POWER GRID SIMULATION, EVALUATION, AND TEST FRAMEWORK

BY

DAVID CARL BERGMAN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Adviser:

Professor David M. Nicol

ABSTRACT

The Virtual Power System Testbed (VPST) at the University of Illinois at Urbana-Champaign is part of the Trustworthy Cyber Infrastructure for the Power Grid (TCIP) and is maintained by members of the Information Trust Institute (ITI). VPST is designed to be integrated with other testbeds across the country to explore performance and security of Supervisory Control and Data Acquisition (SCADA) protocols and equipment. First, we discuss the approach we took to developing virtual SCADA models, validating the models, and how researchers can use the local test bed to validate their own technologies. Then, we discuss potential use cases in order to motivate the integration of VPST with other testbeds, identify requirements of inter-connected testbeds, and describe our design for integration with VPST.

To my parents, for supporting me through this process as well as raising me to believe I was capable enough to tackle such a large project.

ACKNOWLEDGMENTS

I would like to thank David Nicol for all of the advice he has given me along the way. Tim Yardley and Dong Jin both collaborated heavily on the way that VPST would interact with remote test beds and were a pleasure to work with. Matt Davis answered quite a few of my questions about PowerWorld and the power grid in general. Susan Hinrichs contributed many constructive suggestions and I am thankful for them. I also wish to thank the multiple people that corroborated my models and provided ideas for enhancements. This work was supported in part by a grant from the National Science Foundation (CNS-0524695). Also, this material is based upon work supported by the Department of Energy under Award Number DE-OE0000097.

***

TABLE OF CONTENTS

DEFINITION OF TERMS

- *DNP3.0* - A protocol used to gather data and issue control commands between master and slave devices. Also goes by DNP v3.0, DNP3, or just DNP.
- *Power grid* - The collection of buses, generators, lines, transformers, etc. that comprises the system of power delivery from power stations to power consumers.
- *Substation* - Also known as an outstation, the substation is an unmanned location in charge of controlling and collecting data from intelligent electronic devices (IEDs) in the immediate area.
- *Data Aggregator* - A SCADA device at the substation level, which exists to compile data from the RTUs and report the data to the control station.
- *RINSE* - Real-time Immersive Network Simulation Environment.
- *MODBUS* - An older protocol that operates point-to-point, generally over a serial connection such as RS-232.
- *RTU* - Remote Terminal Units are devices in the substation that are responsible for responding to requests from the control station.
- *SCADA* - Supervisory Control and Data Acquisition: A generic term for industrial control systems.
- *Control Station* - Central locations used to coordinate control decisions for regions of the power grid.
- *TCP/IP* - The Transmission Control Protocol and the Internet Protocol. Commonly used protocol combination in the Internet to address packets to processes and route segments to hosts, respectively.

- *CRC* - Cyclic Redundancy Check. The CRC is used to detect bit errors that occur during communication.
- *Relay* - A digital device that measures circuit status and can open/close a physical breaker.
- *PowerWorld* - A steady-state power simulator.

CHAPTER 1

INTRODUCTION

The power grid is an important piece of the infrastructure that allows our nation to conduct business, communicate, and generally enjoy most modern comforts. Essentially, without the power being delivered in a reliable and safe manner, nothing would work. We are also currently going through a rather major upgrade, termed the "smart grid." While the existing architecture is by no means "dumb," it is a heterogeneous mixture of old machines, new machines, old protocols, new protocols, and vastly different networks across domains. As such, it presents a large problem to those who wish to introduce new innovations and standardize the grid. Part of this problem can be solved piece by piece, but in order to get a good picture of how technologies interact in the grid, test beds will play a crucial role. The Virtual Power System Test Bed aims to provide a national resource, whereby new technologies can be tested in a realistic environment, and clear feedback can be given about their efficacies.

As part of the NSF/DOE/DHS supported TCIP project [1], VPST combines large-scale simulation/emulation of networks of SCADA power devices with real power system hardware and software, and a commercial electric flow generation and distribution simulator. VPST is unique in its integration of virtual and physical equipment, as discussed in [2]. It is also unique in that it supports SCADA-specific virtual traffic. VPST's principal role so far has been to demonstrate the feasibility of integrating the components it has, demonstrate the feasibility of a number of cyber-attacks, and study the performance and effectiveness of certain other TCIP developed technologies. It is not yet to the level of general purpose utility achieved by, say, DETER, although we are working to bring it to that state.

Large portions of this chapter, Chapter 2, and Chapter 13 as well as the majority of Chapters 9 through 12, and the figures therein, are from a previous work [3] done by myself and others. The other authors have agreed to its inclusion here.

