Queensland University of Technology
Faculty of Science and Technology

Smart Grid Test Bed Design and Implementation

Nishchal Kush
Master of Information Technology
(Research Project)

Supervisors
Prof. Andrew Clark
Dr. Ernest Foo
Dr. Ejaz Ahmed

November 11, 2010

Abstract

Given there is currently a migration trend from traditional electrical supervisory control and data acquisition (SCADA) systems towards a smart grid based approach to critical infrastructure management. This project provides an evaluation of existing and proposed implementations for both traditional electrical SCADA and smart grid based architectures, and proposals a set of reference requirements which test bed implementations should implement. A high-level design for smart grid test beds is proposed and initial implementation performed, based on the proposed design, using open source and freely available software tools.

The project examines the move towards smart grid based critical infrastructure management and illustrates the increased security requirements. The implemented test bed provides a basic framework for testing network requirements in a smart grid environment, as well as a platform for further research and development. Particularly to develop, implement and test network security related disturbances such as intrusion detection and network forensics.

The project undertaken proposes and develops an architecture of the emulation of some smart grid functionality. The Common Open Research Emulator (CORE) platform was used to emulate the communication network of the smart grid. Specifically CORE was used to virtualise and emulate the TCP/IP networking stack. This is intended to be used for further evaluation and analysis, for example the analysis of application protocol messages, etc.

As a proof of concept, software libraries were designed, developed and documented to enable and support the design and development of further smart grid emulated components, such as reclosers, switches, smart meters, etc. As part of the testing and evaluation a Modbus based smart meter emulator was developed to provide basic functionality of a smart meter. Further code was developed to send Modbus request messages to the emulated smart meter and receive Modbus responses from it. Although the functionality of the emulated components were limited, it does provide a starting point for further research and development. The design is extensible to enable the design and implementation of additional SCADA protocols.

The project also defines an evaluation criteria for the evaluation of the implemented test bed, and experiments are designed to evaluate the test bed according to the defined criteria. The results of the experiments are collated and presented, and conclusions drawn from the results to facilitate discussion on

the test bed implementation. The discussion undertaken also present possible future work.

List of Figures

List of Tables

Contents

Declaration

The work contained in this report has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, this report contains no material previously published or written by another person except where due references have been made.

Signature: _____ Date: November 11, 2010

Acknowledgement

The encouragement, guidance and support of the project supervisors Prof. Andrew Clark, Dr. Ernest Foo and Dr. Ejaz Ahmed is humbly acknowledged and the sincerest gratitude offered for their enthusiasm, motivation and willingness to help complete this project.

Chapter 1

Introduction

There is a trend indicating the development of smart grid systems away from the current traditional SCADA systems. This report evaluates a number of currently proposed and existing test bed implementations. A set of brief reference requirements is extracted from the review process and each test bed implementation is compared against the reference requirements set to identify gaps. Based on the gap analysis conducted, a high-level test bed architecture is proposed which should meet the reference requirements specified. The reference requirements should not be considered as a comprehensive set of requirements, but more as a starting point. The proposed test bed implementation is realized using open source and freely available software tools and libraries.

Ideally, it would be desirable to have a complete realization of the test bed implementation, however due to project constraints, the scope shall be limited to implementing only a limited subset of requirements and producing a feature limited test bed. The developed test bed is intended as an emulator for part of a smart grid infrastructure to provide a means for evaluating and testing communication and protocol level security design, as well as providing a cost-effective means for running emulated simulations in a controlled non-production environment.

1.1 Background

The move towards an electrical smart grid introduces additional problems not previously encountered in traditional environment. To examine these issues an understanding of the landscape is required. Actual electrical power SCADA networks are discussed in detail in Section 2.1. Modern critical infrastructure, such as electrical power, gas and water supply networks reply heavily on supervisor control and data acquisition for the remote management of hardware

due to the geographically dispersed nature of the network. It's generally not feasible for utility operators to design and implement their own systems, thus they reply on external vendors to satisfy such requirements.

Vendor implementations of SCADA solutions have resulted in propriety architectures, as a result, the primary means of security previously had been security by obscurity. Focus was primarily on functionality, i.e. robustness, flexibility, and reliability and less focused on security. Propriety architectures had prevented attacks in the past due to a lack of knowledge of the system [1, 2, 3].

Previously, individual utilities owned and managed their own infrastructure and shared very little knowledge about them [4]. SCADA systems communicated on dedicated communications systems, such as private licensed microwave and Radio Frequency (RF) links or dedicated telephone lines, or even separate physical communications cabling. This limited connectivity and propriety nature provided security for SCADA systems [5, 2].

A number of SCADA implementations exist, but a number of de-facto open standard protocols have become widely accepted, for example Modbus in the last 30 years or so, and more recently DNP3. There is also the next generation of standards based protocols, such as IEC-61850 [6], based on the Common Information Models, which are gaining popularity [4, 7]

"*Not only are the protocols and hardware changing, but the communications links are evolving as well. Expensive dedicated phone lines and microwave links are being replaced by data networks*" [4]

Traditional SCADA implementations required physical access to the SCADA network to implement disturbances. Attackers needed access to the SCADA network or the field bus to implement exploits. With the increased integration of SCADA networks into corporate networks as well as the Internet, attackers are able to implement and realise remote exploits of vulnerabilities. Also with such integration the exposure factor of the SCADA network is also increased, since the integration introduces additional vulnerabilities present in the technology being integrates, e.g. ICT vulnerabilities, as well as vulnerabilities inherent as a result of the integration, i.e. protocol vulnerabilities.

Modern SCADA systems are increasingly using standard networks and open standards make SCADA environments more vulnerable. As a result, security models need to change to provide better security for SCADA systems. In the past a major issue with developing secure solutions for SCADA environment has been lack of proper tools to model and evaluate SCADA system security. Public utility SCADA systems are essential for providing services that are very important for modern daily life, and as such their protection and security is extremely critical and important. Attacks on SCADA systems present a serious

risk to the public health and safety [5, 1, 4].

There are some problems with performing security analysis and evaluation on existing SCADA systems. Firstly, it is not economical to set-up a SCADA test environment due to the scale of SCADA test environments, i.e. large number of distributed hardware [5]. Secondly it's not possible to execute security assessment on production networks due to their critical nature.

1.2 Motivation

Another problem associated with security analysis and developing appropriate security solutions for SCADA systems is a general lack of modelling tools to evaluate the security of SCADA systems. A SCADA test bed allows for the design and development of simplified models of SCADA systems to allow for security analysis and security solutions testing [5].

The importance of test bed implementation is clear with an increase in open standards for SCADA and smart grid implementation as well as a greater need to perform vulnerability assessment on these systems. Whilst there are a number of test bed implementation already available today, a majority of these are of a commercial nature and accessible only to a closed community of engineers [5, 1].

Simulations such as test beds may be used for operator training. Research indicated that operator errors are generally caused by time constraints and performance demands. Efficient recovery may be used to minimize losses from operator training [8].

Other benefits of using a test bed include;

1. An isolated, contained yet realistic environment for analysis without affecting real-world environment nor interference from real-world environment and larger Internet. This there-by reduces the risks involved with testing and analysis by isolating production environments during testing and analysis [5, 1].

2. Reduced system complexity, but captures complexities of a realistic environment using SCADA which presents a complicated process due to the complex hardware and software interactions [4].

3. Allows for the testing of new concepts and experiments with resource optimization approaches in smart grid environments [9].

4. Reduce costs for testing and security analysis and testing as opposed to the high implementation costs of test SCADA networks due to distributed nature of SCADA systems [5].

5. Reduced costs for training SCADA operators due to the costs involved with testing on a production SCADA network [8].

This it may be seen, that test beds are an essential component in developing security in systems as it provides an ideal tool for a non intrusive means to evaluate systems. Test beds provide an idea way to test and evaluate SCADA systems due to the scale and the critical nature of SCADA systems, i.e. testing, evaluation and security analysis can be undertaken without reservation and without affecting the performance and reliability of any production environment.

1.3 Project Objectives

Based on the background and motivation discussed in Section 1.1 and Section 1.2 respectively, this project is intended as a starting point for further research and development, particular for the use as a tool for security analysis and evaluation of emulated SCADA network traffic. Specifically the research project is intended to achieve the following objectives and research contributions

1. Investigate and review current test bed implementations and identify useful test bed characteristics

2. Identify and propose a reference set of test bed requirements that must be implemented to provide a comprehensive feature set.

3. Propose a high level software design for the implementation of a test bed, specifically for the introspective security analysis of network traffic of protocol messages.

4. Implement a set of reference requirements using the proposed high level software design as an informal starting point for security analysis of network traffic.

5. Provide some evaluation of the implemented design against the identified reference requirements, as well as identify and discuss any immediate issues in the implemented design.

1.4 Summary

To summarise, we have identified and discussed some of the emerging trends, and provided a high level background in an attempt to describe the current security landscape within the smart grid environment. The motivation for undertaking the research project have a significant security perspective, and is thus

considered to be particularly relevant to the volatile global environment. Further the achievement of the objectives identified should provide some important research contributions in the field of smart grid and smart grid security.

This research report provides a detailed background of SCADA systems and smart grid and examines the current and related work in the design of test bed identified during the literature review. A set of reference requirements are drawn from a number of sources, these are evaluated in discussed in detail, and a gap analysis conducted of the test bed designs against the proposed reference requirements. The gap analysis provides an opportunity to focus the design and implementation of the test bed on features that are lacking in other test bed implementations. As a result of the project the following outputs are produced

1. Reference set of smart grid test bed requirements
2. Gap analysis of current test bed designs against reference requirements
3. Proposed high level design of smart grid test beds
4. Design and implementation of QUT Smart grid Emulator test bed
5. Evaluation of the implemented design

The next chapter will be presenting the outcome of the literature review process. The chapter will provide an introduction to SCADA systems. The chapter will also provide an overview of smart grids. Since smart grids are an emerging technology, the current IEC and NIST models will also be examined and presented. As part of the literature review process related works will be identified and discussed in detail. The chapter will conclude with a discussion on any limitations of the test bed designs reviewed.

Chapter 2

Related Work

The related work is the outcome of the literature review process. The literature review process was used to establish a holistic understanding of SCADA systems. This is presented as an introduction to SCADA systems in Section 2.1, as previously mentioned in Section 1.1. The latter leads discussion into the modern smart grid approach, and is discussed in Section 2.2. The discussion on smart grid examines the two complimentary approaches based on the IEC Roadmap and the NIST framework. Next a number of current test bed designs and implementations are described in Section 2.3. This work presents an overview of the current work in test bed design and an analysis is conducted to identify perceived limitations, and summarised in Section 2.4.

2.1 Introduction to SCADA

SCADA (Supervisory Control and Data Acquisition) provides for remote control and monitoring of remote field devices. A SCADA environment consists of three primary parts, firstly the hardware in the field that needs to be controlled and monitored, for example, power generators, consumer appliances, feeder sub stations, feeder lines, reclosers, breakers, etc. and secondly the distributed network of telemetry devices that interface with the field hardware and present the data for monitoring and control functions to control actuators, and finally the SCADA application.

As illustrated in Figure 2.1, there are three (3) main features of a centralized electrical system, these are the electrical power generation, transmission and distribution networks. During power generation, electrical energy is created from other forms of energy, e.g. nuclear, solar, wind-turbines, hydro-electric turbines, fossil fuel generators, geothermal generators, etc. In some cases after production, a step-up substation is used to increase the voltage to levels suitable for

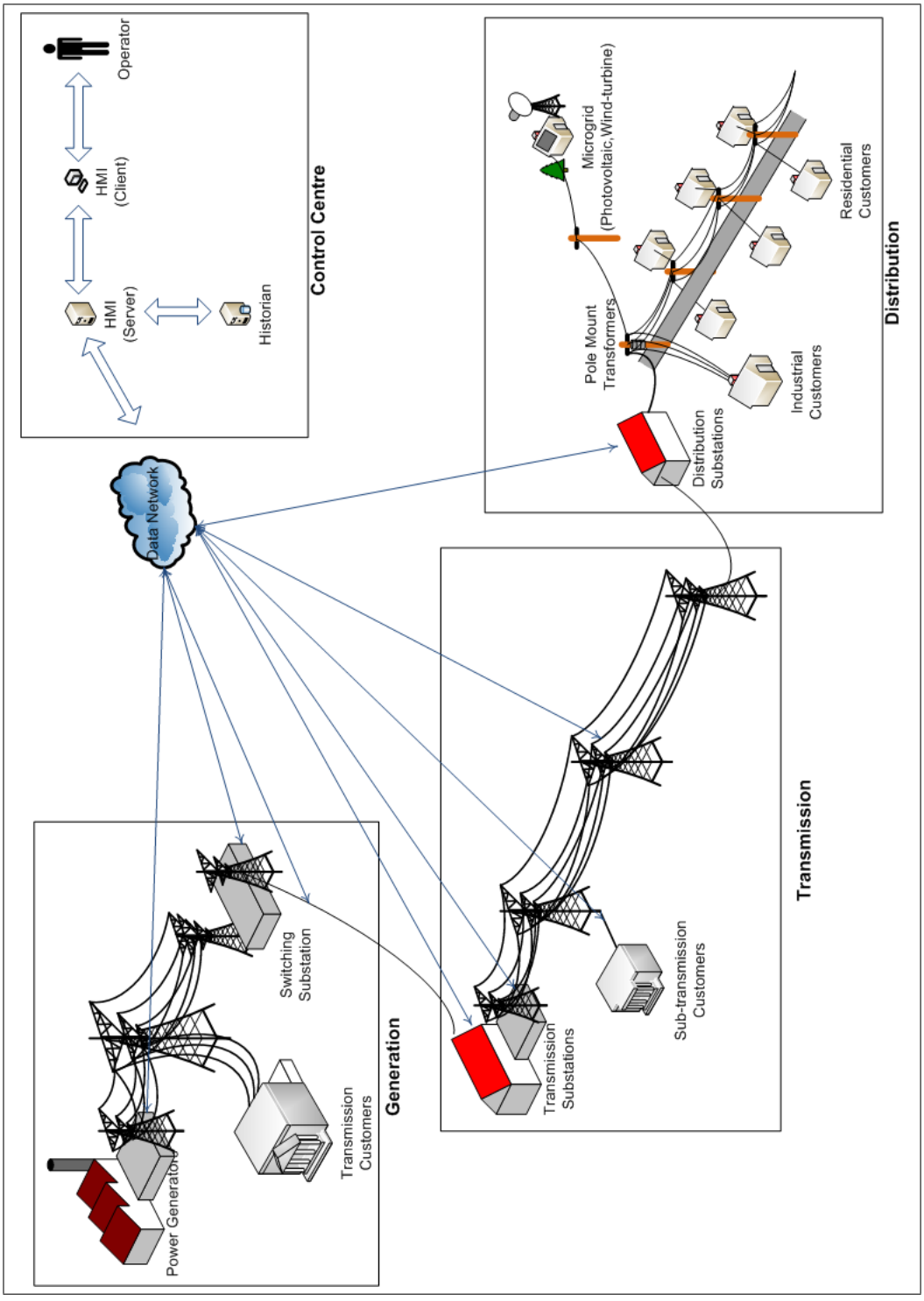

Figure 2.1: High-level overview of a SCADA network.

transmission. Voltage transformers use electrical current to manipulate electrical voltages, i.e. at step up transformers input electrical current is dropped to increase electrical voltage output, and similarly at step down transformers, input electrical voltage is dropped to lower output voltages at higher electrical current.

The bulk transfer of electricity is referred to as transmission. Electrical power is transmitted to transmission substations. The power transmission network works with high voltage (HV), i.e. generally between 69kV and 230kV or extra-high voltage (EHV), i.e. generally between 230kV and 350kV. Some extremely large utilities may also use ultra high voltage (UHV), i.e. generally greater than 350kV. Transmission is performed at such high voltages to overcome system loss. System loss is the loss of electrical energy as a result of transmitting over long distances. Electricity utilities may also wholesale some of the bulk electrical energy to transmission customers, such as other power utility companies, or extremely large industrial customers.

For the electrical energy to be delivered to consumers, a distribution network is used. From the transmission network, electricity needs to traverse through a step-down distribution substation. At the sub-station, the transmission voltage is reduced to levels to be used for distribution. The electricity traverses step-down transformers, which lower the voltage levels, then via bus bars, reaches industrial circuit-breakers, and then onto the distribution network. Depending on the voltages used and the distances transmitted, the step-down process may be repeated several times to convert from medium-voltage (MV), i.e. generally between 1kV and 69kV to low-voltage (LV), i.e. generally less than 1kV. On the distribution network, pole mounted transformers may be used to further step-down voltage to usable levels. The consumers tap off the distribution network for their consumption requirements.

Increasingly there are newer "prosumer" joining the electrical distribution network. These customers own an operate micro grid that are low production electrical networks that connect into the distribution network. The concept allows the independent production and consumption of power by individuals, and any excess electrical power not used by the prosumer is delivered into the distribution network for use by other consumers. A common example of micro grid is the use of photo voltaic cells on the roofs of domestic dwelling.

The overall performance of the electrical network is monitored from a control centre. The SCADA operator performs supervisory control functions for remote plant operations using a Human Machine Interface (MHI), this is also sometimes referred to as a Man Machine Interface (MMI). The HMI presents the remote telemetry reading to the operator and provides for control inputs from the operator. For audit and security purposes there may also be one or

more log servers (Historian) to capture and record event logs, telemetry reading, and control commands. The primary data acquisition is performed by one or more Master Telemetry Units (MTU), the MTU is responsible for the telemetry communications to and from the HMI and to and from the remote sites, i.e. Remote Telemetry Units (RTU). The RTU is responsible for acquiring the reading from the device under control, as well as sending control signals to it. The communication between the MTU and RTU may be implemented using a variety of networking technologies, such as microwave links (wireless), Public Switch Telephone Network (PSTN), fibre optics (fibre), local cabling (wired), or even the Internet. The latter is discussed further as an abstracted conceptual model in Section 3.3.

There is generally a low data rate requirement for communication between the MTU and the RTUs. The communication between the RTU and the hardware, i.e. the device under control usually requires very high data rates. Due to the critical nature of most SCADA systems there is a high level of resilience and robustness build around the architecture, e.g. one or more alternative components for redundancy and security, e.g. backups, duplicate and/or geographically dispersed systems.

