CHAPTER 5

DISCUSSION OF THE RESULTS

5.1 INTRODUCTION

This chapter will be devoted to an exploration of the findings, and the results will be elucidated. Conclusions will be drawn about how successful the proposed architecture has proved to be. The results that were obtained from the experiments described in Chapter 4 will provide insight into the performance of the system under certain specified load and failure conditions.

Solutions have been sought for the proposed research questions that were stated in Chapter 1, specifically 1.3.2, questions 1, 2 and 3. These questions are:

1. How could VOMS be described and contextualised as a theoretical framework of the study?

2. Could VOMS be designed to have a near 100% application service availability during a failure?

3. If it were possible to achieve a near 100% application service availability, would it be economically feasible to build such a new architecture?

The questions were addressed successfully.

5.2 DISCUSSION OF THE RESULTS

In the experimental phase of the study, the client machine which had a JAVA JMeter software tool installed to emulate traffic from clients to the server was situated in an IT domain, while the servers of the newly designed application and database servers were located in a test Mobile Switching Centre (MSC). Latency between IT and the test MSC had an impact on the response times and affected the results. The test IN that was used
was also used by many other test departments and that also had an impact on the overall responses.

5.2.1 Discussion of an existing VOMS

A theoretical model was created to explain how an existing VOMS architecture would handle a failure as compared to the newly designed architecture for VOMS. In Chapter 3, Figure 3.14 displayed the behaviour of an existing VOMS during a failure. Under a failure condition, an existing VOMS system would cease to continue transacting. This was in contrast to the proposed solution for the new VOMS architecture where the system would not fail and continue to be available and continue transacting.

The first test that was conducted showed how an existing VOMS architecture would perform under a failure condition. Directly after the failure was triggered, at the two- (2) minute interval, the results could be seen clearly. In Figure 4.1(Chapter 4) both successes and failures as recorded during the emulations were depicted. After the failure had occurred, it could be seen that the successful transactions failed abruptly. The system did not process transactions from that point on. Even though it was only the application portion of the server that failed, the entire system became unresponsive to the clients’ requests. The mobile operator would need to perform a failover to the secondary site to restore the service until the primary service could be recovered.

5.2.2 Creating a baseline for the newly designed VOMS system

The second test that was conducted was to establish a baseline for the conditions under which the failures would be conducted. No failure conditions were introduced during this emulation. As could be seen from Figure 4.4 the results were clear. Figure 4.3 recorded both the successes and failures of the emulations. The initial emulations ran for 480 seconds to provide an overview of the performance of the newly designed system. It should be noted that the downstream systems, including the network to which these systems were connected, had an impact on the results and throughput.
Emulations with no failures

A total of five emulations were run to determine how many samples could be processed within the allotted 480 seconds. Other parameters were also recorded (see Table 5.1).

**TABLE 5.1: Results from the emulations run with no failures**

<table>
<thead>
<tr>
<th>EMULATIONS</th>
<th>#SAMPLES</th>
<th>AVERAGE</th>
<th>MIN</th>
<th>MAX</th>
<th>STD. DEV.</th>
<th>ERROR %</th>
<th>THROUGHPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run1</td>
<td>57331</td>
<td>839</td>
<td>22</td>
<td>12261</td>
<td>1423.326</td>
<td>0.075</td>
<td>118.030</td>
</tr>
<tr>
<td>Run2</td>
<td>53313</td>
<td>905</td>
<td>21</td>
<td>12114</td>
<td>1413.623</td>
<td>0.110</td>
<td>109.843</td>
</tr>
<tr>
<td>Run3</td>
<td>55959</td>
<td>855</td>
<td>29</td>
<td>16952</td>
<td>1414.792</td>
<td>0.071</td>
<td>115.887</td>
</tr>
<tr>
<td>Run4</td>
<td>60544</td>
<td>1492</td>
<td>29</td>
<td>19680</td>
<td>1377.625</td>
<td>0.125</td>
<td>100.203</td>
</tr>
<tr>
<td>Run5</td>
<td>66887</td>
<td>713</td>
<td>25</td>
<td>11855</td>
<td>971.970</td>
<td>0.116</td>
<td>138.435</td>
</tr>
<tr>
<td>Av. Run</td>
<td>58806.8</td>
<td>960.8</td>
<td>25</td>
<td>14572</td>
<td>1320.267</td>
<td>0.100</td>
<td>116.480</td>
</tr>
</tbody>
</table>

Samples

For the duration of 480 seconds the system managed to process an average of 58806.8 transactions. The sample set of subscribers was only 25000. The Apache JMeter was able to continue with the testing and to loop the pool of subscribers until the tests were completed.

Average, Minimum, Maximum, and Standard Deviation

As can be seen in Table 5.1, the average responses per transaction were 960ms.

