CHAPTER 4

DESCRIPTION AND INTERPRETATION OF THE RESULTS

4.1 INTRODUCTION

In this chapter the results of the laboratory experiments performed are described and interpreted. The research design and methodology were elucidated in Chapter 3, and the problem of the conventional architecture of a VOMS system and its inherent risks were discussed. The concept for a solution was given and the construction of a newly proposed VOMS system, as well as a breakdown of each building block, was explained. The equipment that was used in the laboratory was described in detail, and its technical specifications were provided. A theoretical model for the results was also described.

Emulations were processed in the laboratory for both an existing VOMS architecture and the newly proposed solution. Adjustments to the specifications of the components for the newly proposed architecture were performed to allow for optimal performance. The results were obtained from a load runner tool, which enabled the experiments to provide a meaningful output and result. Hereafter the architecture was put through its paces, as it was tested to withstand various failure conditions. These results were captured as graphs and presented to show the success of the design. This was achieved through a comparison between an existing VOMS system and the newly proposed VOMS system.

A comparison was then done between an existing VOMS systems and the newly designed VOMS architecture with regard to the benefits that the new system would have over an existing system.

4.2 RESULTS

The results were collected from a series of emulations carried out on the newly proposed architecture. It began with the standard conditions where there were no failures and then progressed to all combinations explained in Chapter 3 (sections 3.7 and 3.8).
4.3 EMULATIONS FOR AN EXISTING VOMS ARCHITECTURE UNDER FAULT CONDITIONS

Chapter 3 (3.8) described the experiments that were performed. The laboratory environment was configured to resemble the environment and architecture of an existing VOMS system. The emulation was set up to introduce a failure condition and to show the behaviour of an existing system.

The emulation was started and the transactions monitored. After a specified time of two minutes, the application services were shut down manually to introduce the faulty condition; the response to this failure can be seen in Figure 4.1.

The y-axis denoted the transactional throughput in transactions per second, annotated by 'Number of transactions \ second'. The x-axis denoted time that is, the duration of the emulation being conducted. The blue line graph denoted the number of successful transactions that was processed and the red line graph the number of failures or no responses that was detected.

The results were clear: when the system fault was introduced, an existing VOMS system did not tolerate the severe failure condition and stopped processing transactions.

In Figure 4.3 this is indicated by the blue line graph at a time of two minutes. The red line graph of the failure line graph continued up until the end of the emulation.

As soon as the failure was introduced, the blue line graph, denoting successful responses from the downstream IN system, stopped. The VOMS had encountered an event that it could not tolerate and thus became unable to further process transactions from the client emulating program, Java JMeter. The red line graph denoting the failures received by the load tool, Java JMeter continued to record unsuccessful responses from the VOMS to which it was connected. Java JMeter was still attempting to connect and process transactions until the end of the emulation.
FIGURE 4.1: A system failure of an existing system
The existing VOMS at this point in time would be considered non-responsive. Clients would not be able to process any transactions. The only solution to this faulty condition would be to fail over the services to the Disaster Recovery (DR) system to restore service to the clients. The primary server could then be worked on to recover the failure condition and restore functionality to bring the server back into service. This emulation proved that an existing VOMS was not immune to a failure of its application or database.

Once the services could be restored to the primary server, another failover event would be required to bring the client traffic back to the primary server. There would always be a risk when moving client traffic between primary and secondary sites. Complications due to failovers could adversely affect connectivity to the servers from the clients and prolong a service outage.

In Figure 4.2 it can be observed that there were periods of failure spikes and success dips. When reading the log files generated by Java JMeter, it is evident that the failure spikes (red line) were due an HTTP 500 internal server error. This was a general error and was not specific enough to determine the exact cause of the evident failure. Upon further investigation it was noted in the front-end application server logs that the application did not receive a response from the downstream IN system for those specific transactions.

With regard to the successful dips (blue line), it was discovered that they were caused by slow responses. The responses were valid, but with a significantly lower response rate from the downstream IN system. A plausible explanation for the behaviour of the downstream IN system was that it might have been busy with other tasks during the emulation run.
4.4 ESTABLISHING A BASELINE

It was important to know the performance and overall behaviour of the newly designed VOMS system before any failures were introduced.