A major challenge with electrical power generation is that once, electrical power has been produced there are limitations to storing it. As a result a complex system of monitoring and control is required to ensure that electrical power generation is able to accommodate demand. Although SCADA networks may be considered to be centralised, there is also some level of local automation, e.g. unmanned substations, etc. However, the task of monitoring and control overall is performed by a SCADA system. Traditionally, the SCADA application performed three (3) primary functions.

Data Acquisition and Control functions. To provide remote telemetry management functions, which are remote monitoring and control. The SCADA system monitors and controls field devices via analogue and digital input and/or output (I/O) "points". The data in communicated between the field devices and the SCADA system using SCADA protocols, which may be propriety to the solution vendor or open such a Modbus and DNP3. There are potentially over two hundred (200) SCADA protocols in existence. A SCADA solution may also comprise of several SCADA protocols which may need to be converted at some stage to a common protocol for uniform telemetry management. A primary characteristic of SCADA protocols is that they are provide deterministic performance, i.e. the time required for a protocol transaction to complete given other appropriate factors are known.

Data Communication functions. The SCADA system may communicate with actual field devices for monitoring and control access. This communication may be achieved in a variety of ways, for example previously there were dedicated microwave or radio frequency (RF) links, or even dedicated leased lines and fibre optic links. However more recently there is increased use of Internet service providers (ISP) to provide connectivity to field devices by transporting SCADA protocols over IP. Some utilities have even pilot tested power line carrier (PLC) technology for broadband communications.

Data Management and Presentation functions. All the data acquired on a SCADA network needs to be presented in a consistent manner to SCADA operators. SCADA user interface is commonly referred to as a Human-Machine Interface (HMI) or Man-Machine Interface (MMI). The data communicated over the communications network using SCADA protocols is represented in a meaningful way to the operators. Operators are able to use the HMI to execute Control functions on field devices, for example, tripping feeders, starting and stopping generators, etc. Any and all events within the SCADA network is logged for audit purposes into a Database, which is commonly referred to as the Historian. The SCADA software also performs alarm management functionality, where alarms are automatically triggered based on monitored conditions or specific SCADA events.

Smart grids are an extension to the traditional electrical SCADA system and are intended to provide greater control and intelligence in a distributed environment [10]. The smart grid thus adds additional characteristics to the traditional SCADA system. These characteristics are intended to enhance the efficiency and robustness of the traditional SCADA system. A detailed explanation of smart grid and the key characteristics of smart grid is provided in Section 2.2.

2.2 Smart Grids

Smart grids are intended to provide a more thorough monitoring and control of elements within the electrical grid to ensure efficiency [9]. "It can be said that the Smart Grid is the concept of modernizing the electric grid. The Smart Grid comprises everything related to the electric system in between any point of generation and any point of consumption" [11]. Therefore the smart grid may be considered to a modern electricity network including, but not limited to, the generation, transmission, and distribution networks as well as the consumer

premises. The main focus appears to be on increasing control of the electric power grid.

Energy utilities appear to be in a state of flux. In developed counties, utility networks have ageing infrastructure with steady or increased demand, whereas in, developing counties there is an expanding infrastructure and are facing increasing demands. These are drivers for promoting new methods of operating and managing electrical power systems. One such method is the increased use of information and communication technology (ICT) and techniques. Thus the need for *smart grids* [11].

Thus smart grids are the next evolution of electrical power supply and employs digital technology for bi–directional communication. Using smart appliances and the emergence of Home Area Networks (HAN) established an inherent bi–directional channel, and thus given the two–way communication channel in smart grid there is a greater requirement for information security, particularly relating to consumer privacy, as the customers are required to share more information with the electrical power suppliers [**?**].

The integration of ICT techniques and technology are intended to enhance the characteristics of existing SCADA network and provide additional benefits, both to the consumers as well as the suppliers. Some of the key characteristics are of smart grid are [12];

1. Greater security and resilience to cyber attacks.

2. Greater resilience to electrical power failure and better recovery from failure.

3. Optimization and operational efficiency of electrical grid assets.

4. Optimization of all electricity generation and storage needs.

Leading the work on standardising the smart grid and smart grid related technology are the IEC and the NIST. The IEC developed Roadmap and the NIST framework are discussed in detail in the following sections. Initially both groups had been working independently, however, recently there has been a move for greater collaboration to rapidly achieve the benefits from the standardisation of the smart grid.

2.2.1 IEC Roadmap

The IEC Smart Grid Standardization Roadmap Ed.1, put forward by the SMB Smart Grid Strategic Group (SG3) in June of 2010 acknowledges the need for standardisation the smart grid. At the time of writing this report, the IEC

claimed to be spear-heading a "comprehensive set of global standards" to support the requirements of the smart grid.

The IEC-SG3 is currently undertaking requirement and use-case acquisition for the development of a "Generic Reference Architecture" for smart grids, which would provide a natural extension to the current roadmap [11]. The primary outcome of the IEC-SG3 initiative is the development of action plans towards global standards to support smart grid requirements.

The roadmap acknowledges the complexity and dimension of the scope for smart grids, but suggests that given active engagement of stakeholders makes adoption possible, and previously mature standards and industry best within the traditional electric networks. Further it highlights, that standards developed in isolation present inter-operability issues as they may address the same issue, but at different levels [11].

The IEC-SG3 roadmap also proposes recommendations which include

1. Standardising interfaces to enable legacy systems, as well as providing product requirements.

2. Avoid standardising connection interfaces to be regulated by local governing authorities.

3. Seek cooperation with market stakeholders, who have already made significant progress in smart grid technology.

4. Seek cooperation with NIST and prioritise IEC responsibilities within NIST roadmap.

The document acknowledges that the "whole framework of IEC standards and a roadmap of further actions are defined in order to help the Smart Grid vision to become a reality" [11]. The primary focus of the standardisation being to add value to the smart grid implementation. Although it's claimed that there is a general lack of availability of literature relating to smart grid standardisation, the NIST Framework is highlighted as an exception, where a majority of the work as been driven the United States Energy Act. IEC proposes a more integrated set of standards extending beyond the NIST framework.

The roadmap also highlights drivers for the smart grid effort as well as current challenges facing smart grid implementations. IEC-SG3 suggest that the smart grid drivers and priorities differ across implementers, and goes on to highlight the NIST Framework Concept Model and uses the model to highlight possible drivers for smart grid implementation as part of the model.

Finally the IEC roadmap examines the existing standards and performs a gap analysis to propose new standards to be established. The procedure employed include

1. Capture function requirements

2. Evaluate impact of requirement on design

3. Identify and access high impact requirements

Although some requirements may already be government by existing standards, these were still considered in the analysis. The key requirement identified were

1. Interoperability (Communication)

2. Security

Interoperability is required at all levels of the smart grid, as a result the "increased exchange and automatic interpretation of information across all major domains of a future power net is therefore investigated in detail". Since the smart grid enable increased control of critical infrastructure "a new level of cyber security" [11] is required. The communications requirements to enable interoperability are based on a common data and protocol models.

The IEC road map identified requirements within the NIST framework and discusses any existing standards associated with the requirement as well as identified any gaps in the standards from a smart grid perspective, and finally puts forward recommendations for the requirement.

A recommendations arising from the gap analysis relating to the interoperability requirements include the need for mapping between existing standards to ensure sub-system communication. Further, it's recommended to make communication standard future-proof by ensuring that the communication standard is independent of the communication technology.

Security related recommendations included the use of granular access control to manage smart grid requirements. Use of smart grid segmentation to segregate smart grid domains, as well as enhancing the overall security of legacy systems which lacked security considerations during initial design and implementation.

A majority of the gaps and recommendations proposed were related to the planning of smart grid. However an interesting consideration in the document was that consideration of the IEC 62443 standard, i.e. Industrial communication networks – Network and system security standard of low relevance for smart grid. It's assumed that this is due to the lack of support of smart grid specific components.

Another claim supporting the need for data security within the smart grid is made within the IEC roadmap, i.e. "Smart grids and their sub-systems built on public infrastructures and open standards are highly vulnerable to security attacks and qualify as a critical infrastructure. Therefore adequate security

policies, mechanisms and algorithms must be implemented. In the context of smart metering, tampering with meters, illegal access to or falsification of billing and fiscal data and / or credit / debit data, illegal changes of configuration, illegal connection / disconnection of supply, denial of access can be mentioned" [11].

2.2.2 NIST Framework

Similar to the IEC framework, the NIST has been tasked with ensuring interoperability of smart grid devices and systems, and states "the expedited development of an interoperability framework and a roadmap for underpinning standards is fundamentally important" [13].

The NIST framework and roadmap plans identified eight (8) priority areas for investment. Of particular interest to this research are cyber security and network communications. Cyber security deals with the traditional security goals of confidentiality, integrity and availability within the smart grid, and network communication areas deal with the wired and wireless infrastructure environments as well as appropriate access controls which are critical to smart grid

Standards are key to the enablement of interoperability. As such mature standards are seen as a market driver to allowing multiple vendors to mass produce standards compliant products for implementation and integration into the smart grid. A systematic approach to standards maturity also ensures appropriate investments for sector growth. NIST's focus appears to be on developing appropriate frameworks to enable to testing and certification of standards compliance.

NIST has identified a number of domains and actors within the smart grid space and proposed a conceptual model. The conceptual model is intended to ensure appropriate "planning and organization of the diverse, expanding collection of interconnected networks that will compose the Smart Grid" [13]. The seven (7) domains are:

1. Customers – consumers and pro-sumers, may be considered to be residential, commercial or industrial.

2. Markets – operators of electricity markets

3. Service Providers – providers of service to customers and electrical utilities

4. Operations – electricity movement managers

5. Bulk Generation – electricity generation

6. Transmission – long-haul electricity carriers

7. Distribution – electricity distribution to or from customers

Also underlying the identified domains is a framework for regulatory governance. To achieve the objectives of the smart grid, there needs to be managed interaction amongst actors within as well as across the identified domains. In some scenarios the domains and the actor roles may overlap depending on the dimensions of the smart grid infrastructure and deployment, i.e. smaller markets may have a single actor who is present in all domains, e.g. Fiji Electricity Authority in Fiji is solely responsible for bulk generation, transmission, distribution and is also the technical regulatory body in Fiji.

The proposed NIST conceptual model is intended to be used as a tool for regulatory policy constructions, as well as to ensure current and emerging standards and protocol compliance to ensure consistent smart grid implementation. The conceptual model also proposed possible communication channels within a smart grid infrastructure.

NIST also examines existing standards and tabulates its application the smart grid. The standards coordination effort is prioritised based on some key requirements, which may be re-defined as the standards definition efforts proceed. Several Priority Action Plans (PAP) are defined, which include high level details of requirements as well as justification for the PAP. The NIST PAP also provides listed objectives of the PAP as well as the project team responsible for the PAP.

The NIST framework and roadmap also dedicate an entire chapter to Cyber Security Strategy. The focus is not only directed at deliberate disturbances, but also at security in relation to accidental disturbances such as user error, system malfunction, natural disasters, etc. Additionally NIST also identified a number of additional risks inherent in smart grid implementations, specifically those relating to increased complexity, common vulnerabilities, communication disruptions, greater exposure to potential attackers, increased impact of exploitation. Alternatively with greater integration it is intended that smart grid may improve reliability, security and efficiency of traditional electricity network SCADA system as well as provide dynamic optimisation of operations and resources [13].

2.3 Test bed Designs

The discussion in Section 2.1 and 2.2 provides a reasonable foundation for the understanding of the literature reviewed to examine the existing test bed designs and implementation herein. Further the examination of the IEC and NIST

documentation has provided a clear direction in which the smart grid movements are headed in. The latter provides the context for the remainder of the project. A number of test bed designed have been proposed and developed. These test bed are examined in some detail and evaluated qualitatively against the IEC and NIST context to form a basis for a gap analysis and to further provide a baseline for a comparison between the test bed implementations examined.

The results of the review of the test bed design allows conclusions to be drawn about the limitations of the individual test bed. The test bed limitations and the results of the qualitative evaluation are discussed in Section 2.4. Further, this review provides a significant contribution to the project as it provides a basis for the development of the reference requirements set as well as the gap analysis to provide a comparison between the test bed implementation as well as highlight areas not adequately addressed by the test bed designs and implementation. These are all part of the project outputs as previously identified in Section 1.3. The actual requirements and gap analysis are detailed in Sections 3.1 and 3.2 respectively.

2.3.1 RMIT

Proposed an open source test bed solution, due to the limited access associated with propriety test bed solutions.The intended design is to emulate a real SCADA network including functionality to monitor and control actual hardware sensors and actuator hardware. The proposed test bed used a layered design model.

The design incorporated use of actual hardware sensors and actuators to be able to clearly identify and evaluate the effects of attacks on real-world hardware. The overall architecture was based on a real SCADA network. The design incorporated RTUs, sensors, actuators, MTU, HMI server and HMI client.

Hardware layer abstraction is implemented using a proxy for the actual hardware, that mimics the state of the field devices. The RMIT test bed also provided actual field device hardware simulation using Lego Mindstorm hardware.

The HMI client was implemented as an external stand-alone application, which was represented in the simulation, along with the physical sensors and actuators using software proxies, which communicated via the common gateway.

The RMIT design used a network simulator implemented using OmNet++. OmNet++ (http://www.omnetpp.org/) is a C++ library that implements a discrete event simulator. The MTU/HMI functionality is implemented as modules for OmNet++.

The actual SCADA protocol used in the test bed was Modbus, which was transported over TCP, hence the libmodbus (http://copyleft.free.fr/wordpress/index.php/libmodbus/)

library, written in C, was used for its implementation. Modbus was selected as its claimed to be a widely adopted protocol for Critical Infrastructure (CI) implementations [5].

The Modbus TCP implementation is transported using TCP/IP. The MTU (master devices) communicated via socket connections to RTU (slave devices) on port 502. Each message contains a single Modbus Protocol Data Unit (PDU). The RMIT simulation uses the INET framework for TCP/IP implementation, which was modified to be retrofit into the OmNet++ engine. The INET framework was also modified to limit the number of simultaneous TCP connection to assist in the demonstration of a DoS. This was achieved using the open source Rease project.

All communication for the simulated hardware traversed through the gateway which relays the information to the simulation proxies in standard Modbus messages, however the reverse is implemented using a custom communication protocol called ExtComm, which the gateway employs to send messages to the hardware proxies.

The HMI client also communicate to the HMI server via the gateway, which sends the messages to the HMI server via the HMI client proxy, and follows the reverse communication path when the HMI server needs to communicate with the HMI client.

Using the RMIT test bed simulation the team was able to demonstrate the effects of a DoS attack. In particular the RTU was flooded with TCP SYN packets. Future work on the simulator include the emulation of other SCADA protocols such as DNP3.

2.3.2 SAP

Karnouskos and Holanda present a realistic agent-based smart-city micro-grid simulation. They proposed a real-time dynamic smart grid in an effort to analyse, design, implement and evaluate an approach to simulate a dynamic infrastructure.

Some of the design requirements put forward included, almost real-time simulation of energy consumption and production, ability to change device behaviour by administering control functions, generate real-time reporting, as near as possible to real-word device behaviour. The team used JADE to implement software agents in the MAS. MAS was used to allow for intelligent and distributed management. Each electrical energy device was mapped to a single instance of an agent. The MAS design was FIPA compliant and the JADE web interface was used to provide external access control.

The design used a business approach to modelling and used a business layer

for communication with web services, a middle logic and strategic layer for rules and configuration management and communicated with the agent simulator, and finally a simulator layer that contains all the simulated agents. Scenario complexity is realized with the agent communication to achieve predetermined objectives.

User interaction with the agent variables enables real-time scenario changes and enables the implementation and execution of multiple scenarios independently or in parallel. Data is stored in an audit database and can be used for further analysis. The simulated agent relationships and behaviour was implemented to match real-world structure and behaviour as closely as possible.

The consumption modelled by the simulated agents was somewhat limited and lacking in that they only maintained a binary state of ON or OFF. However their behaviour was government by individual device profiles that were defined using energy profiles from the Residential Energy Consumption Survey (RECS) 2001 by the United States Department of Energy (US DoE). The survey did not provide details of the usage of the appliances.

The evaluation consisted of running a simulation of 3 smart cities, with 300 house-holds using various appliances, and 52 vehicles per city. Each appliance event was recorded in the database. An energy controller agent was used to monitor the electrical power generation and consumption to ensure that they were within tolerance. If the limits were exceeded, the control agent would realize control actions, e.g. turn off devices due to excessive consumption or start generators. Ironically instead of reducing power production, the control agents turn ON devices to handles cases where power production exceeds consumption.

Future work includes the expansion of the appliance profiles to include detailed consumption profiles e.g. winter-summer, workday-weekend, etc. Introduction of new appliance states is also planned.

2.3.3 Hannam

The Hannam test bed is focused more on proving vulnerabilities rather than providing a platform for design, analysis and testing of vulnerabilities. The paper was aimed at analysing vulnerabilities of SCADA systems by virtual scenarios, design and implementing a test bed to verify the vulnerabilities and provided recommendations for possible solutions to mitigate the vulnerabilities.

The Hannam test bed design for vulnerability verification assumed attackers gained access to internal network. The test bed design focused on attacks focusing on traffic interception using packet sniffing. For the actual SCADA protocol the design used Modbus protocol over RS232 using personal computer serial ports, and used RTU mode Modbus to utilize the higher efficiency over

ASCII mode Modbus.

Various attack scenarios were discussed in detail, as well as discussion of open tools available to attackers, such as nmap for port scanning, nessus for vulnerability checking, wireshark for network traffic analysis, and several others. There is a brief discussion in regard to network traffic analysis in a switched network environment using techniques such as ARP redirection, ICMP redirection, as well as switch-port SPAN monitoring. The attack scenarios proposed network analysis between Modbus frame transmissions, as there is a brief delay between each frame.

