the processing time in the TC server layer, which is causing most delays. Moreover, the delay is violating the SLA (equals 4s), which means all the transactions times should be returned in in less than 4 seconds. For future work, we recommend a full investigation of the solutions to this problem. One solution could be replacing the TC server with a faster one. Another solution might be limiting the connections times between the application's units to improve the performance of the applications and reduce the cost of the infrastructure. Such solutions can be investigated in an another paper.

## References

[1] X. Chang, "Network simulations with OPNET," Proceedings of the 31st conference on Winter simulation, Simulation a bridge to the Future, Volume 1, pp. 307-314.

[2] C. Zhu, O. W. Yang, J. Aweya, M. Ouellette, and D. Y. Montuno, "A comparison of active queue management algorithms using the OPNET Modeler," IEEE Communications Magazine 40, pp. 158-167, June 2002.

[3] K. Salah and A. Alkhoraidly, "An OPNET-based simulation approach for deploying VoIP," International Journal of Network Management 16, pp. 159-183, March 2006.

[4] M. S. Hasan, H. Yu, A. Carrington, and T. C. Yang, "Co-simulation of wireless networked control systems over mobile ad hoc network using SIMULINK and OPNET," IET Communications 3, pp.1297-1310, 2009.

[5] M. Young, P. Leys, J. Potemans, B. Van den Broeck, J. Theunis, E. Van Lil, and A. Van de Capelle, "Use of the raw packet generator in OPNET," OPNETWORK 2002 Handbook, Mill Valley, CA University Science, 2002.

[6] Q. Duan, "Modeling and delay analysis for converged network-cloud service provisioning systems," Computing, Networking and Communications (ICNC), January 2013.

 ISBN: 978-1-61208-514-2 