4.4.1 Results under normal conditions with no failures for the newly designed VOMS system

In Figure 4.3 it will be noted that the y-axis denotes the transactional throughput in transactions per second annotated by ‘Number of transactions \ second’. The x-axis denotes time, that is, the duration of the emulation being conducted. The blue line graph denotes the number of successful transactions that was processed and the red line graph the number of failures or no responses that was detected.

In Figure 4.3 the blue line graph (Activate HTTPClient Success) denotes the successful transactions that took place. The red line graph (Activate HTTPClient Failure) denotes the unsuccessful transactions or non-responses. These unsuccessful or non-responses were primarily due to the downstream test IN not responding in time.

It must be noted that during these laboratory emulations the resources on the downstream test IN were limited and had an impact on transactional throughput.

The base line graph proved useful to indicate what the system limitations were and how the system responded to the emulated client requests.

A conclusion as to the factors that contributed to the dips in the successful transactions was that they were caused by the environment.

The client requests were emulated from outside the laboratory. JAVA JMeter did detect some latency with regard to the transactional requests. While this was observed it was believed that this would not negatively influence the data collected during the evaluation.

The baseline emulation was run for a period of eight minutes. Five such emulations were performed and the data recorded. The results of these emulations will be discussed in Chapter 5.
FIGURE 4.3: Number of successes vs. failures
Figure 4.4 depicts a further analysis of the graph to highlight the observations of the behaviour of the success versus failure graph. For the successful blue line graph, it was observed that the successful transactions had a varying response rate. The majority of the transactions had a response rate of between 150 and 210 transactions per second. There were many dips in the transaction rate. It was investigated and found to be as a result of the downstream IN system that did not respond timeously. In this scenario, the VOMS acts as the middle layer system to the downstream IN. It was determined that these dips did not influence or had any impact on the outcomes of the experiment. It was seen as merely the behaviour of the downstream IN system.

The red line graph which denoted failures was observed as being prevalent throughout the emulation. An investigation for the reason of those failures was noted as valid failures. The response received in Java JMeter’s logs was an HTTP 200 message, which meant a successful response, although the Java JMeter logged this as a failure as they did not exist on VOMS or IN. The list of numbers or MSISDNs that were used in the emulation and passed on to VOMS did not all exist on the VOMS database or downstream IN.

4.5 FAILURE ON THE NEWLY DESIGNED VOMS ARCHITECTURE, LOSS OF THE FRONT END

The baseline was established for the new architecture and then the system had to be tested to determine if the newly designed VOMS systems would be fault tolerant, and specifically and more to the point, would the application service be available during a fault condition?
The theoretical behaviour of the newly designed VOMS system has been described in Chapter 3 (3.7), where it is indicated that the system would not fail under an emulated fault condition.

The experiment was described in Chapter 3 (3.8.2). Figure 3.12 in (Chapter 3) indicates exactly which part of the architecture would be emulated to fail.

4.5.1 Results for the loss of the front-end application servers

In Figure 4.8 it will be noted that the y-axis denotes the transactional throughput in transactions per second annotated by 'Number of transactions \ second'. The x-axis denotes time, that is, the duration of the emulation being conducted. The blue line graph denotes the number of successful transactions that was processed and the red line graph the number of failures or no responses that was detected.

In Figure 4.5 the blue line graph (Active HTTPClient Success) denotes the successful transactions that were taking place. The red line graph (Active HTTPClient Failure) denotes the unsuccessful transactions or non-responses. The emulations were configured to show the effect on the newly designed VOMS system after a fault was introduced after two minutes.

The fault that would be triggered for this emulation run was the loss of the front-end system or the application server. Figure 4.1 depicts an existing VOMS system that failed and stopped processing transactions after the fault was introduced. An observation was made that after 120 seconds, when the failure was introduced, the transactional throughput decreased as the system was now only utilising one application server.

The new design for VOMS was proven to be available during the failure and not fail under this faulty condition. The success of the system’s behaviour can be seen in Figure 4.8. The successful transactions (the blue line graph, 'Activate HTTPClient Success’) continued to be processed whilst the faulty condition continued.