The proposed test bed consists of a Cimon or Autoeye SCADA server, a Moxa RS485 multi-port unit, RTU and IED's, simulated digital and analogue I/O, and the Modbus protocol stack. The proposed security solution for the test bed involves the use of a proposed Security Device Prototype module. The Cimon (SCADA server) server communicated via the Moxa, the Security Device Prototype Modules are used to ensure security on the Modbus network and communicate with the Security Device Prototype Modules on the RTUs and IED's. The IED's and RTUs receive simulated digital and analogue I/O. The Security Device Prototype Module used SEED, which is a block encryption algorithm for symmetric key encryption and decryption within the telecommunications networks. Proposed future work includes detailed investigation of hacking tools.

2.3.4 OSECS

Open Secure Energy Control Systems (OSECS) in conjunction with the United States Department of Energy and the United States Department of Homeland Security proposed an open-source toolkit based on the IEC-61850 protocols and later integration of IEC-61400-25, for wind power control standards.[7] Klein claims that IEC-61850 may be considered the core technology for smart grids. The toolkit it intended to build products to provide a means of assessing the benefits of IEC-61850 product. Specifically security is addressed using role based access control (RBAC) as provided by the protocols. The IEC-61850 provides an XML based Substation Configuration Language (SCL).

The purpose of IEC-61850 is to provide modern information and communication technology automation to legacy electrical networks. IEC-61850 replaces traditional systems architecture with an object based one, for example instead of using points for data acquisition and control, it used object names, part of the object names are standardized, and the other part is customizable to the implementers requirements. Other benefits of IEC-61850 include the use of layer network infrastructure, instead of traditional point-to-point wiring, which en-

ables the use of conventional security tools, such as encryption, firewalls, and others.

Klien also briefly discusses the slow acceptance of IEC-61850 within the unit states, and the rapid acceptance and planned implementation world-wide, and attributes it to the turn-key solution adopted for substation commissioning around the world.

The OSECS model uses the IEC-61850 engine and Manufacturing Messaging Specification (MMS) stack as its core, and provides security using RBAC for message passing through the engine. All messages are expressed as (Extensible Mark-up Language) XML and converted to MMS or Simple Object Access Protocol (SOAP) for communications, and processed by the web engine. The engine refers to a database for RBAC security. The model essentially created two secure networks, once including the control centre and corporate facilities, and the second including the control centre and substation equipment.

The OSECS implementation of IEC-61850 was achieved using open source tools and utilities, and released the developed components under an open source license as well as a commercial license to cover propriety products used in the development.

2.3.5 Berkeley, Mellon, and Vanderbilt

Giani, et. al. [1] have provided a discussion paper on a proposed test bed implementation. The test bed architecture based on a reference model very close to real-world implementations and present a number of realization scenarios. The paper provides a brief overview of the overall SCADA architecture, and discussed three different realizations of a simulated test bed environment, as well as the discussing tools and libraries which may be suitable for each implementation.

The paper discusses proposed test bed designs to uses the Department of Defence High-Level Architecture (HLA) with realistic hardware and network simulation using EmuLab. The rational behind such an approach is to provide highly accurate, realistic and scalable solution.

The planned experimentation on the test bed include attacks such as Denial of Service (DoS) attacks, Integrity attacks as well as Phishing attacks. They attempt to provide a framework for a realistic implementation by using actual SCADA devices along with SCADA software running on realistic hardware, whilst using network simulation and real-time plant simulation.

At the time of writing, work had commenced on a basic single simulation-based implementation, as well as future emulation and instantiation based version of the test bed. The test bed implementation uses commercial SCADA

RTU and commercial SCADA software.

Future work includes improvements to the current implementation, high-level attack scenarios and finally migration from single simulation based solution to a full instantiation based model.

2.3.6 Virginia Tech.

The test bed focus on an design and implementation for an Intelligent Distributed Autonomous Power System (IDAPS), which is a distributed smart grid concept proposed by Virginia Tech Advanced Research Institute. The proposed test bed is a Multi-agent System (MAS) where multiple agents work in collaboration to achieve smart grid system objectives [12]. In a MAS the agents work autonomously and achieve collaboration by exchanging data using an Agent Communication Language.

In the implementation of the test bed the agents were intended to be wholly autonomous. The agents are required to perceive their environment, react to changes in it, and communicate with other agents. The paper discusses the design of a MAS system for IDAPS and specifically to work in collaboration to detect electrical network outages and react accordingly to allow operation of part of the electrical grid in isolation.

The proposed design is specifically intended to handle the operations of the IDAPS micro-grid in relation to outages from its upstream commercial electrical supplier. The micro-grid consists of electrical power generators and controllable loads. In the event of an upstream outage, the micro-grid should activate internal electrical power generation to secure critical loads, which are identified by a predefined priority list. This thus implements the key characteristics of a micro-grid, which are intelligence, distributed architecture, and operational autonomy.

The proposed MAS architecture for the implementation used four (4) types of agents to abstract the complexity of the system, as well as reducing the number of agent messaging. Each agent has a unique function. The actual agent implementation is done using Zeus, which is an open source, FIPA compliant, Java based toolkit.

In this implementation, the MAS is specified using the following agents;

1. Control Agent (CA) - the CA specified agent roles and responsibilities, including monitoring the system and detecting failures, as well as providing control functions to provide micro-grid isolation.

2. User Agent (UA) - the UA provides an interface to the IDAPS system to the user. The responsibility of the UA is to provide real-time information

regarding the IDAPS, provide customer information such as consumption, nature of load, e.g. critical/non-critical, as well as providing control the status of the load based on priority.

3. Database Agent (DBA) - the DBA is responsible for storing system information to ensure that the system remains in a known state. The DBA also persistifies agent communication and provides a data access point for agents.

4. Distributed Energy Resources Agent (DERA) - the DERA is responsible for storing DER associated information, such as, the DER ID, power rating in Watts, type of source e.g. solar cells, micro-turbine, generator, etc. functional costs, fuel availability, as well as DER status and availability.

During initialization each DERA and UA update the DBA, by providing its ID and IP. The DBA then initiates agent registration and provides an associated CA for the DERA and UA. The CA responds to the registration requests. The CA receives updates and publishes to its registered agents. During failure the CA sends control messages to its registered agents to take appropriate action. The UA sends power load requirements from appliances and sends commands to the DERA. The DERA updates UA with electrical power production data. The DERA and UA react to environmental changes. A visualizer collects messages exchanged by the agents for display

The IDAPS and MAS simulations are run on separate Personal Computer (PC) hardware and communicate via a middle-ware TCP/IP server written in Matlab/Simulink to allow multiple TCP/IP connections to the IDAPS simulator.

The tests run in the IDAPS MAP indicate that the MAS is able to respond in a time efficient manner and effectively isolate the micro-grid following an upstream electrical power supply failure. The results reinforce that software MAS may be a suitable substitute to previous hardware based zone based protection systems [12].

2.3.7 Illinois

Presents the design and development of a test bed for the assessment of vulnerabilities to SCADA systems as a result of using public networks for communication and new technologies. There is increased use of Intelligent Electronic Devices (IED) which are intended to allow for more efficient network communication and utilize next generation SCADA protocols and implement more intelligent behaviour. However using "*standard networks and protocols opens the devices to possible attacks*"[4].

The authors argue that using hardware to perform vulnerability analysis is not an economical pursuit due to the complex hardware and software interactions involved, and suggests a solution may be to constructs a less complex system to capture the general system complexities and operation such as a test bed.

The paper briefly discusses threats, vulnerabilities and countermeasures and the role utilities need to pay to secure critical architecture as mandated by government authorities within the United States of America.

The simulation architecture is intended to simulate the real-world environment closely. The simulation architecture consists of;

1. PowerWorld Server (Server)
2. Network Client (Client)
3. Custom Networking Protocol (Client-Server Protocol)
4. Network Emulator (Emulator)
5. Protocol Converter (Converter)

The Server acts as an interface for a real-world electrical power grid and provides a simulation of the electrical power grid using highly accurate modelling facilities. The server emulated a SCADA network by generating SCADA data for the clients. The server accepts control commands from clients, as well as propagating environment changes to clients

The purpose of the Client is to provide functionality such as the graphical user interface to emulate the HMI of a SCADA system. The client communicates with a server using TCP/IP to populate and update the HMI. The client also provides control functionality to allow control of electrical power system elements such as feeders.

The custom Networking Protocol (Client-Server Protocol) is used the client and server for communication. It is a simple request-response protocol that run on top of TCP/IP.

The Network Emulation is performed by the Real-time Immersive Network Simulation Environment for Network Security Exercises (RINSE). RINSE is used as it provides realistic simulation of large networks as well as being able to emulate test scenario's such as attacks, defences and network transactions. RINSE is able to run simulations in parallel in varying detail. Another reason stated for selecting RINSE was the ability to integrate real hosts for integrating actual traffic together with the RINSE simulated traffic.

The Protocol Converter is a program designed to convert the Server (PowerWorld Server) protocols to actual SCADA protocols. The Converter is required

to enable Clients to interface with actual SCADA hardware. For the purpose of the test bed the Modbus SCADA protocol was selected. A free open source Modbus implementation was used for the Converter.

The implementation included setting up proxy servers to provide translation of virtual IP addresses on the simulated network to actual IP addresses. The real client would send a request, the request would arrive at the proxy server, the proxy server would then translate the address and inject the traffic into a VPN tunnel was used to accept the packets for injection into the RINSE simulator, after the RINSE simulation, eventually the traffic would reach the virtual server, and then egressing the RINSE network via another VPN tunnel to the server proxy, which would then translate the IP address to that of the actual server. The response traffic from the server to the client would follow the reverse path.

The purpose of injecting the traffic into the RINSE emulator was to enable the examination and observation of simulated attacks on the simulated network. In the implementation the RINSE network is completely transparent to the client and server. Several test scenarios can be emulated on RINSE without affecting the implementation of the client and server.

As a result of the test bed and the experimentation, they observed a denial of server (DoS) attack, which clearly illustrated that the client-server communication is disrupted. The DoS also renders the system in an unknown state because the client continues to display information that is out of date and may not be a true reflection of the actual real-world state of the system. Work is continuing to extend the test bed to include additional functionality for the network client and server.

2.4 Test bed Limitations

The test bed designs and implementations reviewed as part of this project, as detailed in Section 2.3 appear to only partially fulfil the requirements outlined in the IEC and NIST guidelines and general test bed design. A major concern is that there are no single test bed design that address both the traditional SCADA system as well as the modern smart grid approach to electrical infrastructure management. While it may be accepted that smart grid infrastructure would eventually replace traditional SCADA technology, it would be unreasonable to assume that this would occur at a rate such that traditional SCADA systems would be rendered obsolete over-night. As such it's important to allow for a test bed design and implementation that would accommodate both architectures as well as allow for a transition from the traditional SCADA approach to the modern smart grid approach of management.

There is also an alarming lack of actual network emulation technology in

the design and deployment of the evaluated test bed architectures. The designs appear to have discrete event network simulators, such as Omnet++ and real-time network simulators such as RINSE. Network simulators provide abstraction of the Network Operating System (NOS) and the Networking Protocols into a simulated model to provide statistical analysis of the network performance. Network emulators on the other hand emulate the actual network protocol stack of network connectivity devices, such as routers through some form of virtualisation [14]. It would be naive to expect full network support in test bed implementation, however some form of actual network emulation is required to provide realistic network behaviour for appropriate security assessment, e.g. acquisition of network traffic, etc.

There is no realistic modelling of electrical consumption. There has been some limited modelling, but they fail to incorporate variable usage patterns such a during summer and winter periods, day-time and night-time consumption, week-day and week-end consumption modelling. For a test bed simulation to be accurate there is a need to accurate modelling of electrical power production and consumption. Current models only provided limited scope and modelling, i.e. only provided constant consumption based on binary ON/OFF states. Such modelling would be essential for testing and evaluating actual smart grid characteristics such as load management, self healing, etc.

Additionally there were some software licensing issues which may limit the application and acceptance of some test bed designs. Some components of the test bed implementation were available freely but only for academic and personal use, for example the OmNet++ libraries. If the solutions were released as open source products, there may be greater acceptance and application as users would be able to freely modify and adapt the solution for specific requirements without worrying about software costs associated with commercial licensing. Such use would also produce test bed enhancements that are well tested applicable to user requirements.

2.5 Summary

Based on the literature review conducted it is evident that there needs to be a more consistent approach to the design, implementation and evaluation of test bed. The literature review was used to illustrate an overview of the SCADA and smart grid systems, as well as identify and review related work in the design and implementation of test beds. This naturally required a qualitative comparison of the designs which was discussed previously. The need to address the limitations in the proposed smart grid test bed architecture was identified. Following a detailed evaluation of the test bed designs reviewed and using a detailed set of

requirements for smart grid test bed the limitations were analysed, to address the limitations.

In the next chapter a smart grid test bed design will be proposed. The proposed design will be a result of the initial requirements specification, which will be detailed in the next chapter. The requirements will be drawn from the IEC and NIST documentation as discussed in the previous chapter. The next chapter will also be presenting a discussion the results of a gap analysis conducted using the requirements to be specified and the test bed designs identified during the literature review process.

Chapter 3

Proposed Smart Grid Test Bed Architecture

The proposed smart grid test bed architecture consists of a detailed set of requirements extracted from the IEC and NIST guidelines. The approach to determine the requirements for the test bed was a bottom up one, i.e. requirements elicitation, was performed starting from the distribution network and traversing the SCADA network, as illustrated in Figure 2.1, towards the control centre. This approach was used, as it started with the part of the SCADA network where most of the smart grid focus has been. For each requirement specified a qualitative evaluation was performed against each reviewed test bed to determine the extent to which the test bed design addressed the requirement. The evaluation is part of the gap analysis. The results of the evaluation is summarised in Table 3.1. The table forms part of a gap analysis which has been identified as a project output in Section 1.3. The gap analysis conducted provides a significant contribution to the research as it allows the identification of inadequacies in the current related work, and allows the opportunity to address such inadequacies in the proposed design.

3.1 Requirements

The requirements are determined using a bottom–up approach as described in Section 3. Once a requirement is identified, it is discussed as being in two main categories, either a functional requirement or a non-functional requirement. Functional requirements are generally understood to refer to those requirements that specify a function or specific behaviour of the solution. Non-functional requirements on the other hand usually refer to qualities, such as

constraints, compliance and other behaviours of the solutions.

The following high level requirements have been identified

- R1 – Smart appliances are modelled
- R2 – Multiple real-world energy consumption profiles supported
- R3 – Hardware interface or integration
- R4 – Real-time smart grid emulation
- R5 – IP based communication
- R6 – Full SCADA solution
- R7 – Support for traditional SCADA as well as smart grid
- R8 – Design based on NIST concept model
- R9 – Dynamic test scenario manipulation
- R10 – Full audit trail implemented
- R11 – Graphical User Interface
- R12 – Test bed readily available to greater community

As previously mentioned, these requirements have been drawn from a review and analysis of current works relating to SCADA and smart grid test bed designs and implementations, and the IEC and NIST guidelines on smart grids. The works were reviewed to determine available, as well as proposed, functionality and system capabilities. Additional project team expectations of a test bed were integrated as requirements. Whilst it is not a purpose of this project to duplicate the features and functionality of already existing test bed solutions, it it however a project driver to make such features freely available to the larger community for use. Additionally the set of proposed reference requirements has been identified as a principal project output in Section 1.3.

R1 – Smart appliances are modelled. This is considered to be a functional requirement. For the home area network emulation, the test bed should implement appliance emulation, for example modelling of actual appliance hardware should be undertaken to ensure future integration into smart grid, i.e. modelling of smart refrigerator, smart air conditioning unit, smart meter, etc. so that the smart grid test bed can be used to manage such home area appliances for a more comprehensive emulation of the smart grid. According to the NIST framework, the HAN appliance

communication should be standardised by the year end, 2010. Such standards are important to enable the development of smart devices to foster innovation and realise smart grid potential.

There is a need for modelling smart appliances to be able to test the functionality of such devices, not only for technical reasons such as security assessment purposes, but also for business reasons, such as to ensure that smart appliances are able to respond to price and demand signals to ensure that smart grid objectives and characteristics, such as those defined in Section 2.2 are realised. Further, this is important to be able to demonstrate that at the consumer level smart grid capabilities can be realised and the benefits extracted from the use of smart grid technology, since this is the part of the smart grid the average consumer will interface with.

Of all the test bed designs identified in the literature review, only the SAP test bed [9], provided an actual appliance within their test bed. The modelled appliances represented common house–hold white goods, and provided simple live–dead state, that was managed using an internal schedule or an external command. The actual appliance consumption was based on the Residential Consumption Survey by the US DoE. The SAP design also proposes future work to enhance the consumption modelling to better emulate real world consumption.

R2 – Multiple real-world energy consumption profiles supported. In addition to the smart appliance modelling, in the smart grid test bed, there should be some form of modelling of the distribution consumption to emulate consumer behaviour within the electrical network, i.e. the test bed implementation should model varied consumption, e.g. there should be modelling of day–night consumption cycle, summer–winter cycle, etc. This requirement relates to the modelling of smart appliances, and would be useful for testing smart grid functionality, such as self healing, recovery, load balancing, etc.

None of the test bed designs reviewed support this requirement fully. The SAP test bed [9], provided partial compliance to this requirement, as it used internal time schedules to emulate consumption. However the model used for the simulated consumption does not appear to be complete and does not address consumption cycles such as those identified here.

R3 – Hardware interface or integration. This is a functional requirement. The test bed is expected to provide actual hardware integration or at least an interface to be able to integrate with actual hardware. This requirements is considered important as it will provide a realistic implementation

of the test bed. The test bed must at the very least implement some level of simulation of SCADA hardware to enable real time interpretation of the application level emulation. This requirements may be further extended to provide integration of the test bed in actual hardware based test bed environments.

Using physical hardware within the test bed will enable the evaluation and testing of real time characteristics. Additionally this this requirement will also enable the testing of potential hardware, i.e. the test bed may be useful for testing new smart appliances or traditional SCADA hardware within a controlled environment with impacting external users. This also enables hardware testing without the need to set up and manage a hardware test bed environment.

Only two test bed implementations reviewed provided some form of hardware integration. The RMIT solution incorporated Lego Mindstorm hardware into their test bed to provide control system functionality. Since the hardware uses propriety communication protocols, a wrapper was designed to convert conventional SCADA communication protocols to the propriety protocol. The UC Berkley, Carnegie Mellon University and Vanderbilt University, test bed design also makes provisions for hardware integration. Hardware integration or at least hardware emulation is an essential component of a test bed as it provides mechanisms for realistic and scalable evaluation of smart grid systems.