Error Percentage

Table 5.1 shows that the sample of subscribers chosen was not all created on the test IN. Of the 25000 subscribers, 1000 did not exist. These would account for a failure of 4%. The failures noted during the emulation runs were above this value. Upon investigation, it was noted that the responses were timeouts. The downstream test IN did not respond in a timely manner, which could also be attributed to latency over the network.
Throughput

In Table 5.1 it can be seen that the throughput of the system during normal runs averaged 116 transactions per second. Note that the emulations were set up to have 100 concurrent connections to the server. This value would be used as the average expected transactions that this system was capable of handling.

5.2.3 Emulations with the loss of the front-end application server

The third test conducted was to determine the behaviour of the system under the failure of the application server. The failures were introduced during this emulation after two (2) minutes and the results noted. Figure 4.5 recorded both the successes and failures during the emulations. These emulations were run for 240 seconds.

The results of the emulations with the loss of the application server (see Table 5.2).

<table>
<thead>
<tr>
<th>EMULATIONS</th>
<th>#SAMPLES</th>
<th>AVERAGE</th>
<th>MIN</th>
<th>MAX</th>
<th>STD. DEV.</th>
<th>ERROR %</th>
<th>THROUGHPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN1</td>
<td>29157</td>
<td>530</td>
<td>30</td>
<td>6171</td>
<td>516.405</td>
<td>0.060</td>
<td>121.328</td>
</tr>
<tr>
<td>RUN2</td>
<td>29362</td>
<td>515</td>
<td>20</td>
<td>5492</td>
<td>543.050</td>
<td>0.063</td>
<td>122.107</td>
</tr>
<tr>
<td>RUN3</td>
<td>28438</td>
<td>558</td>
<td>25</td>
<td>6999</td>
<td>517.075</td>
<td>0.063</td>
<td>118.485</td>
</tr>
<tr>
<td>RUN4</td>
<td>20819</td>
<td>537</td>
<td>19</td>
<td>6778</td>
<td>675.647</td>
<td>0.050</td>
<td>86.0171</td>
</tr>
<tr>
<td>RUN5</td>
<td>27165</td>
<td>496</td>
<td>6</td>
<td>6170</td>
<td>731.141</td>
<td>0.054</td>
<td>113.084</td>
</tr>
<tr>
<td>Av. Run</td>
<td>26988.2</td>
<td>527.2</td>
<td>20</td>
<td>6322</td>
<td>596.664</td>
<td>0.058</td>
<td>112.204</td>
</tr>
</tbody>
</table>

Samples

The failures for the emulation of the front-end application servers were induced over a period of 240 seconds. During this emulation an average of 26988.2 subscriber transactions were processed.

Note that the average number of transactions processed (Table 5.1) for the base-line transaction was higher than for the failure of the front-end application server. The deduction that could be made from this was that for this hardware setup there were only two front-end servers. When the application server was made to fail, the transactional throughput was reduced.
Average, Minimum, Maximum and Standard Deviation

The heading describes the system’s response to the transactions. Table 5.2 shows the average response per transaction was 527ms. This response time was considered to be acceptable as it was found to be the baseline in the mobile operator’s live system for any transactions with a response time of less than 800ms.

Throughput

The findings depicted in Table 5.2 demonstrate that the throughput and the samples processed confirm the statement made previously, namely ‘When the application server was made to fail, the transactional throughput was reduced’ - indeed, there was a slowdown in the throughput of the system.

Error Percentage

From the findings in Table 5.2 it can be deduced that the errors that occurred during this emulation run were relatively low. This could be attributed to various reasons, including the shorter duration of the emulation and the time of day the emulations ran. The emulations that formed the baseline were done during the day. The failure emulations were run after-hours.

Summary for the front-end application failure

From (Chapter 4) Figure 4.5 and Table 5.2, the following deductions can be made:

- Connectivity to the VIP remained alive and the client (Java JMeter) did not lose a connection after the failure occurred, therefore the client would not have experienced any loss of connectivity.
- For this configuration setup, there were only two front-end servers; when the failure occurred it was observed that the transactional throughput rate decreased.
5.2.4 Emulation with the loss of the back-end database server

The fourth test conducted was to determine the behaviour of the system under the failure of the database server. The failures were introduced during this emulation after two (2) minutes and the results noted. As could be seen in Figure 4.7 (Chapter 4), the results were clear. Figure 4.6 recorded both the successes and failures during the emulation. These emulations ran for 240 seconds.