In Figure 4.5 it can be observed that the red line graph (Activate HTTPClient Failure) did not increase during the time of the failure condition, nor did it increase post the event. Five such emulations were performed and the data recorded. The results of these emulations will be discussed in Chapter 5.
FIGURE 4.5: Front-end application server failure with no disruption to service
Three main observations can be made from Figure 4.6. With regard to the successful transactions denoted in blue, the graph shows that there were many transactions being processed above an average rate of 120 transactions per second. One dip can be seen, which was investigated and attributed to the downstream IN responding slowly to the load running tool.

After the failure was introduced at the two-minute mark, the successful transactions continued in Figure 4.6. The graph post the failure condition indicates that not as many transactions were processed as before the failure. There were no significant failures recorded during the experiment. Those failures that were present during the emulation were investigated and attributed to subscribers not found on the downstream IN system. It was observed that there were no red line failure spikes during and after the failure point.

4.6 FAILURE ON THE NEWLY DESIGNED VOMS ARCHITECTURE, LOSS OF THE BACK END

The next emulation focused on the loss of the backend database server. The results were expected to be the same or very similar to those indicated in Figure 4.8, according to the theoretical model.

The experiment was described in Chapter 3 (3.8.3). Figure 3.13 (in Chapter 3) indicates exactly which part of the architecture would be emulated to fail.
4.6.1 Results for the loss of the back-end MySQL cluster node

In Figure 4.7 the y-axis denotes the transactional throughput in transactions per second, annotated by 'Number of transactions \ second'. The x-axis denotes time, that is, the duration of the emulations being conducted. The blue line graph denotes the number of successful transactions that was processed and the red line graph the number of failures or no responses that was detected.

The emulations were configured to show the effect on the newly designed VOMS system after a fault was introduced after two minutes.

The fault that was triggered for this emulation was the loss of the back-end system or the database server. In Figure 4.1 it was indicated that an existing VOMS system failed and stopped processing transactions after the fault was introduced.

The new design for VOMS was proven to be fault tolerant and did not fail under this faulty condition. The success of the system’s behaviour can be seen in Figure 4.7. The successful transactions (the blue line graph, ‘Activate HTTPClient Success’) continued to be processed whilst the faulty condition continued. An observation was made that after 120 seconds, when the failure was introduced, the transactional throughput decreased as the system was now only utilising one database.

Another observation noted is that the red line graph (Activate HTTPClient Failure) did not increase during the time of the failure condition, neither did this happen post the event (see Figure 4.7).

Five such emulations were performed and the data recorded. The results of these emulations will be discussed in Chapter 5.
FIGURE 4.7: Back End database server failure with no failure of service
From Figure 4.8 the following observations were made with regard to the successes (blue line) before the failure. The response rate varied substantially and was not consistent. At the point highlighted in a blue circle before the failure, a low response rate is noted. The log files of Java JMeter were analysed and no error or abnormalities were noted. Upon further investigation, it was found that the downstream IN system was responding slowly to the client requests made from Java JMeter. A possible reason was that the downstream IN system was busy.

When the failure was introduced a spike was noted in the success graph (blue line), just after the two minute mark. The higher response rate level lasted for around fifteen seconds before slowing down to a rate of between fifty and one hundred transactions per second. A possible reason for this behaviour could be attributed to the clustered database. The cluster in the experiment consisted of two database servers. They worked independently, but replication bound the two servers, so that data integrity was maintained. When a transaction was sent to one database server, the transaction was sent to the clustered node - in this scenario, the second database server. The transaction was confirmed on the second database node before committing the work. This does add as an overhead on the overall transaction time. When the failure occurred, one of the databases servers was taken out of service. The ACE load balancer knew that it had lost a connection to the faulty database server and removed it from the available resource pool. The replication link would also fail and the cluster would be updated to note the failed database server. No transactions would be sent to the faulty database server and neither would the replicated data be sent. This would allow the database to process the transactions without the overhead, thus resulting in an increased transaction rate. The increased transaction
rate did not last long before the success rate settled down. The downstream IN system was not consistent with the overall response rate, thus making exact arguments difficult.

The overall failures (red line) observed were similar to those in Figure 4.5, the failure of the front-end application server. It was observed that at the time of the failure condition, there was no increase in the overall failure rate. The inference that can be made would be that the new architecture performed well and provided a stable connection to the clients that were transacting on the overall system. It was observed that there was a single spike before the failure was introduced and it was investigated. The log files in Java JMeter showed that there were errors, specifically, an HTTP 500 internal server error. When the application front-end logs were analysed, it was noted that they were not responding due to the downstream IN.