R4 – Real-time smart grid emulation. This is essential for evaluating time specific smart grid functionality. SCADA systems require high speed low latency communication between the device under control and the RTU. However since the test bed is only intended as an emulator, there may be some concession with this requirements, i.e. since the test bed may be deployed on a non real time platform, near real time performance may be suitable for the purpose of evaluation and assessment under test bed conditions, i.e. soft real time constraints may be sufficient. This is another functional requirements. Without the use of real time operating system, it may not be possible to get the deterministic performance required of SCADA or smart grid systems. All test bed designs appeared to support this non-functional requirement, although none made recommendations for execution on real time operating platforms.

R5 – IP based communication. To enable distributed use and remote access. IP based communication should be used between all major components of the test bed. This is an essential requirement not only for

correct emulation of the current generation of smart grid network but also to enable distributed emulation, i.e. components of the test bed may be implemented and shared from geographically diverse networks to enable better utilisation of resources.

This would enable not only a collaborative test bed development and utilisation but also foster innovation. The need for IP based communication is required not only from a technical implementation perspective but also to implement the functional and system management requirements. Specific smart grid goals and applications as identified in the IEC standardisation roadmap [11]. Although there are no specific references to IP based communication in the roadmap, references are drawn in the IEC roadmap to illustrate that a secure, reliable and economical electrical power network is dependent on a responsive, and reliable communications network. Even the NIST framework documentation [13] acknowledges the need to standardised network communications.

IP based communication is inherent in smart grid test bed, as a majority of network communication paths in actual smart grids are based on IP networks. Thus smart grid networks are given greater flexibility, but also introduced to higher vulnerabilities that exist in IP based networks. Using IP based communications in smart grid test beds would also enable the rapid prototyping and assessment of IP based attacks and vulnerabilities. For the purposes of this project this may be considered a functional requirement.

R6 – Full SCADA solution. In addition to the actual emulation or integration of SCADA hardware as previously identified as requirement R1, the test best should provide a complete suite of features to enable the implementation of a complete SCADA solution, i.e. the test best must implement an end–to–end SCADA system to have a human machine interface, historian database, complete audit log, SCADA protocol implementation, etc. This is to ensure that the application of the test bed is not limited to the analysis of security issues, but can also be adapted for the emulation and testing of different events, including the training of SCADA operators.

In addition to the identified uses of the complete smart grid test bed, the IEC document [11] also discussed communication standards for connecting to overall SCADA systems, e.g. IEC-61850 [6]. Even the NIST framework [13] provides discussion on using SCADA in smart grid for Operations domain applications. As mentioned previously work was being done on the Berkley test bed to ensure full instantiation of the SCADA model, further the RMIT also provided modules for HMI implementations, etc.

and thus provided a full SCADA implementation. The Illinois test bed is considered to provide a partial implementation of a complete SCADA implementation, since it only uses a PowerWorld Server to simulate a power grid, instead of emulating one with actual protocols. Whilst the simulated environment does provide the necessary features for some testing it cannot be extensively used to emulate an actual physical environment.

R7 – Support for traditional SCADA as well as smart grid. Since smart grids are an amalgamation of traditional SCADA technology and modern ICT technology it should be considered as core functional requirement that any implemented test bed must support both implementations, i.e. the traditional SCADA protocols must be implemented to be able to perform analysis on protocol related security assessment, and secondly the test bed must support the implementation of smart grid intelligence to realise the benefits of smart grid, such as self–healing, rapid recovery, etc. as detailed in Section 2.2. Further the test bed must also implement modern ICT topology, such as IP capabilities. The latter is essential for performing network related security and performance analysis. There is also a requirement for IP version 6 support as service providers migrate to the new platform.

Unfortunately none of the test bed reviewed supported both implementations. They either implemented a SCADA test bed or one that implemented only parts of the smart grid, e.g. the intelligent power routing, etc., but not a full environment. Whilst it may be acceptable to test certain aspects of the environment in isolation, to enable contextual assessment, an implementation of the smart grid technology as well as the underlying SCADA structure is required.

R8 – Design based on NIST concept model. The NIST concept reference model [13] provides a comprehensive high level design of smart grid environment. The NIST model may be used as a baseline for the design and implementation of test bed to ensure a comprehensive feature set that will provide a holistic test bed environment, which in turn would lead to more realistic test scenario development.

The NIST model defined not only the technical and operational domains such as parts of the electrical power network, e.g. generation, transmission, distribution, but also business domains, such as markets, service providers, etc. The inclusion of these domains within the test bed is an important aspect of realising the benefits of smart grids as discussed in Section 2.2. Without the inclusion of such domains into the test bed it

would be difficult to use the test bed to emulate smart grid outcomes, e.g. not being able to integrate to enterprise resource planning applications, or even assessing the performance of the smart grid given different markets.

Again, unfortunately none of the test bed designs reviewed provided this feature. The focus of the test bed appeared to be primarily on technical aspects of the smart grid, and thus design and implementation efforts focused primarily on this. This may be considered a low priority requirement for the test bed, but should still be considered as part of the design efforts.

R9 – Dynamic test scenario manipulation. This is considered an important functional requirement for the test bed. The ability to dynamically alter test scenarios would provide greater flexibility in the evaluation and execution of test scenarios. This is intended to ensure the effective time management when designing and executing test scenarios. This requirement would be extremely useful if using the test bed for operator training purposes. Since the design of the test bed is intended to be flexible to allow various application, instead of just limiting it to the purpose of security assessment.

Operator training is perhaps another aspect that should be considered to increase cyber security in relation to smart grid. Security awareness and training are critical consideration when maintaining traditional security goals. As previously discussed the smart grid is an amalgamation of traditional SCADA environments with ICT networks, this the need for security awareness and training to the management and implementation of security should be paramount. In addition to the dynamic alteration of test scenarios to achieve higher efficiency, and the use of test bed for operator training it is also important to understand the cost benefits associated with using such test bed. For these reasons, this is considered a essential requirement for holistic test beds.

R10 – Full audit trail implemented. This requirement is implemented as part of a full SCADA implementation, but for the purposes of a test bed implementation may be considered to be a separate and distinct requirement. The test bed should provide full audit trail. This is generally implemented via logging functionality. Audit trails are important from an evaluation and analysis point of view as they provide an indication of the state of the test bed during the evaluation. The logs may be investigated to understand events in detail. Additionally audit trails also provide a useful source of information for forensic analysis within the test bed.

Audit trails also provide useful information during forensic investigation, thus a test bed may be used for evaluation of current and emerging exploits for SCADA and smart grid in a control environment, i.e. the test bed, and forensic analysts can use audit trails and logs for analysis of the exploits. Having audit trails also provides transparency of the test bed functionality, i.e. a chain of transactions are logged to indicate exactly what events have been executed on the test bed.

The NIST framework [13] also highlights the importance of providing auditing and logging functionality for cyber security purposes. Specifically the document draws reference to audit log requirements for its Customer and Market domains identified in it reference model, for the purpose of implementing "integrity and non-repudiation".

R11 – Graphical User Interface. A graphical user interface is another non–functional requirement that may be useful for an implemented test bed. A GUI will enable greater ease of use of the test bed. This will encourage greater participation, as well as provide an effective means to interact with the test bed for the design, development, and execution of test scenarios. As with some other requirements this is related to the primary requirement of implementing a full SCADA system. However it needs to be addressed separately as the test bed itself should have a graphical user interface for ease of management and use. This requirement is particularly important to ensure that the test bed manager is able to perform test bed management functionalities effectively.

Although the IEC roadmap [11] does not directly advocate the use of graphical user interfaces it does highlight the potential for Common Information Model based graphics exchange, i.e. IEC 61970-453. This acknowledges the increasing need for visualisation of complex situations, e.g. critical infrastructure networks, and suggests that graphical elements may be exchanged between control centres. A GUI may be extended to address this recommendation. Additionally using a GUI for the HMI would provide an opportunity to use the test bed as a SCADA operator training tool, in addition to provide a more realistic test bed implementation.

All test bed designed reviewed for the purposes of this project as per the literature review, appeared to provide some form of graphical user interface.

R12 – Test bed readily available to greater community. This non–functional requirements grew out of project team comments and feedback. Current test beds are available either on a commercial basis or have access re-

stricted for security reasons. It is considered necessary to have a test bed that should be readily available to the greater community as an open source or free ware software. This is intended to allow for rapid development and maintenance cycles to enable the design and development of a richer feature set. It is believed that having an open source test bed would allow greater acceptance and testing of the test bed itself, along with fostering greater innovation as well as providing community maintenance of the test bed itself.

Although none of the test bed implementations stated that they were freely available, when an attempt was made to obtain the RMIT test bed via e-mail correspondence, it was discovered that the test bed development work was undertaken in partnership with un–named third parties, that prevented the release of the test bed for evaluation. Other test bed designs appeared to use commercial or propriety hardware or software.

3.2 Gap Analysis

Following the specifications of the reference requirements in Section 3.1 and the comparison of the individual requirement against the test bed design and implementation identified during the literature review process, it because necessary to evaluate the individual test beds against the reference requirements. The evaluation is summarised as a gap analysis. The gap analysis identifies the requirements and test beds and identified how adequately the test bed implement a specific requirement.

The results of the evaluation is summarised in Table 3.1. The table highlights each of the current and proposed test bed designs and compared it the reference requirements. The purpose of this is to highlight any gaps in the test bed design and to provide a basis for a uniform comparison of the different test bed designs. A major limitation of the test bed implementation reviewed was that none of them emulated a holistic electrical smart grid network, i.e. they either emulated a traditional SCADA network or a part of a distributed smart grid environment, and where there attempted to provide a complete solution, they used software simulations instead of actual emulation of the hardware, as was the case with the Illinois test bed. Other solutions such as the RMIT test bed where they did enable hardware integration, no actual SCADA hardware was used, instead Lego Mindstorm hardware was used, thus the test bed was more akin to a control system rather than a SCADA system.

Only the SAP and Illinois test bed enabled dynamic changes to the emulated environment. The SAP implementation provided web interfaces for the

implemented agents, and the Illinois test bed used the RINSE network emulator which could be altered. However none of the test bed examined provided a fully alterable solution.

Another requirement lacking in all test beds evaluated was the lack of actual appliance modelling. Given the recent increase in smart device production that monitor and report consumption, there should be some appliance emulation to enable evaluation and analysis of such devices within a smart grid. Finally the most limiting factor was that the test bed were not freely available for use by the greater community. The development and use of such test beds have been limited to commercial arrangements and the use use of commercial components used prevent use by others.

As summarised in Table 3.1, the RMIT implementation appears to be the most comprehensive test bed implementation. Although the test bed did not use actual smart grid hardware, it did provide and use interface to hardware. Unfortunately, due to contractual agreement with the project sponsors, etc, the test bed is not be made available outside the RMIT project. Again from the table it appears that the Virginia Tech test bed met the least number of requirements, however it was the only test bed implementation that provided integration to actual electrical grid hardware, and successfully tested it.

Requirement	Requirement type	Virginia Tech.	SAP	Berkley Mellon Venderbilt	Illinois	RMIT
R1 – Appliance	Functional	No	Yes	No	No	No
R2 – Consumption	Functional	No	Partial	No	No	No
R3 – Hardware	Functional	No	No	Yes	No	Yes
R4 – Real–time	Non-functional	Yes	Yes	Yes	Yes	Yes
R5 – IP	Functional	Partial	No	Unknown	Yes	Yes
R6 – SCADA	Functional	No	No	Yes	Partial	Yes
R7 – SCADA Protocols	Functional	No	No	No	No	No
R8 – NIST	Non-functional	No	No	No	No	No
R9 – Dynamic	Functional	No	Yes	No	Partial	No
R10 – Audit	Functional	No	Yes	Yes	No	Unknown
R11 – GUI	Functional	Partial	Yes	Unknown	Yes	Yes
R12 – Open source	Non-functional	Unknown	Unknown	Unknown	No	Unknown

Table 3.1: Summary of gap analysis

3.3 Proposed Design

The Initial design and implementation shall model and support a limited set of activities of a real-world SCADA system. In future implementations it is expected that the test bed shall wholly support the real-world SCADA environment as well as integration for full support of smart grid applications. Specifically the first phase of the project is intended to satisfy the following objectives;

1. Provide capabilities to demonstrate the end-to-end processing of SCADA systems.
2. Provide facility for basic testing, evaluation and integration of the limited features implemented.
3. Provide evaluation of the design architecture and related interfaces.

The proposed test bed design is will be a layered one. Only a small subset of the reference requirements are implemented in the test bed. The test bed planned for design and implementation shall provide the following functionality;

1. Test bed should support a smart grid architecture over a traditional SCADA implementation, to accurately model possible future deployment.
2. Design and developed using open source libraries to ensure design and implementation flexibility and to encourage acceptance and usage.
3. Network emulation shall be used to provide the functionality of data communications, instead of a simple simulation of the network.
4. Design should be extensible to enable support for multiple current and future SCADA protocols.

The following abstract conceptual verticals have been proposed to achieve SCADA objectives for data acquisition and control, data communications and data management and presentation within a test bed perspective;

1. Master Telemetry: This consists of the actual SCADA MTU software, the HMI applications, the events historical or RDBMS of some sort. Of course if multiple telemetry communications protocols are used, then some sort of protocol converter will be required to unify the protocols for the MTU services. Provides data management and presentation functions and provides a control interface.
2. Communications: The communications consists of the communications network, which is comprised of the communications infrastructure, i.e.

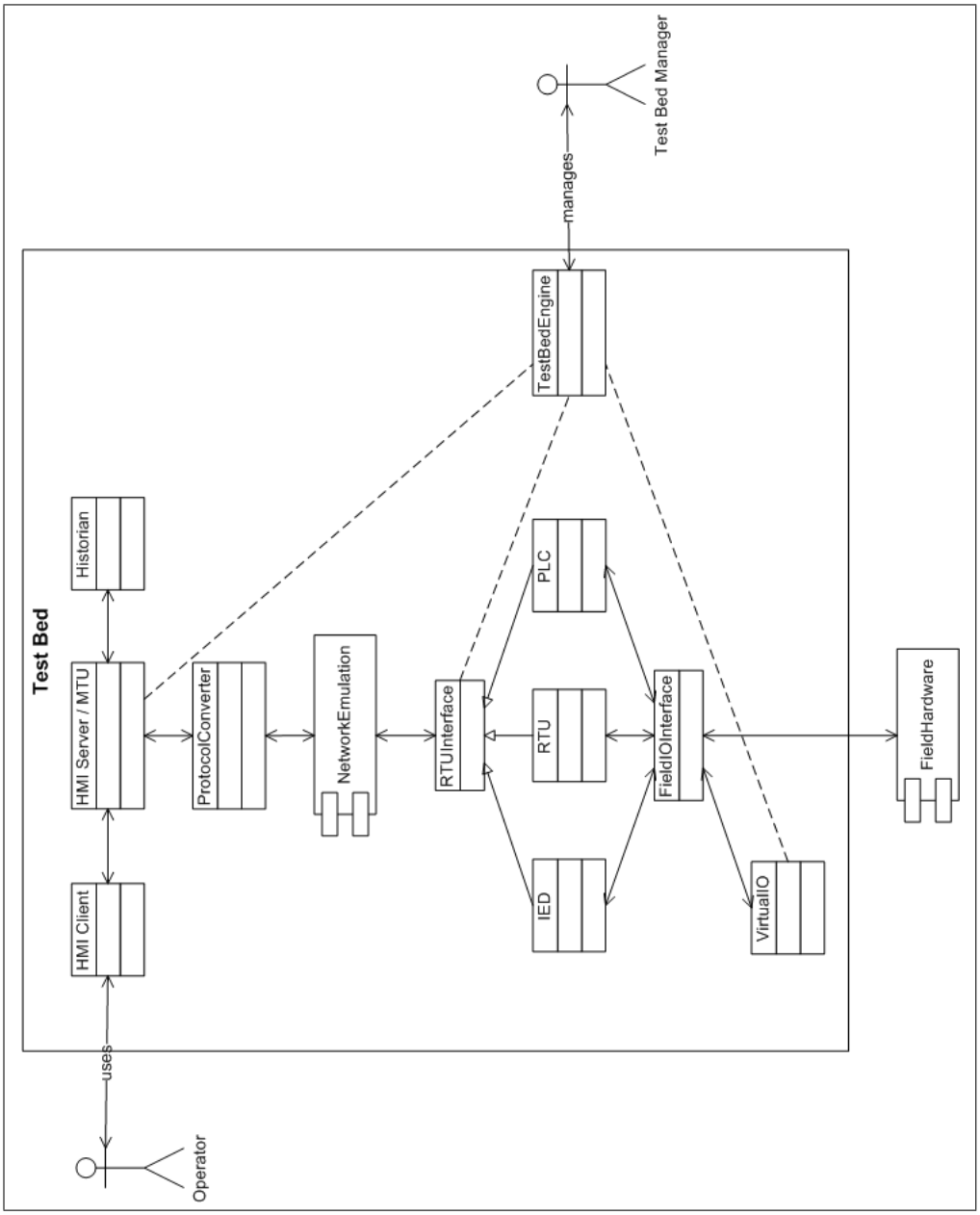

Figure 3.1: High-level design of proposed test bed architecture.

physical cabling, microwave/radio frequency wireless links, leased lines, etc., the telemetry communications protocols such as DNP3, Modbus, IEC-61850, as well as the communications transport protocols such as UDP over IP. Provides data communications functions.

3. Remote Telemetry: This consists of remote telemetry units, such as RTU's, IED's, PLC's etc. These devices are responsible for data gathering and actuator control functions. They read data from field devices and forward the data to the MTU, as well as sending actuator control commands to the field devices as received from the MTU. The readings are obtained as digital or analogue readings and similarly the actuator commands are either issues as digital or analogue controls. Provides data acquisition and actuates control functions

4. Field Bus: This consists of actual field bus hardware, i.e. the physical hardware used to connect the physical field devices to the RTU. These may be serial communication lines, etc.