The results of the emulation with the loss of the database server (Table 5.3)

<table>
<thead>
<tr>
<th>EMULATIONS</th>
<th>#SAMPLES</th>
<th>AVERAGE</th>
<th>MIN</th>
<th>MAX</th>
<th>STD. DEV.</th>
<th>ERROR %</th>
<th>THROUGHPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run1</td>
<td>27932</td>
<td>453</td>
<td>32</td>
<td>6128</td>
<td>478.629</td>
<td>0.074</td>
<td>116.121</td>
</tr>
<tr>
<td>Run2</td>
<td>35526</td>
<td>537</td>
<td>30</td>
<td>8569</td>
<td>754.383</td>
<td>0.074</td>
<td>147.744</td>
</tr>
<tr>
<td>Run3</td>
<td>30998</td>
<td>462</td>
<td>30</td>
<td>6008</td>
<td>481.599</td>
<td>0.048</td>
<td>129.237</td>
</tr>
<tr>
<td>Run4</td>
<td>28868</td>
<td>470</td>
<td>25</td>
<td>6198</td>
<td>609.304</td>
<td>0.050</td>
<td>119.468</td>
</tr>
<tr>
<td>Run5</td>
<td>30106</td>
<td>453</td>
<td>30</td>
<td>6153</td>
<td>560.360</td>
<td>0.058</td>
<td>123.841</td>
</tr>
<tr>
<td>Av. Run</td>
<td>30686</td>
<td>475</td>
<td>29</td>
<td>6611</td>
<td>576.855</td>
<td>0.060</td>
<td>127.282</td>
</tr>
</tbody>
</table>

Samples

The failures for the emulation of the back-end database servers were induced over a period of 240 seconds. During this time frame an average of 30686 subscriber applications were processed.

Note that this value was lower than for the failure of the back-end database servers. The deduction that can be made is that for the hardware setup there were only two back-end servers. When the one was made to fail, the transactional throughput increased. This was due to the fact that the cost of fault tolerance was a loss in performance for the databases. To maintain the cluster, throughput was reduced proportionately, as seen in Table 5.3.
Average, Minimum, Maximum and Standard Deviation

The system’s response to the transactions is indicated in Table 5.3. It is indicated in Table 5.3 that the average response per transaction was 475ms. This response time was considered to be acceptable, as this was found to be the baseline in the mobile operator’s live system for any transactions with a response time of less than 800ms.

Throughput

The results provided in Table 5.3 clearly indicate that the throughput and the samples processed confirm the statement that, “A loss of a back-end database would increase the performance and throughput of the system”. Therefore, during the loss of a database server, there was an increase in the throughput of the system as compared to the throughput indicated in Table 5.1 and Table 5.2.

Error percentage

From the data in Table 5.3 it is clear that the errors that occurred during this run were relatively low. This could be attributed to various reasons; including the shorter duration of the emulation, and the time of day the emulations ran. The emulations that formed the baseline were done during the day, while the failure conditions ran after hours. The downstream IN system, which was required for this functional testing, was used by various divisions during the day. The reason for the failures was discussed in Chapter 4, (4.6.1).

Summary for the Front-end Application Failure

The following inferences can be made from the data provided in Figure 4.5 (Chapter 4) and Table 5.3:

• Connectivity to the VIP remained alive and the client (Java JMeter) did not lose a connection after the failure occurred, therefore that the client would not have experienced any loss of connectivity.
• In this setup, there were only two back-end servers; when the failure occurred it was observed that the transactional throughput rate had decreased even though there was a spike in throughput rate at the point of failure.

5.3 CONCLUSION

Chapter 5 was devoted to a discussion of the results of the laboratory tests. After an analysis of the emulation data, it was evident that during a fault condition the transactions continued to be processed by the new VOMS system. By virtue of the comparison done on an existing architecture, transactions would not, or rather, could not, be processed during a failure condition as mentioned. The examination of the results proved that the new architecture, built from high availability concepts, had been proven to be effective during failures.

The research questions were addressed; VOMS and the components that constitute the makeup of a VOMS system were described in detail. Further details were provided on the functionality that VOMS provided, and the discussion of the findings put the importance of the newly developed VOMS in context.

The overall cost of developing a new VOMS system was presented in Chapter 4, and it was noted that it would be economically feasible to build such a new system, when the cost is compared to that of the installation of an existing proprietary VOMS system.

In this chapter the results of the tests that were run have been discussed. These results indicate that the application service remained available during the failure conditions. Clients that connect to such a system will not lose their connectivity and may be oblivious of a failure occurring. While an application service availability of 100% was achieved during the experimentation, the duration of the emulations was for a short period. While the experiment configuration proved to have a better service availability when compared to an existing VOMS architecture, it could be further enhanced as the experiment architecture only catered for 100% capacity. The architecture could be enhanced to achieve an eight 9’s application service availability. To achieve this extremely high service availability, additional capacity would need to be provided according to Holenstein et al. [39]. Chapter 6 will address the guidelines to further enhance the overall service availability.
In the next chapter, Chapter 6, entitled, **Guidelines, recommendations and conclusion**, guidelines will be given and recommendations made, after which the report will be concluded.