4.7 PROBLEMS ENCOUNTERED WITH THE SETUP

Of the various problems that were dealt with, the time that was required to set up the laboratory was the most challenging. Consultants were required to assist with the building of certain components for the new VOMS configuration and setup. The complexity of the setup and importation of data to the MySQL cluster took a fair amount of time to achieve. Due to licensing, the software that was required to perform the load testing had to be open source. After some investigation was undertaken, Java JMeter was chosen as the tool to perform the analysis.

4.8 HIGH LEVEL FINANCIAL COMPARISON BETWEEN EXISTING AND PROPOSED VOMS

A comparison was performed between an existing VOMS system and the newly proposed VOMS. The pricing model described in Table 4.1 used approximate values to convey the different costing models.
<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>NEARLY PROPOSED SYSTEM</th>
<th>PRICE</th>
<th>EXISTING SYSTEM</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware CPU and Memory</td>
<td>Fujitsu RX 900</td>
<td>R900 000</td>
<td>HP-NSK</td>
<td>≈R30 000 000</td>
</tr>
<tr>
<td>Load Balancers</td>
<td>Cisco ACE 6500</td>
<td>R400 000</td>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>Storage Disk Array</td>
<td>Provided by Hitachi</td>
<td>R100 000/ TB HDD</td>
<td>Provided by HP</td>
<td>≈R10 000 000</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>R1 400 000</td>
<td>≈R40 000 000</td>
<td></td>
</tr>
<tr>
<td>Software Licensing</td>
<td>Red Hat</td>
<td>R150 000</td>
<td>HP-NSK</td>
<td>≈R15 000 000</td>
</tr>
<tr>
<td>Operation System Support</td>
<td>MySQL Cluster</td>
<td>Free</td>
<td>NSK-SQL</td>
<td>≈R5 000 000</td>
</tr>
<tr>
<td>Total</td>
<td>VENDOR SUPPORT</td>
<td>R2 000 000</td>
<td>HP Certified Vendor</td>
<td>≈R12 000 000</td>
</tr>
<tr>
<td>Total Price for single site</td>
<td>Single Site</td>
<td>R5 750 000</td>
<td>Single Site</td>
<td>≈R62 000 000</td>
</tr>
<tr>
<td>Total Price for Dual Site Redundancy</td>
<td>Dual Site</td>
<td>R11 500 000</td>
<td>Dual Site</td>
<td>≈R124 000 000</td>
</tr>
</tbody>
</table>

Based on the information in Table 4.1, it can be inferred that the new system would be substantially more cost effective to implement and thus it would be economically feasible to migrate to the new VOMS architecture.

There are two main parts to any new system, which are the costs of CAPEX and OPEX. As indicated in Table 4.1, the highlighted section in blue indicates the CAPEX portion of the systems, while the highlighted sections in orange indicate the OPEX. The CAPEX costs for the new VOMS system would be cheaper to procure. The maintenance thereof or the budget that would cover this cost indicated that the OPEX would be cheaper than an existing system.

Proprietary systems and the third party software required for these systems, such as that of HP-NSK systems, were costly, while the non-proprietary systems and third party software, such as the Fujitsu Siemens RX900 proved to be more cost effective and a proven alternative.

4.9 CONCLUSION

Chapter 4 provided an overview of the description and interpretation of the results. The different failover conditions were presented. Emulations that were conducted to show the results of an existing VOMS during a failure, and which highlighted the loss of service, were discussed. A
baseline test was run to provide a control for the study. The new VOMS system architecture was subjected to failure of the front-end application server as well as the back-end database. Emulations proved that the new architecture was able to persevere through faulty conditions as compared to an existing VOMS which could not. Therefore, the new VOMS had higher service availability as compared to an existing VOMS.

A cost comparison was conducted between the existing VOMS system and the new VOMS system. It was shown that commodity hardware and software would be more cost effective to use as compared to proprietary systems.

In the next chapter, Chapter 5, Discussion of the results, the results will be discussed in detail.