5. Hardware: This refers to the actual electrical grid hardware such as generators, transformers, reclosers, circuit breakers, etc., that are connected using the field bus to the RTU devices. The hardware produces status, readings, metered values, etc. The values from the field hardware are to be read by RTU's, IED's, PLC's and other such hardware and communicated to the RTU. Provides low level data acquisition functions by providing access to the field bus.

In addition to providing abstraction, the secondary motivation for the verticals approach proposed here is to enable the implementation and manipulation of each component independently within the test bed environment to maximize evaluation and analysis, i.e. for a given test, the test environment variables and parameters may be changed independently to evaluate the system performance. For example if during an operator training test, an RTU reading may be changed independently by a test bed Manager to evaluate operator reaction or other such system functionality on-the-fly. Such parameterised controls should add flexibility in the design, deployment and management of the test bed, and allow for a magnitude of test bed scenarios for evaluation and analysis.

Since it is the purpose of the test bed is to provide a realistic and cost effective platform for SCADA and smart grid analysis and evaluation, it is important to ensure that the features of a SCADA and smart grid systems are emulated wholly. In an effort to provide a scalable communication infrastructure for the test bed the network shall be emulated. Emulation is a cost-effective means to deploy the topology as the network protocol stacks are virtualised.

Figure 3.2: Abstraction of communication in the proposed test bed architecture.

Further unlike conventional virtualisation, network emulation only virtualises the network protocol stacks without the need to virtualise the guest network operating system (NOS), the data storage mechanisms, etc. [14] The network topology implementation is provided using simulated links between virtualised network protocol stacks. Implementing the topology in this manner also allows for rapid deployment.

3.4 Summary

The introduction to SCADA and smart grids provides a good foundation for the requirements elicitation exercise. The NIST and IEC roadmap documentation were used to provide a basis for the requirements specification. Arising out of the literature review process for related work, a number of test bed design and implementations were identified. The specification of these requirements formed an integral part of the project. The requirement provided a baseline for the comparison of the test bed designs previously identified.

The comparison of the test bed against the reference requirements set, is provided as a gap analysis. The latter, was also a primary project output. Since the requirements were used as a baseline for the gap analysis comparison, it enabled a uniform means of comparing the individual test bed designs. The gap analysis was summarised to highlight the limitation of the reviewed test bed designs.

Following the gap analysis, it was evident that none of the current test bed designs adequately addressed a majority of the requirements, and thus for the purposes of the project were considered to be incomplete. Thus a high level design was proposed, based on the reference requirements identified to address the gaps and limitations of the current test bed designs. The high level test bed design was another major project output. The proposed high level test bed is to be used a foundation for the design and implementation of the smart grid test bed.

The following chapter will be used to present the details of the smart grid test bed implementation. The implementation will details individual aspects of the project, for example the design and development of the test bed source code, the use of network emulators, etc.

Chapter 4

Smart Grid Test Bed Implementation

The smart grid test bed implementation is based on the high level test bed design proposed in Section 3.3. The implementation of the smart grid test bed has been identified as one of the major project output in Section 1.3. The implementation provides a natural progression from the proposed high level design. The implementation attempts to address the gaps identified during the gap analysis, as well as the reference requirements proposed in Section 3.1.

4.1 Implementation

The smart grid test bed implementation is based on the proposed architecture. The primary focus during implementation has been to employ open source and freely available software and hardware tools. The rationale supporting this is to encourage free and open usage of the smart grid test bed to ensure community support and development for the project. The latter has also been identified as a high level requirement during the requirements specification. Due to the scope of the project it has not been feasible to perform a comprehensive analysis of the technology to employ, nor implement an complete test bed to support all the high level requirements identified. The project instead is intended as a starting point for a test bed to be extended further in subsequent work to implement a more comprehensive solution.

Thus development has focused in designing and implementing a proof of concept for the test bed, specifically the emulation of the TCP/IP stack using open source tools, and using it to communicate SCADA protocols. Again due to the diverse range of production SCADA protocols in use, only the Modbus protocol

was selected for use due to it's simplicity, maturity and ease of implementation. The smart grid test bed implementation thus attempted to produce a Modbus centric MTU and RTU that communicated over an emulated network to produce actual SCADA network traffic which may be captured for analysis using common networking utilities and tools such as tcpdump, etc. The captured traffic can be used for security analysis and evaluation.

Based on the proposed design as illustrated in Figure 3.1 and the conceptual verticals identified in Figure 3.2, a limited feature set test bed design was implemented. The implemented design is detailed in Figure 4.1. The design incorporates part of the high level design as well as the conceptual verticals identified. The illustration only provides a high level design of an end to end SCADA system.

The same abstracted communication architecture is used as a basis for discussing the implementation, i.e. the functionality is details in the following section to explain the implementation of the QUT smart grid test bed. The project was initially implemented using ANSI C, however after the initial implementation design deficiencies were identified. To over come these design deficiencies, the test bed was then ported to C++. The test bed implemented in phase 1 was not extensible, in that, to support additional SCADA protocol, large parts of the test bed would have had to be rewritten. When the project was ported to C++, proper object oriented design was employed to ensure that the test bed was sufficiently extensive to support additional SCADA protocols without the need for extensive changes.

The actual QUT Smart Grid Test Bed implementation is detailed in Figure 4.2. The implementation implements a software library to enable the implementation of Modbus based MTU and RTU. The figure is provided as an indicative architecture and not as an exercise in Unified Modelling Language (UML).

4.1.1 Master Telemetry

In the master telemetry verical, the MTU is implemented. The development of the MTU relies on the use of application protocol libraries, specifically on Modbus. During the actual design of the test bed several freely available libraries were examined and attempt was made to use such libraries. However due to a lack of author support and documentation relating to these libraries, these attempts were abandoned. The Modbus reference documentation used to implement the Modbus protocol libraries for use with the project. This was implemented as a ProtocolModbus class. The class provides the functionality to instantiate Modbus messages for use as the application layer protocol.

During the design and testing of the ProtocolModbus class, it was identified

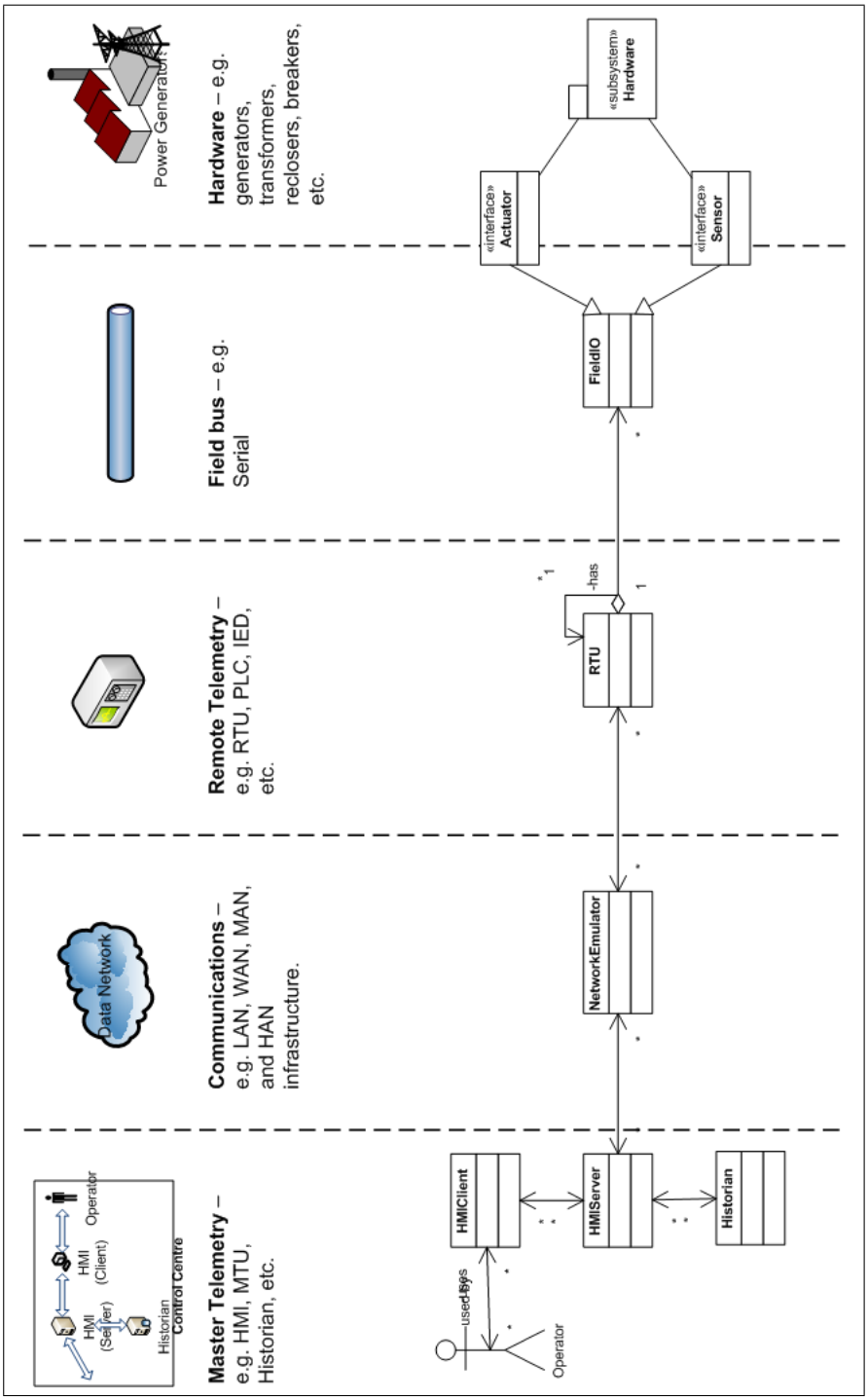

Figure 4.1: Proposed QUT smart grid emulator (QUSE) architecture using the abstracted communication architecture.

that the endian orientation of the platform needs to be considered when exchanging messages. This is because Modbus messages are exchanged in big-endian format. Thus utility code was developed to assist developers easily determine and manipulate byte level messages into the correct endian orientation. The ProtocolModbus class is used both in the development of the RTU and MTU. Details of the ProtocolModbus is discussed further in Section 4.1.3

The other major component in the Master Telemetry abstract vertical is a TCP/IP client. The client is required to instantiate a connection from the MTU to one or more RTU, on TCP port 502. Once a connection is established the Modbus request can be instantiated and sent to the RTU and an appropriate response received. Since the operating platform for the test bed is intended to be a Linux platform in keeping with the project objective of being freely available, the NetworkSocket class is implemented as a C++ wrapper for the underlying Linux libraries. The class can be readily extended or altered to accommodate Microsoft Windows platform if required.

The class is intended to provide a uniform mechanism for MTU and RTU implementations. To handle any exception conditions the NetworkSocketException class is provided to instantiate exceptions for handling. A sample agent and monitor applications were implemented to demonstrated the usage of the network library. The same sample application was then extended to provide actual MTU and RTU implementations.

4.1.2 Communications

To perform the network emulation a number of solutions were investigated. A number of these were commercial solutions and this not suited to the project requirement of being freely available. Although Network Simulator, i.e. NS can be run in emulator mode to perform network emulation this was considered to be unsuitable due to its ease of use. CORE was identified as a suitable network emulator. Common Open Research Emulator (CORE) provides a tools for the emulation of network topologies.

In addition to emulating a network CORE also permits integration into an actual physical network topology as well. CORE may be deployed on FreeBSD or Linux systems and is available in source under the new BSD License. CORE is based on the open source Integrated Multi-protocol Emulator Simulator (IMUNES project, developed by the University of Zagreb, Croatia. CORE allows for the virtualisation of light-weight network protocols stacks and further provides a GUI to link the virtualised instances to form a network topology. CORE may be used to implement the data communications infrastructure for the test bed.

Using CORE allows the rapid deployment of appropriate network topolo-

gies without requiring the need for code re-deployment, i.e. the Master and Remote Telemetry code remains unaffected by changes within CORE. This allows greater flexibility in the use of the test bed for evaluation purposes. For the purpose of rapid prototyping and proof of concept other network emulation and virtualisation technology were not evaluated in any great detail. Although there are a number of virtualisation mechanisms available, system level virtualisation such as OpenVZ, which is employed by CORE is considered to be easy to manage and flexible [15].

Detailed instruction on the installation and configuration of CORE for the purposes of the project are detailed in Section A.1.1. CORE uses OpenVZ containers were used for deploying the RTU and MTU emulation software on the hosts. The actual network interfaces of the emulated hosts are available outside the CORE environment as sub interfaces. Generic network tools and utilities can be used to interrogate these interfaces, just as a conventional network interface may be used. Network traffic can be routed into and out off the CORE environment by using a bridged interface which can be instantiated by adding an emulated network hub onto the emulated CORE topology. This procedure is detailed in Section A.1.3.

4.1.3 Remote Telemetry

The code design consists of an RTU class, that has a composite relationship to one or more FieldIO devices. Since real world implementation of RTU may be chained, an emulated RTU may have reference to other associated RTU objects as well. The FieldIO class is discussed further in Section 4.1.4.

The RTU location is modelled using an Address class, that provides a consistent interface for the RTU address. Presently only IPv4 addresses are modelled using the Address class, but this may be extended to provide additional addressing schemes such as IPv6, unit ID, etc. to provide RTU addressing. To implement SCADA protocols, and abstract Protocol class is implemented. Actual SCADA protocols can be implemented by extending this class. For example the ProtocolModbus class, which implements the Modbus protocol extends this Protocol class. Abstract methods in the Protocol class that must be implemented are toBytes, toBytesLen, and init. The actual method prototypes are defined in the code documentation.

The ProtocolModbus code was developed using the Modbus and Open Modbus standard, and the Modbus RTU was developed to provide Class 0 Modbus compliant functionality, i.e. implementing Modbus function code 0x03, i.e. 3 and 0x10, i.e. 16. These functionality allow for Modbus requests to read and write register values. To implement additional Modbus function code, the im-

plemented Modbus RTU and the Modbus protocol classes will need to be enhanced. The actual design and implementation details are described in Section 4.1. To enable the testing and evaluation an emulated Modbus meter code was developed that allowed a user to perform registry read requests.

As previously discussed in Section 4.1.1, the RTU used the same TCP/IP wrapper libraries as the MTU. This enables the RTU to be developed as a TCP/IP server, that listens on port 502 for Modbus request messages. The RTU developed for testing purposes was an emulated smart meter appliance, that reported the voltage, current, frequency of the electrical consumption, and provided an ON/OFF state. Such appliances can be developed rapidly using the ProtocolModbus class and the NetworkSocket classes.

4.1.4 Field Bus

The field bus vertical is used to provide an interface between the remote telemetry vertical and the actual hardware. In the test bed the field bus is instantiated using the FieldIO class. A FieldIO object may be an Actuator or Sensor. Actuator and Sensor interfaces are provided to enable to design and implementation of emulated hardware actuators and sensors readily, as well as provide an interface for further enhancements including the use of actual hardware actuators and sensors onto the test bed. Due to time scale and scope limitations of the project the FieldIO class was not developed further, and are only provided as abstract classes. These can be extended in subsequent development work to provide greater functionality.

4.1.5 Hardware

In the current version of the test bed there is not actual hardware emulation provided. This was due to time scale and cost limitations of the project. In future works, actual Modbus and other smart grid applicance may be integrated into the test bed. During the development of the test bed a few Modbus compliant meters were investigated, but were found to be too costly. Specifically the Satec Global PM180 meters (http://satec-global.com/UserFiles/satec/files/_PM180_Flyer.pdf) and the SEL 734 revenue meter (http://www.selinc.com/sel-734/).

4.2 Summary

A number if unforeseen issues were encountered during the implementation. The attempts to use third party libraries for the implementation of specific SCADA protocols proved to be quite problematic. In addition to a lack of support on freely available libraries, there was also a lack of documentation to

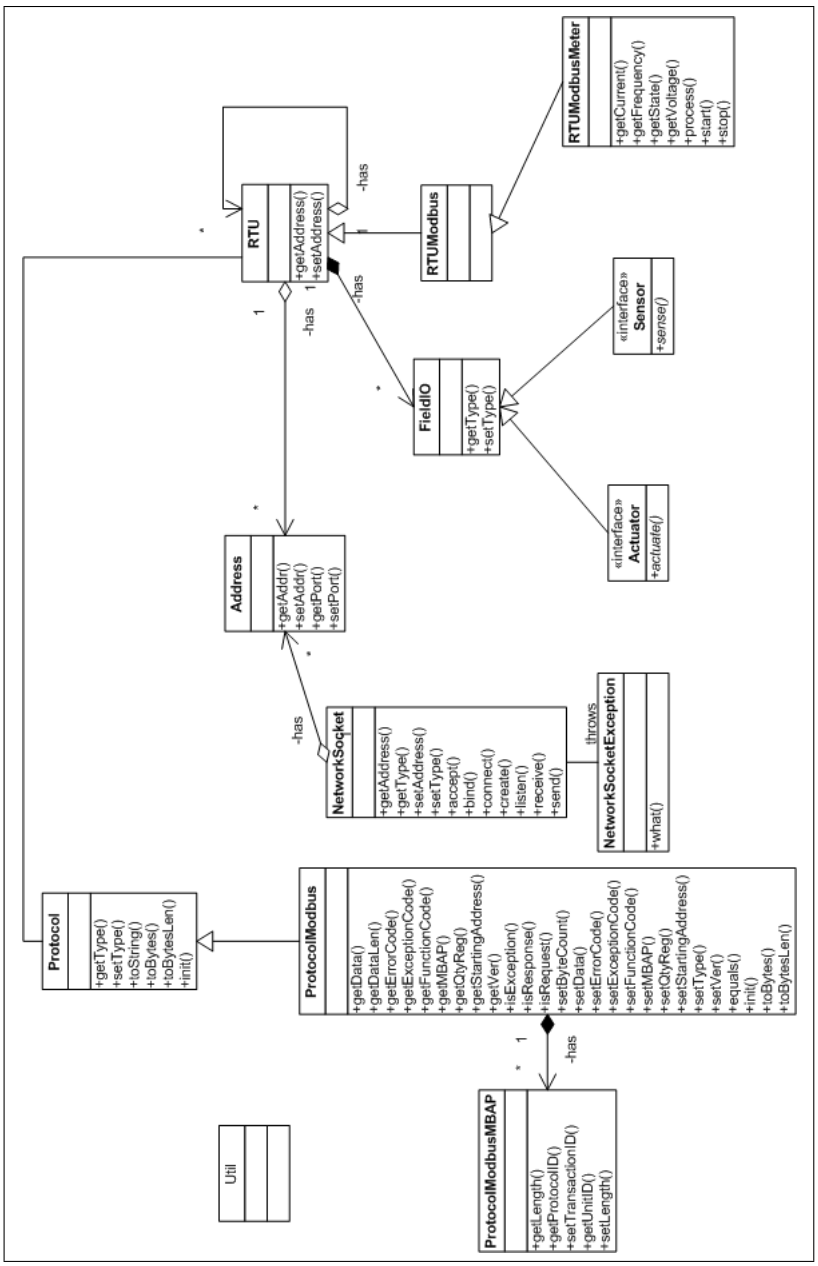

Figure 4.2: Implemented QUT smart grid emulator (QUSE) classes.

assist users of the libraries. For these reasons the C++ classes to implement the Modbus protocol were developed. The advantage of using these, is that, they can be easily extended to include any additional requirements, and it eliminated the dependency on external libraries. Another problem encountered related to the fact that the host platform endian needs to be considered during low level development. Specifically the Modbus protocol exchanges messages in large or big endian format. Therefore the code needs to be able to handle both little and big endian platforms. For this reason utility classes were developed to assist developers deal with such issues.

Although the implementation does not provide a complete implementation of all the reference requirements, it is intended as an initiation for the test bed emulator. Thus with future development efforts, the feature set can be extended to include additional requirements, as well as improve the performance and quality of the current code and features. The test bed design and implementation need to be evaluated to identify performance issues and to determine the suitability of the test bed for security evaluation and actual smart grid emulation.

In the following chapter the evaluation of the test bed is detailed. The chapter will provide an overview of the evaluation methodology, i.e. the evaluation criteria that will be used, both qualitative and quantitative. It will also provide an explanation of the experimental evaluation that will be conducted and present the results of the evaluation. The chapter will conclude with a discussion of the results.

Chapter 5

Evaluation

The implemented test bed as detailed in Section 4.1 is evaluated to ascertain the extend to which the implementation addresses the reference requirements. To perform the evaluation a set of qualitative and quantitative evaluation criteria are identified. The criteria is used to define the measure to use to evaluate the test bed. These measures are detailed in Sections 5.1 and Section 5.2. The actual quantitative evaluation is implemented as experiments to evaluate and test specific criteria. The actual experiments are defined in Section 5.2.2. The results of the experiments are discussed in Section 5.2.3.

5.1 Qualitative Evaluation

The primary qualitative evaluation criteria used is the examination of what subset of the reference requirement set were implemented by the test bed and further to discuss how well each of the requirements were implemented. The qualitative features defined by the reference requirements are discussed in detail in Section 3.1. It should be noted that during this evaluation there is no empirical or statistical data available to substantiate the evaluation results.

5.1.1 Results

From the set of requirements identified a limited subset was used for the test bed implementation. This was due to the limited time frame for the project. Since the project was undertaken in two phases, the initial premise was to commence development on the test bed with a limited feature set, and in subsequent phases add additional features as and when required.

Although the test bed does not implement a comprehensive realisation of the requirement, the requirements implemented by the test bed are highlighted

below. An indication of the extent to which the requirement is addressed is also provided. The following requirements are implemented by the test bed.

1. R5 – The test bed provides communication using TCP/IP, i.e. it support IP based communication between the MTU and the RTU over an emulated IP network implemented using CORE.

2. R7 – The test bed provides emulated smart grid intelligence built on top of traditional SCADA topology and protocols. The test bed emulated MTU and RTU that use the Modbus protocol.

3. R11 – The test bed provides an audit trail by logging protocol messages send and received to a log. Since the test bed is implemented in a Linux platform it uses the syslog logging feature.

4. R12 – The test bed is readily available under the GPL license and used open source and freely available software components, that can be easily deployed on a number of hardware platforms.

The following requirements are partially implemented by the test bed

1. R1 – The design provide emulated sensors and actuators that may be enhanced to provide actual modelled consumption. Currently they are providing emulated reading by producing random values with valid ranges

2. R3 – The design of the test bed provides for the integration of hardware components into the RTU software. The interfaces developed currently only provide for emulated hardware, but can be used for actual hardware, but software code will be required.

3. R4 – The design used Linux operating system for both, the network emulation as well as the host operating system for the MTU and RTU software. Ideally a real time operating system should be used, but the code is designed using ANSI C++ and thus should be easily ported to other appropriate platforms

4. R8 – The design is based on the NIST concept model, but does not provide implementation of all components of the model. Additional components can be developed and integrated as and when required.

In respect to the gap analysis that was perform, the current test bed design also does not completely address all reference requirements specified, however the design and implementation of the test bed is flexible and extensible to allow future enhancements to be readily implemented and integrated to address current as well as future requirements.

5.2 Quantitative Evaluation

The quantitative evaluation is intended to provide empirical data as a means of evaluating the test bed design and implementation. The experiment set up described in Section 5 are intended to provide a repeatable means to evaluate the test bed. The experiments proposed are executed a number of times to get normalised results. The individual experiment results are documented and interpreted to provide insight into the behaviour and functionality of the test bed.

The quantitative evaluation of the test bed deals specifically with the performance of the network emulation. The throughput and latency of the network emulation is evaluated using software tools such as ICMP ping request and response messaging and iperf. As discussed previously the network emulation is provided using CORE, which uses OpenVZ to emulate the TCP/IP stack capabilities of the host. The actual tools that may be used for the evaluation are identified and discussed herein, however details of the implementation and use of the tools are provided in the appendix of this report.

The following sections will present the evaluation criteria for performing the quantitative evaluation, it will also discuss the experiment set up required to conduct the evaluation and summarise with the results of the evaluation. The interpretation of the results and discussion is conducted in Section 5.3

5.2.1 Evaluation Criteria

Throughput. Iperf utility (http://iperf.sourceforge.net/) is used to perform throughput testing on the emulated network. The iperf tool measures the maximum TCP and UDP bandwidth performance between the host OS and the emulated hosts. The actual details of obtaining and compiling iperf on the host network is provided in the Appendix, Section A.4.1 . Version 2.0.5 of iperf was used for this evaluation.

Latency. Ping is a common network tool for troubleshooting network latency issues. The ping utility is widely available on all modern network operating systems. Ping implements the ICMP echo request and response functionality, and provides a round trip time (RTT). Using the ping utility is a reasonable approach to check the latency of the emulated network as it does not reply on any packet processing by the host being sent the ping request.

Attacks. Since one of the objectives of the project was to provide a means to performing security assessment of smart grid networks, the evaluation by means of implemented attacks is an important one. This is intended to

evaluate the capacity of the emulated network under attack. Due to the limited feature set implemented, complex attack scenarios and exhaustive security assessment cannot be undertaken. Under normal operation of the smart grid emulator, Modbus protocol messages can be sent to the emulated devices to obtain register values. Thus a possible attack scenario may be a denial of service attack on the RTU device.

For this evaluation multiple Modbus protocol messages to read register values can be sent to the RTU device and the behaviour of the RTU observed and documented. It is expected that the functionality of the system will be affected, i.e. since the RTU devices are implemented as single thread, the RTU would block, and thus cause delayed responses to the legitimate MTU.

Another potential attack scenario may be to perform incomplete TCP handshake and thus attempt a denial of service attack on the CORE node. This is intended to prevent legitimate MTU requests from being processed by the RTU. The latter may be best described as a TCP SYN flooding denial of service attack. Since the RTU itself is not directly attacked, but only the TCP stack on the CORE emulated node, it is expected that the RTU would be able to successful respond to legitimate MTU requests, but the response may be delayed slightly.

5.2.2 Experiment Set-up

Since an extensive analysis of all possible aspects of a SCADA test bed is not possible, and it considered beyond the scope of this project only a limited set of experiments have been conducted. The experiments set up are discussed in detail and attempt to address the evaluation criteria identified in Section 5.

The set-up for all experiments was limited to running a single hosted node on the CORE emulate. Since CORE is limited only by the processing power, i.e. hardware platform on which CORE is deployed, it use of additional nodes was not considered as it would introduce additional variables into the experiment. Thus for the experiment set up, only a single node was deployed, with an IP address of 10.0.0.10. The bash script created as part of the project was used to generate the configuration file for the experiment. This is detailed in Section A.1.2 in the appendix. The emulated test bed deployed using CORE is illustrated in Figure 5.1.

The virtual interface on the host system was assigned an IP address, 10.0.0.250, to bridge the connection between CORE and the host system. Once core was started basic ping tests were conducted to confirm connectivity. To simplify this task a bash script was created, as detailed in Section A.1.3. A CORE node

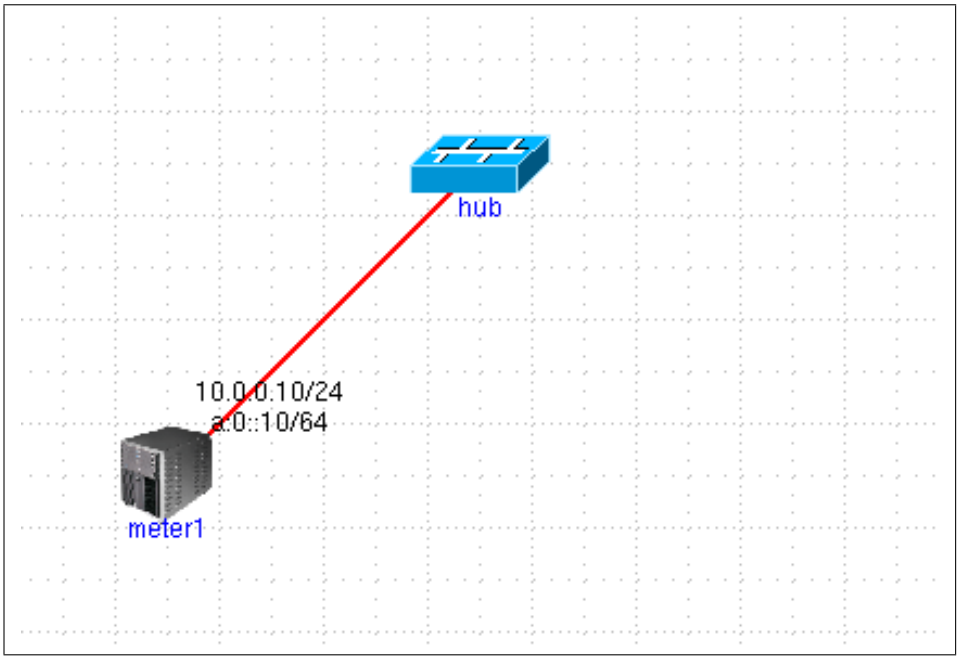

Figure 5.1: CORE emulated network for test bed evaluation experiments.

console was obtained to provide access to the CORE node, and to instantiate experiments. For the experimentation conducted as part of this project only IPv4 was used. Although the tests can be easily altered to be run on IPv6 network stack for the purpose of brevity, the latter was not done.

Latency. The latency experiment is conducted on a single host instance running on the CORE network emulator. As with experiment the previous experiment, the host is bridged with the hosted node on CORE. A baseline is obtained to provide a comparison against the actual test results. For the purposes of establishing a baseline thirty (30) ping requests were sent to the loop back address. The ping utility sent ICMP echo request messages and received the ICMP echo response messages. The round trip times for each test was manually recorded. Specifically the minimum RTT, the maximum RTT, the mean RTT and the standard deviation was noted. Any packet loss was also recorded. The test was repeated ten (10) times to obtain consistent latency for the loop back address. The results are discussed in details in Section 5.2.3.

To get the latency results, thirty (30) ping requests are sent to the hosted CORE node from the host system. Again as with the baseline, this experiment is repeated ten (10) in an effort to obtain a sample of consistent

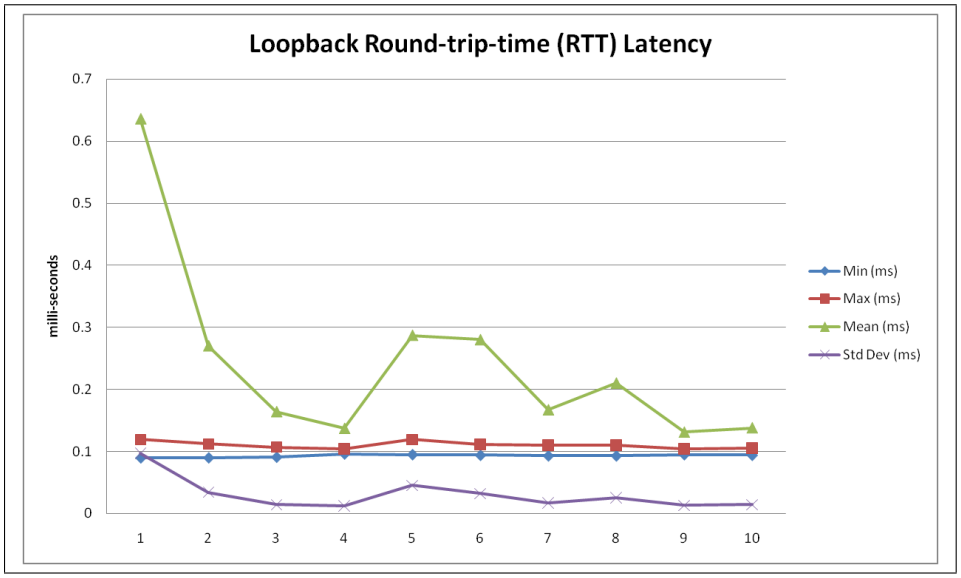

Figure 5.2: Baseline established by performing ping tests on loop back address

results. The results are manually recorded and discussed in Section 5.2.3.

Throughput. The throughput experiment is carried out on a single host instance running on the CORE network emulator. The iperf server is instantiated on the emulated host, and a bridge created between the host network operating system and the hosted CORE host instance.

Iperf client is executed on the host system to connect to the CORE instance. Just as with the baseline experiments, the experiment is repeated thirty (30) times in an effort to obtain a normally distributed result. The results are manually documented and discussed in Section 5.2.3. For details on setting up, configuring and running the iperf utility, please refer to the details command and instructions provided in Section A.4.1.

As a baseline for the Iperf testing, iperf throughput tests were conducted on the local loop back interface, i.e. lo, with IP address 127.0.0.1. The iperf server was run to bind to this IP address, and then subsequently iperf was executed in client mode for a duration of thirty (30) seconds to connect to the server using twenty (20) parallel threads. Both the iperf client and server operated in TCP mode on port 502. The actual commands used for running the test are documented in Section A.4.2. This test was repeated a number of times, thirty (30) to be exact, in an attempt to get a normalised distribution of results. The throughput results were manually recorded and tabulated. The tabulated results are

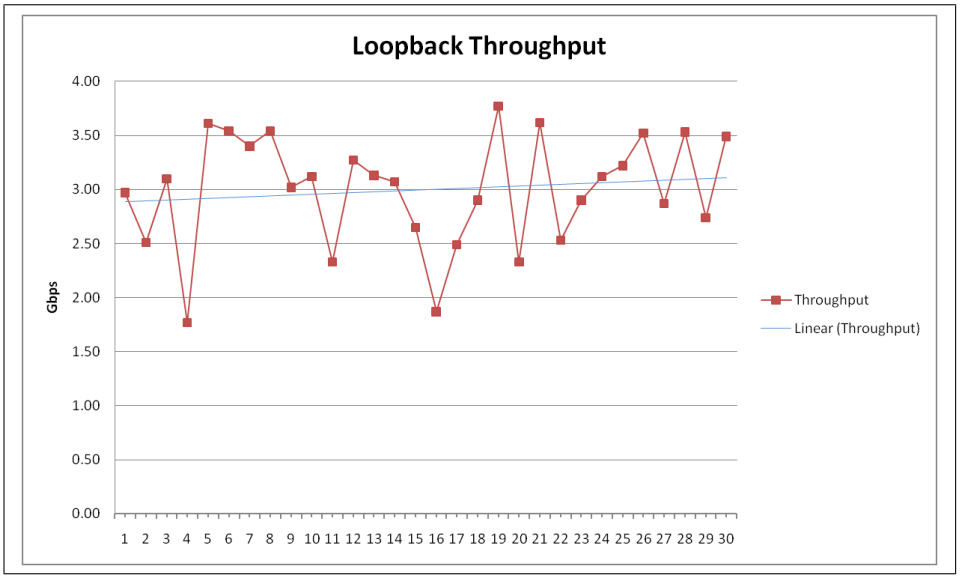

Figure 5.3: Baseline established by performing throughput tests on loop back interface

presented in Section 5.2.3.

Once a baseline was established iperf throughput tests were performed between the host interface and the hosted node on CORE. The results are detailed in Section 5.2.3. The actual command parameters supplied were identical to the command parameters used in the baseline testing, apart from the IP address of the host of course. It should be noted that in some cases there was a lack or corroboration between the statistic reported by the iperf client and the iperf server. The results were off by as much as 0.44 Gbits/sec between the statistics reported by the iperf client and those reported by the iperf server. The exact cause of this discrepancy was not investigated, as it was considered to be beyond the scope of the project.

Denial of Service. The set up for the denial of service experiment is different from the remainder of the experiments. In this set up, two hosted CORE nodes are used. This is illustrated in Figure 5.4. Once is used to communicate with the MTU, and the other is used to launch the attacks. As discussed in the quantitative evaluation criteria for attacks, there are a number of possible attack scenarios that may be evaluated.

For the purpose of evaluating denial of service attacks, the TCP SYN flooding technique is used. This is intended to exploit the single execution path of the emulated RTU to prevent legitimate MTU request from

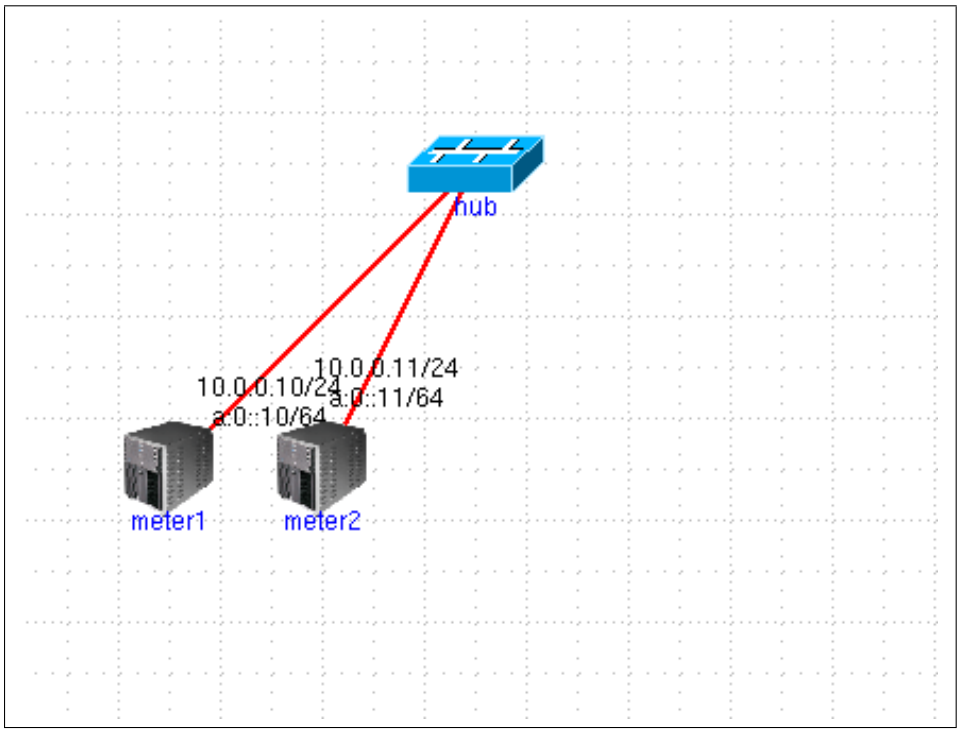

Figure 5.4: CORE emulated network for denial of service evaluation experiments.

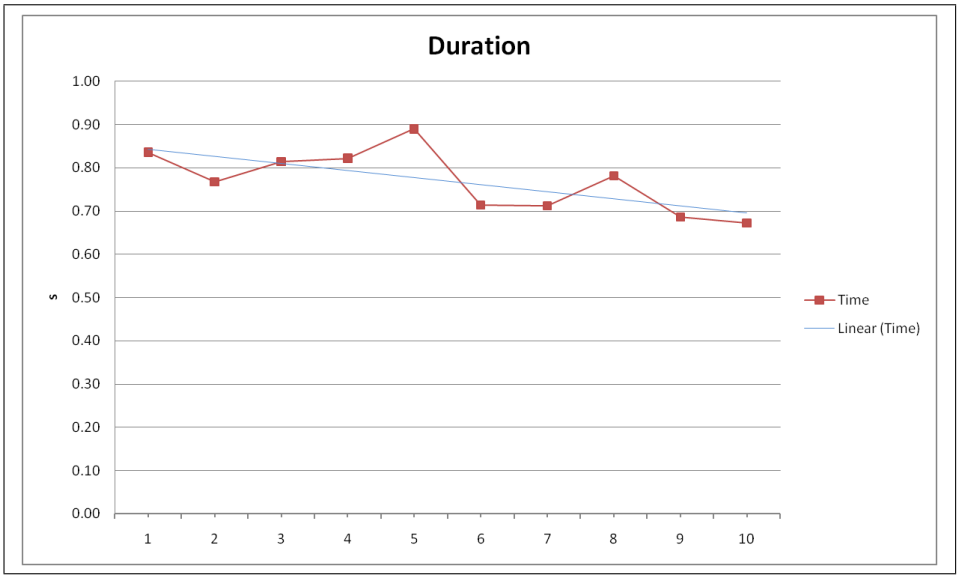

Figure 5.5: Time taken to complete MTU requests before attack scenario

reaching the RTU. The hping utility was used for implementing the SYN flooding on the RTU. The details of installing and running the hping utility for this experiment is details in the Appendix, in Section A.5.1.

Initially, the actual time stamps are noted by using the touch command on empty files. This command will update the time stamp on the touched file to the current system time, thus a time is recorded before executing the test and another after the tests have completed. However since this relied on file system operations it was considered to be inaccurate, thus the time stamp calculation was performed in the actual BASH script used to execute the tests. Essentially, the time stamp is obtained using the date command, and the difference between the start and end time stamps considered to be the duration of the test. The limitation of this experiment is the date command, since the duration is presented in seconds. The experiment should be enhanced to provide a more accurate time stamp mechanism, i.e. accurate of milli–seconds would be preferred. Thus the standard Linux "time" command was used. The results were recorded manually and graphed as illustrated in Figure 5.5.

A baseline for the attack is performed by executing thirty (30) legitimate MTU to RTU Modbus requests. The duration of the script was calculated using the time command, and the real CPU time values used. This is repeated ten (10) times to get an average for the baseline results.

Once an appropriate baseline performance is obtained, the same experiment is repeated by this time with the TCP/IP stand on the CORE hosted node under load from the hping command. The duration of the test is calculated in the same way.

5.2.3 Results

In Section 5.2, the evaluation criteria was specified for the implemented test bed design. The quantitative evaluation criteria was used to specify concrete experiments, as detailed in Section 5.2.2. These experiments were set up and executed, and the results presented here.

Latency. The baseline latency testing performed using the ping utility consisted of ten (10) test runs each consisting of thirty (30) ICMP ping messages. The total average or mean RTT was 0.242 milli-seconds. During the baseline testing there were no request time out message nor dropped packet messages observed, as expected for a loop back address. The results of the baseline test have been graphed in Figure 5.2.

Just as with the baseline the same ping results were executed onto the CORE hosted node using the identical set of parameters, apart from the destination IP address. Essentially the ping utility was used to send thirty (30) ping echo request messages to the host, the test was repeated ten (10) times. The obtained minimum, maximum, average and standard deviation data was recorded, and plotted on a graph, refer Figure 5.6 to illustrate the latency performance.

There does not appear to be any direct correlation between the two graphs for the loop back and CORE latency tests. The latency on the CORE hosted node appears to be higher than running the latency test on the loop back interface. This may be attributed to the additional network hop between the host OS and the CORE hub to get to the CORE hosted node.

The average of the mean latency for the baseline test is approximately 0.2ms, where as, the latency to the CORE hosted node is approximately 0.7ms. The results of test run number 9 appears to break away from the trend as it spikes away from the norm. The denial of service attack experiments are conducted next.

Throughput. Following the establishment of the baseline, the same experiment was repeated on the hosted environment. The same anomaly, as described in the experiment set up was observed, i.e. iperf client and

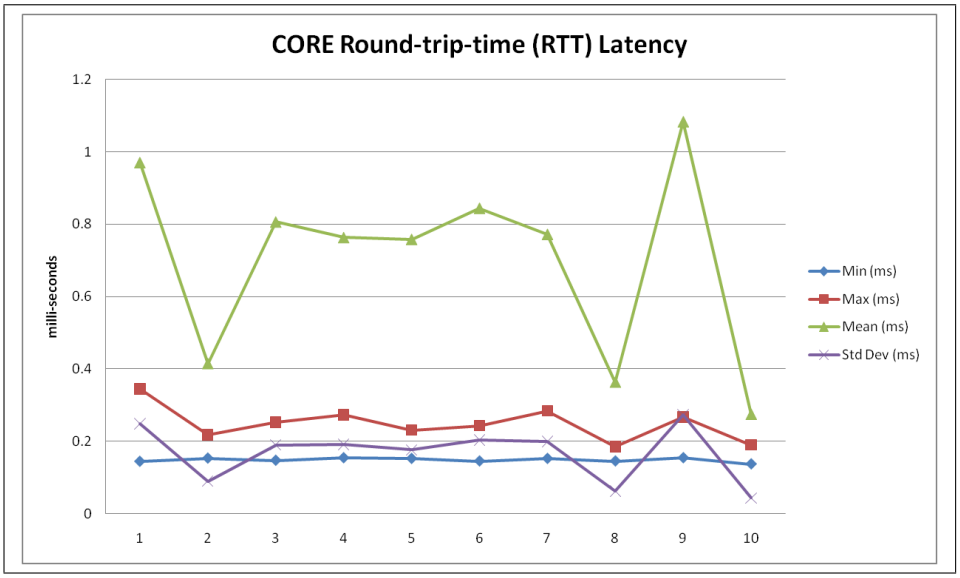

Figure 5.6: Ping tests on CORE

server reports do not match, when running the tests on the CORE emulated link and node. However the difference between the reported rates were higher in the emulated environment. For the purpose of reporting the results, the values reported by the iperf server were used, since they were generally lower than the values reported from the iperf client.

Another key observation was the extremely high CPU usage during the iperf throughput tests, generally the combined CPU utilisation of the iperf client and iperf server was in access of ninety-five per cent (95%). The actual results of the throughput testing in the loop back interface is illustrated in Figure 5.3. The graph illustrated the individual tests performed (x-axis) and the actual throughput achieved (y-axis) in Gbps, i.e. giga-bits per second. A trend graph is plotted in the figure as a linear graph.

The Figure 5.7 illustrated the results of the observed iperf throughput testing. As previously mentioned the iperf server statistics are used for the analysis. The graph provides the throughput results in mega–bits per second (y–axis) against the number of test runs (x–axis). The linear trend graph is also plotted for illustrative purposes. The emulated network interface has significantly lower throughput performance than the testing on the local loop back interface. It should be noted that the iperf throughput testing should only be executed to evaluate isolated networks as iperf is considered to be an intrusive test, and may disrupt production environments. Following the throughput test the latency tests were conducted.

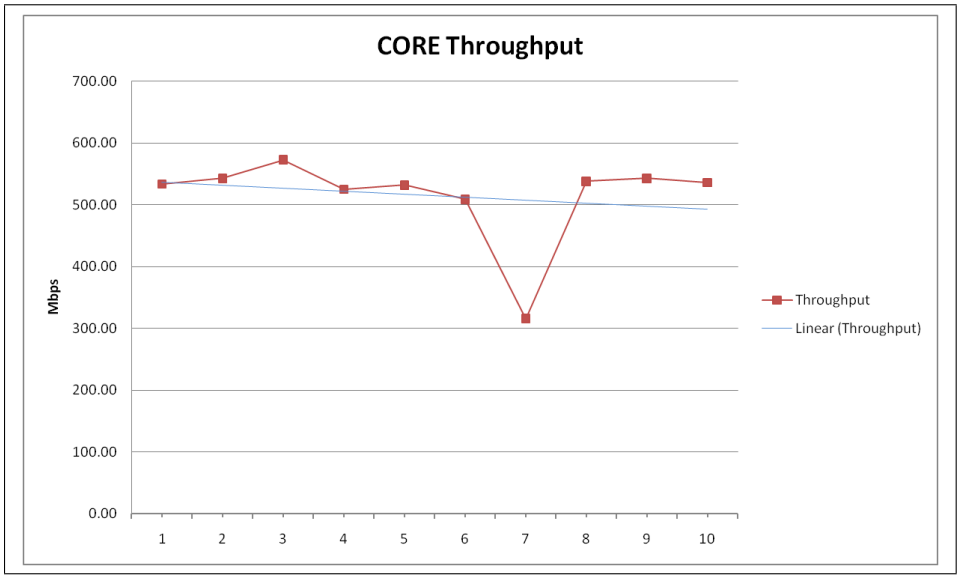

Figure 5.7: Throughput tests on CORE

Denial of Service. The denial of service attack is performed a SYN flood on the RTU. The attack is implemented using the hping utility. The installation and execution of the hping command is provided in detail in Section A.5.1.

The time delays recorded were not significant, i.e. 1 second, thus the script was used to include the nanosecond output, again the timing accuracy was not precise enough to yield any empirical data to indicate processing delays nor a denial of service. Thus the CORE architecture was altered to include five (5) CORE hosted nodes, one RTU running the Modbus agent, and the remaining for running hping to the RTU node. The results yielded were almost identical to all previous results. The results are illustrated in Figure 5.8, which is not dis–similar to Figure 5.5.

Given the lack of accurate time stamp calculation, the results produced for the experiment cannot be considered to be conclusive. The overall results are discussed in detail in Section 5.3.

5.3 Discussion

The discussion chapter provides a natural evolution to the evaluation process. The discussion is intended to interpret and summarise the results obtained as part of the experimental testing. The discussion also highlights recommended

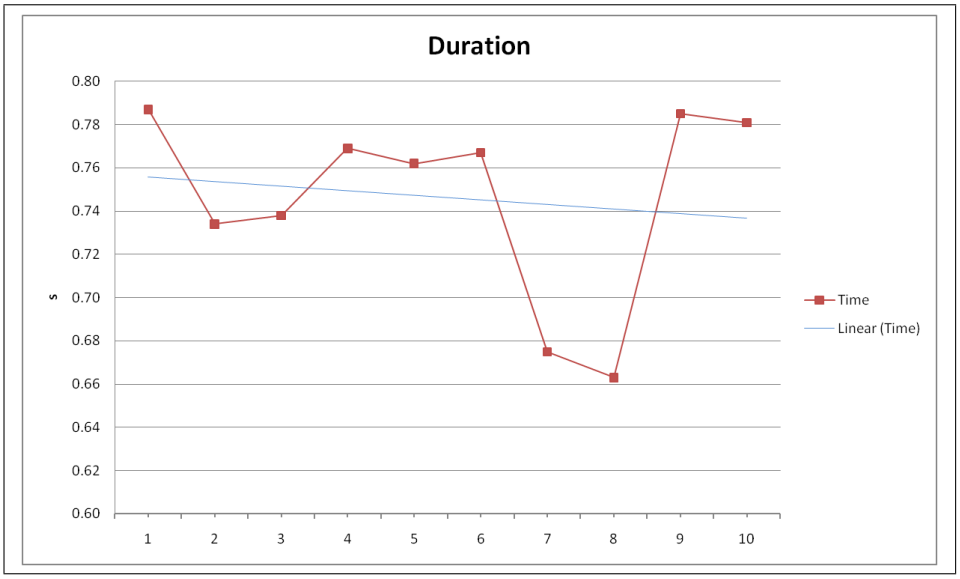

Figure 5.8: Time taken to complete MTU requests during denial of service attack

future work.

The result of the qualitative evaluation provides a basis for comparison of the QUSE test bed implementation against the test bed designs reviewed in the gap analysis. As discussed in Section 5.1, the implemented test bed design does not address all the high level requirements. However it appears to be better placed than the reviewed test bed designs as highlighted in the gap analysis. Since the design and implementation was undertaken using the specified requirements, the design is flexible and extensive enough to enable the realisation of the remaining requirements in subsequent work. The QUSE test bed presents an opportunity for further development and enhancements to be undertaken to implement a full feature set test bed to address all specified high level requirements.

The quantitative evaluation results provide an over view of the technical aspects of the CORE network emulator used to implement the Communication for the test bed. The performance parameters that were evaluated were latency, throughput and response to denial of service attacks. The ping results used to determine latency was useful but not conclusive. A baseline was established by running pings to the local loop back address, the RTT latency was 0.242 ms. The ping responses from the hosted CORE node, averaged to 0.705 ms. The results indicate that the CORE network does have a higher latency. The difference in the average latency results is not significant, thus a greater sample set is required to obtain a normally distributed set of data for analysis. However,

the performance provided by CORE is still adequate for the requirements of the test bed.

The second aspect of the network performance that was evaluated as the network throughput. The throughput attained on the loop back interface was approximately 2.99 Gbps. The same test on the CORE network yielded only an average of 514Mbps. This immediately limits the application of the test bed to throughput rates lower than this. For example the test bed cannot be used to evaluation or testing of scenarios or smart grid topologies that specify a requirement for greater throughput, i.e. it cannot be used for 1000 Mbps networks, etc. It should be noted that the sample set for the loop back testing was three (3) times larger than the sample sets for the remainder of the testing. The trends identified by graphing the results indicate an upward linear trend for the loop back testing, but a downward trend for the CORE hosted node tests. Since iperf is an intrusive test, the CPU utilisation during iperf testing spiked to 100%. Hence, since CORE implements virtual TCP/IP stack for the hosted node, and thus relies on the host resources, the iperf test may be adversely affecting the performance of CORE.

The final quantitative evaluation that could be completed to successfully meet the time constraints of the project was an implementation of a denial of service attack on the TCP/IP stack of the CORE hosted node. The data collected for this test used the Linux "time" command. The hping utility was used to generate a TCP SYN flood. The results of this experiment presented some very anecdotal results. Specifically the time taken by the RTU to successfully respond to thirty Modbus requests was averaged to 0.7696 seconds, or 769.6 ms. The same number of requests during the flood ping were completed on average in 0.7461 s or 746.1 ms. This suggests that the RTU performed better under load. However the variance between each test run was lower for the normal operations, i.e. during load the variance was much greater between individual test runs. Again this may be a result of using a sample size that may not be adequate for the analysis.

Additionally there is a need for further experiments to be designed and implemented to gain a better understanding of the performance of the test bed, for example the denial of service attack focused only on the TCP/IP stack of the CORE hosted node. There is a need to develop further experiments to evaluate higher level layers, i.e. the application later protocol. Modbus specific messages can be generated from PCAP files and executing these using utilities such as tcpreplay. PCAP files may be obtained from online resources such as http://www.pcapr.net/home or commercial sources such as http://www.digitalbond.com/ Further evaluation may also be conducted using security evaluation and assessment tools such as honey net.

Given the limited data sample size produced as a result of the limited test run, it it difficult to draw and conclusive interpretations. However the data does provide an understanding of the behaviour and performance of the test bed, specifically the network emulation, thus indicative conclusion may be safely drawn as discussed above. There is a need for further testing to generate a larger data sample set, as well as to develop additional test scenarios and experiments to provide a comprehensive set of test results relating to the performance of the test bed.

The current work also presents appropriate opportunities for further development and enhancement, which may be undertaken as part of subsequent or future work. These are presented in the following section and may present future direction for the project.

5.3.1 Future work

Requirements. The requirements specified as a project output are only a set of high level reference requirements. There needs to be further investigation and analysis work to elicit design level requirements. The high level requirements only provide an overview of the required functionality of potential test bed implementations, but a design level set of reference requirements will also provide design and implementation guidance in addition to the required feature set. Potential non functional requirements that need to be determined would relate to the performance of the test bed to emulate real work scenarios, such as latency to provide real-time or near real-time performance. These should be specified empirically.

Attack Scenarios. Due to time scale limitations of the project, detailed evaluation of attack scenarios could not be conducted. Whilst it may not be feasible to set up and evaluate a comprehensive set of requirements, there is a definite requirement for further testing. As mentioned previously, additional tools, such as HoneyD, etc. would be useful for further testing and work had commenced on installing and configuring these, but could not be included in this report due to time constraints.

Network Topology. Since CORE is used to provide network virtualisation by using the emulated TCP/IP stack, further experiments need to be conducted to evaluate various network topologies. CORE should be investigated further to use various network topologies and to provide network specific performance attributes, e.g. experimenting with CORE to manage bandwidth, latency and throughput parameters, etc. Also further work is required using CORE to determine the performance of different network

topologies for network emulation

Enhancements. Since the test bed design and implementation only provided a very basic implementation of smart grid functionality, there is great potential for further implementation and enhancement to the libraries developed as well as the existing smart devices. For example, the current Modbus functionality implemented is limited to reading and writing register values, this can be enhanced to include additional Modbus functionality. Additionally additional SCADA protocols such as DNP3, etc. can be implemented to make the test bed support a heterogeneous suite of protocols. Most of the appliance code only supports single thread processing, this is another potential area for enhancement, i.e. the code can be extended to support multiple threads to be better able to support simultaneous request and response processing

HMI. The current test bed does not provide a HMI. Since the implementation of a holistic SCADA system has been specified as a core high level requirement, the test bed should be extended to include a full featured HMI. The process visualisation library (http://pvbrowser.de/pvbrowser/index.php), was briefly examined to be used to implement a suitable HMI, again due to time scale tolerances of the project this would could not be undertaken.

Hardware. Hardware integration is another requirement that should be investigated further. This is important not only to demonstrate the functionality of the test bed, but also to enable the test bed to be integrated in future to hardware based test bed implementation to enable more detailed assessments to be undertaken. The current design provides interfaces, which may be used to implement hardware integration.

5.4 Summary

The evaluation of the implemented test bed was another main output of the project. The evaluation is considered to be an important aspect of the research and development process as it provides an indication of how adequately the requirements were addressed. To undertake the evaluation using a systematic approach qualitative and quantitative evaluation criteria were specified and detailed. The qualitative criteria related specifically to the examination of the requirements specification and the implemented design to understand how adequately the specific requirements were addressed. The quantitative evaluation focused on the actual technical performance of the implemented design. Due to time limitations comprehensive evaluation could not be undertaken.

The quantitative evaluation criteria was defined and addressed using specified experiments. The experiments descriptions were provided as a guide to setting up the experiment. The experiments were then realised as test runs to obtain actual results. To ensure consistent requests, the test runs were repeated a number of times to obtain an average performance. The results of the experiments are provided and discussed.

The next chapter of the report provides a conclusion to the project. The conclusion provides an overview of the project and presents a summary of the work undertaken.

Chapter 6

Conclusion

In an effort to gain more from traditional critical infrastructure, such as electrical power networks, modern information and communication technology is being integrated with it to provide benefits not normally attainable under current circumstances. The smart grid is an example of such integration. SCADA systems allow the control and management of remote assets. The smart grid integrates traditional SCADA systems with ICT technology to enable better control and monitoring. Smart grids are also intended to provide realisation of benefits such as:

- Greater security and resilience to cyber attacks
- Greater resilience and recovery from electrical power failures
- Greater efficiency and operational optimization of electrical power network assets
- Greater efficiency and operational optimization of electrical power generation, transmission, distribution, and use.

The integration of tradition SCADA environment, which were not initially designed for security, with externally connected communication networks, such as the Internet, introduces a number of security concerns. Due to the critical nature of such infrastructure the traditional security goal of availability is a primary concern. Security assessment needs to be undertaken to analyse and mitigate vulnerabilities, however it cannot be performed in production environment due to their critical nature, and setting up test environments may not be economical. Thus there is a need to emulation of production environment, such as test bed. The benefits to be realised from the implementation and use of test bed are listed below.

- Provides an isolated environment for assessment and analysis, thus addressing risk associated with testing on production environments.
- Reduces system complexities, by providing a controlled environment for assessment.
- Provides cost benefits by reducing costs for assessment work, and for costs associated with operator training.
- Provides platform for assessing novel ideas and concepts which may not be possible in production environments.

The IEC and NIST framework reference documents were used to gain a detailed understanding of the smart grid landscape. The project identified and reviewed current test bed designs and implementation. The test bed designs were identified as part of the literature review process. The designs included work by academic as well as commercial research institutes. A number of different approaches were adapter in the designs, and most designs implemented different feature sets. This is discussed in detail in the Related Work section. Based on the NISt and IEC roadmap documentation reviewed earlier, a set of high level requirements were specified. These requirements were used as a basis for comparing the reviewed test bed and performing a gap analysis.

The gap analysis identified the requirements that were addresses by these designs, but more importantly highlighted that none of the designed adequately addresses all of these high level requirements. Based on the gap analysis and the high level test bed design requirements, a high level test bed design was proposed. The design proposed a layered approach to the development. Specifically the SCADA domain was divided in conceptual verticals based on the abstracted function of the components. Thus the Master Telemetry, Communication, Remote Telemetry, Field bus and Hardware abstraction was provided.

This high level design was used to implement a subset of reference requirements using the abstraction layers specified. Initially the development was undertaken in C, but then migrated to C++. The Modbus protocol was selected for implementation due to its simplicity and open standard. C++ libraries were created to implement a basic set of this protocol. CORE was used to provide network emulation. CORE achieved this by virtualising the TCP/IP stack on the host system, and providing OpenVZ containers for nodes. These containers were used for the deployment of MTU and RTU implementations for evaluation. The implemented design was evaluated and the results discussed.

The evaluation was undertaken, using a systematic approach, i.e. evaluation criteria were specified for both qualitative and quantitative evaluation, experiments were designed to address the quantitative evaluation criteria and results

collated to provide empirical data for analysis. The test runs were limited due to time constraints, this resulted in reduced sample data sets for analysis. Analysis based on results enabled inferences to be drawn, however further testing and analysis is recommended to gain a detailed basis for inferences. Also there needs to be more work done to demonstrate the usefulness of the test bed for exploit assessment, i.e. although some work was done during the project, due to time constraints these could not be completed. Tools to use and deploy on the smart grid test bed for analysis included using snort, metaspoit, honeyd, and tcpreply.

This initial development provides a good platform for future work. The discussion of the results provides direction for future work. The future work is recommended to address some of the limitation of the current test bed implementation. The research work undertaken has been useful in identifying areas of research need within the smart grid test bed landscape, and has reiterated the need for test bed for security assessment and analysis within critical infrastructures. In conclusion, QUSE provides a good reference point for further development and innovation. There is a particular need for further enhancement and implementation of additional features. The project produced the following outputs.

- Reference set of smart grid test bed requirements
- Gap analysis of current test bed designs against reference requirements
- Proposed high level design of smart grid test beds
- Design and implementation of QUT Smart grid Emulator test bed
- Evaluation of the implemented design

Appendix A

Instructions

The instructions provide a step–by–step guide to what commands where executed. This includes instructions for the installation, configuration and execution of components of the test bed implementation as well as evaluation scripts to evaluate the test bed design. The individual modules are highlighted below with a brief description of their function and performance.

A.1 CORE

CORE stands for Common Open Research Emulator, and is a utility to emulate entire networks by virtualising the stack with OpenVZ. CORE is built on top of Integrated Multiprotocol Emulator/Simulator (IMUNES). CORE works in FreeBSD and Linux kernels. For the purpose of the test bed CORE was used to provide communication network emulation.

A.1.1 Instalation

The set-up instructions were executed as the root user on a Redhat Enterprise Linux 5.5 installation. Set up includes the installation of OpenVZ and CORE as well as required dependencies. The instructions are detailed in the CORE documentation. At the time of this project only CORE version 3.5 was available, however towards later stages CORE version 4.0 was released. The new version of CORE was not evaluated nor assessed for any purpose.

1. sudo bash
2. yum update
3. cd /etc/yum.repos.d

4. wget http://download.openvz.org/openvz.repo
5. wget http://download.openvz.org/RPM-GPG-Key-OpenVZ
6. rpm –import RPM-GPG-Key-OpenVZ
7. yum install ovzkernel vzctl
8. vi /etc/grub.conf
9. wget http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.
10. rpm -Uvh epel-release-5-3.noarch.rpm
11. yum -y install bridge-utils
12. yum -y install iproute
13. yum -y install ebtables
14. yum -y install automake
15. yum -y install gcc
16. rpm -ivh core-3.5.1.i386.rpm
17. rpm -ivh core-root.3.5.1.i386.rpm

There were no major issues encountered when deploying CORE on RHEL and CEntOS platforms. The graphical user interface provided by CORE makes it easy to use. It's simple configuration files also enable the rapid deployment of networks by generating such configuration files using custom scripts.

A.1.2 Configuration

The network topologies can be created manually using the CORE GUI interface. This provides great flexibility in the layout of the network, as well as providing an opportunity to experiment with different topologies and set up. However, it would not be feasible to use the GUI to generate configuration file for, say a hundred (100), or even a thousand (1000) nodes.

Thus a simple PERL script was written to automatically generate a flat network structure with the required number of nodes. The script is located in the core directory in the project root directory.

1. /usr/bin/perl -w ./makeCoreConfig.pl X

The configuration generated by the script add a CORE hud device. The hub device may be used to bridge the network interface from the host system into CORE. This is detailed in Section A.1.3. Bridging is essential to enable communication between the host system and the emulated CORE systems.

A.1.3 Bridging

External connectivity is achieved by using a bridging interface between the host system and the CORE network. The bridging interface is automatically created by CORE once a hub is added into CORE and CORE is started. To achieve connectivity into and out of CORE an IP address needs to be assigned to the bridging interface on the host system. Once this is done IP connectivity is achieved between the host environment and the CORE environment.

To simplify the process of assigning an IP address to the bridging interface, a simple bash script was created to assign the IP address of 10.0.0.250 to the bridging interface. The script is located in the core directory in the project root directory.

1. ./makeBridge.bash

For IP connectivity out of the CORE network, the nodes must have a possible route out of the CORE network. In a simple case this may be achieved by adding a default route to bridged interface.

A.2 Honeyd

Honeyd is a light weight daemon that enables the virtualisation of hosts on a network. Based on the configuration the virtualised hosts can run arbitrary network services for the purposes of simulating a network. Honeyd is also able to virtualise network topology and allow the simulation of actual network routes and routers, as well as allowing the management and configuration of network attributes such as latency, network packet loss, etc. Honeyd simulates the operating systems TCP/IP stack, and may be useful for setting up nodes for vulnerability assessment within the test bed

A.2.1 Instalation

1. wget http://www.honeyd.org/uploads/honeyd-1.5c.tar.gz
2. tar zxvf honeyd-1.5c.tar.gz
3. sudo yum -y install libevent libevent-devel

4. sudo yum -y install libdnet libdnet-devel
5. sudo yum -y install libpcap libpcap-devel
6. sudo yum -y install libpcre
7. sudo yum -y install libedit libedit-devel
8. ./configure
9. make
10. sudo make install

A.2.2 Running honeyd

To execute a honeyd network using a network configuration file the following command may be used

1. /usr/local/bin/honeyd -p nmap.prints -f ./honeyd-modbus.config 10.0.1.10-10.0.1.15

A.3 FARPD

The Fake ARP Daemon, or FARPD, provides the implementation of a fake ARP daemon for use with HoneyD. It responds to ARP requests for IP addresses modelled using HoneyD, to a specific interface MAC address. FARPD is required to set up HoneyD networks on the test bed. However it has been initially develop for the BSD platform, as a result installation on SysV systems needs a bit of work if compileing from a source tar ball. Detailed instruction for compiling are provided in Section A.3.1.

A.3.1 Installation

1. wget http://farpd.sourcearchive.com/downloads/0.2-10/farpd_0.2.orig.tar.gz
2. wget http://farpd.sourcearchive.com/downloads/0.2-10/farpd_0.2-10.diff.gz
3. tar zxvf farpd_0.2.orig.tar.gz
4. gunzip farpd_0.2-10.diff.gz
5. cd farpd-0.2
6. patch -p1 ¡ ../farpd_0.2-10.diff
7. mkdir -p /usr/lib/bin

8. ln -s /usr/bin/dnet-config /usr/lib/bin/dnet-config
9. ln -s /usr/bin/dnet-config /usr/lib/bin/dumbnet-config
10. ./configure –with-libdumbnet=/usr/lib –with-libevent=/usr
11. ln -s /usr/include/dnet.h /usr/include/dumbnet.h
12. make
13. sudo make install

A.4 Iperf

Iperf is a network utility used to generate TCP and UDP data streams for the purposes of testing network performance. It is a freely available utility deployable on a variety of modern operating platforms. Iperf can be used to determine network performance characteristics such as throughput and latency. The following commands were executed on the development and testing environment to download the iperf source code and install it. For the purposes of the project, iperf is utilised to test the bandwidth and throughput of the emulated CORE network links.

A.4.1 Installation

1. wget http://sourceforge.net/settings/mirror_choices?projectname=iperf&filename=iperf-2.0.5.tar.gz
2. tar zxvf iperf-2.0.5.tar.gz
3. ./configure
4. make
5. sudo make install

A.4.2 Running iperf

During the loop back iperf baseline the following command were used to execute the iperf server and client respectively. The first command is executed to instantiate the iperf server, and the second command to invoke the iperf client. The server invocation binds the iperf server to the address and port specified. In the example above the address 127.0.0.1 and port 502 are used. By default iperf uses TCP.

1. /usr/local/bin/iperf -s -B 127.0.0.1 -p 502

2. /usr/local/bin/iperf -c 127.0.0.1 -p 502 -P 20 -t 30

The client command invokes the client to connect to the server specified at the address following the –c switch, in this example 127.0.0.1 is used, and the port 502 is specified using the –p switch. To ensure full throttling of the interface to wholly utilise the bandwidth, multiple threads of the iperf client need to be invoked. The –P switch is used to specify the number of threads to use. For the experiments performed, twenty (20) threads are invoked, and finally the –t switch is used to thirty (30) seconds as the transmit time. The same client and server commands are repeated to separated tests, however the IP addresses used are altered to suit the network topology and host configuration.

A.5 hping

hping is a command line utility that can be used to implement denial of service attacks on TCP/IP stacks. hping is a packet assembler and analyser that support TCP, UDP, ICMP and raw IP protocol messages. hping is used in the test bed to implement a denial of service attack as a TCP SYN flood on a CORE hosted node.

A.5.1 Installation

1. sudo yum -y install hping

Bibliography

[1] *A Testbed for Secure and Robust SCADA Systems*, vol. Proceedings of 14th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS '08)., IEEE, April 2008.

[2] S. Jung, J. gu Song, and S. Kim, "Design on scada test-bed and security device," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 3, pp. 75–86, October 2008.

[3] U. DoE, *21 Steps to Improve Cyber Security of SCADA Network*. Washington, DC, USA: United States Department of Energy, September 2005.

[4] C. M. Davis, J. E. Tate, H. Okhravi, C. Grier, T. J. Overbye, and D. Nicol, "Scada cyber security testbed development," in *Power Symposium, 2006. NAPS 2006. 38th North American*, pp. 483 – 488, September 2006.

[5] C. Queiroz, A. Mahmood, J. Hu, Z. Tari, and X. Yu, "Building a scada security testbed," in *2009 Third International Conference on Network and System Security*, 2009.

[6] I. E. C. T. C. 57, *Communication networks and systems in substations. Part 10, Conformance testing.* International Electrotechnical Commission, 1st ed., 2005. "2005-05" "Reference number, IEC 61850-10:2005(E)"–Cover. "International Standard IEC 61850-10 has been prepared by IEC Technical Committee 57: Power systems management and associated information exchange"–Foreword.

[7] *A Secure IEC-61850 Toolkit for Utility Automation*, Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security, (Washington, DC, USA), IEEE Computer Society, 2009.

[8] C. Santoni, J.-M. Mercantini, M. F. Q. V. Turnell, A. Scaico, and J. do N. Netoé A., "A real-time interface simulator for operator's training: a proposed architecture," in *SCSC: Proceedings of the 2007 summer*

computer simulation conference, (San Diego, CA, USA), pp. 460–467, Society for Computer Simulation International, 2007.

[9] S. Karnouskos and T. N. de Holanda, "Simulation of a smart grid city with software agents," in *Simulation of a Smart Grid City with Software Agents*, 2009 Third UKSim European Symposium on Computer Modeling and Simulation, 2009.

[10] S. Collier, "Ten steps to a smarter grid," in *Ten steps to a smarter grid*, pp. B2 –B2–7, April 2009.

[11] *IEC Smart Grid Standardization Roadmap*, 2010.

[12] IEEE, *Multi-Agent Systems in a Distributed Smart Grid: Design and Implementation*, March 2009. Multi-agent systems in a distributed smart grid: Design and implementation Pipattanasomporn, M. Feroze, H. Rahman, S. Adv. Res. Inst., Virginia Tech, Arlington, VA; This paper appears in: Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES Publication Date: 15-18 March 2009 On page(s): 1-8 ISBN: 978-1-4244-3810-5 INSPEC Accession Number: 10614353 Digital Object Identifier: 10.1109/PSCE.2009.4840087 Current Version Published: 2009-04-24.

[13] N. I. of Standards and Technology, "Nist framework and roadmap for smart grid interoperability standards, release 1.0," tech. rep., January 2010.

[14] J. Ahrenholz, C. Danilov, T. Henderson, and J. Kim, "Core: A real-time network emulator," in *CORE: A real-time network emulator*, pp. 1 –7, nov. 2008.

[15] Y. Z. D. R. S. J. G. Bhanage, I. Seskar, "Experimental evaluation of openvz from a testbed deployment perspective," in *Proceedings of the 6th International Conference on Testbeds and Research Infrastsructures for the Development of Networks and Communities*, May 2010.