Work on VPST has thus been focused on two aspects. First, work has been done on the local simulation engines to support a SCADA-specific environment. Our local network communications simulator, RINSE, has been useful for many years in analyizing malicious traffic patterns on the Internet. The scalability that it offers presents a good fit for modeling the SCADA network – one of the few systems that rival the modern Internet in size and complexity. Building on the RINSE framework, we have designed virtual models of hosts and protocols which represent their physical analogues. The two models developed so far provide an important subset of the SCADA environment's functionality. Namely, we have implemented virtual models for relays and data aggregators. These two models serve as the basis for modeling substations within the control network and can serve as archetypes for implementing new SCADA models. We have also developed a module to interface with the electrical simulator which runs concurrently. This module periodically polls the electrical simulator to retrieve information pertinent to the operation of the virtual relays. The collection of these virtual mdoels allows an environment in which new technologies can be evaluated prior to deployment. Thus, this work allows one to scale a virtual SCADA network beyond what could be accomplished if one were restricted to using purely physical devices. Therefore, research groups requiring a scalable SCADA model to test their ideas now have a platform to do that. In order to ease the use of our framework, automating information extraction from the electrical simulator to provide an analagous reprensentation for both RINSE and the Electricity Management Software (EMS) has been a priority as well.

Concurrent with the work on the local test bed and beyond the original work on VPST [2], we have redesigned the VPST architecture to support integration with other testbeds, and have already demonstrated basic integrative functionality. We aim towards enabling one to leverage resources from external collaborators and accomplish tasks that may not have been feasible without them. For example, one lab may have expertise in cyber warfare and another may emphasize actual SCADA devices. VPST provides detailed models of SCADA-specific protocols, enabling studies where an attack is mounted on a large-scale SCADA network. Such integration has a unique set of requirements, and we have extended VPST with these in mind.

The beginning of this thesis focuses on describing the SCADA simulation portion of VPST. First, we will discuss the motivation for our system in

Chapter 2. Then, we give background on the network communications simulator, RINSE, that provides the basis for our SCADA models in Chapter 3 as well as the unique characteristics of the power grid. Next, in Chapter 4, we describe DNP3, one of the primary protocols used in SCADA, as well as attacks against DNP3 and the approach used to model it. From there, Chapter 5 discusses virtual hosts and the control flows for each of the SCADA-specific virtual hosts. Chapter 6 details the local lab enivronment and what each portion is capable of providing. Chapter 7 describes the steps we have taken to develop and validate our efforts to create the virtual testbed. Next, Chapter 8 describes a potential workflow for testing a new technology in the local test bed.

After describing the local test bed, the thesis will discuss VPST in terms of how it interacts with remote test beds. This portion will view VPST and its components through the lens of three potential use case scenarios, providing some concreteness behind VPST design decisions. Chapter 9 begins the discussion of the extension to VPST, starting with three use cases. Next, we discuss testbed requirements in Chapter 10. Chapter 11 gives a brief overview of VPST and then delves into the details of our extension. Chapter 12 discusses related projects and how they may use VPST as a national resource. Finally, we conclude in Chapter 13 while detailing some of our ongoing work.

CHAPTER 2

MOTIVATION

There is an emerging awareness of the need for security in SCADA systems, and the Department of Energy (DOE) is giving considerable attention to securing the power grid. The power grid is a critical infrastructure (CI) due to our country's reliance on electricity. Simply put, there would be chaos without a reliable energy delivery system. We have seen what happens when power is cut to large areas [4, 5, 6], and it would be disasterous for it to happen on an even larger scale. Blackouts have occured for a variety of reasons, be they mechanical failures, control failures, network failure, operator failure, or a combination. There are also realistic attack vectors – both cyber [7, 8, 9] and physical [10] – that, while have yet to be exploited, must be better understood. Investigation of all of these failure modes will result in remediation techniques that can provide protection. Both independent researchers and government officials have formed workgroups and task forces [11, 12] to investigate and offer their advice [13], but these suggestions need to be tested and refined before they can be implemented.

However, working directly with the power grid to conduct these investigations is a poor decision for multiple reasons. For the very same reason that the power grid must be made secure – that it is critical to the operation of our country – modifications to the grid must not be done until adequate testing has taken place. As such, an important approach to studying cyber-security of the power grid is through test systems, so as not to interfere with real systems. These test systems may use actual equipment in order to properly understand how the equipment will react in various situations. However, the scope of the power grid makes it infeasible to create a physical test system anywhere near its full scale. Simulation and emulation help to alleviate this concern, as it is possible to create large-scale network simulation/emulations of models as large as regional or even national power grids.

Specifically, certain technologies would stand to gain a lot from detailed simulation. One such area of work is cryptography. Cryptography research can take different forms in the SCADA environment. It can be used to investigate the practicality of retrofitting bump-in-the-wire devices between legacy devices [14], research the practicality of various forms of key management [15], test the practicality of using puzzles to confirm identity in a large scale network [16], or examine protocols such as DNP3 with Secure Authentication [17] or DNPSec [18].

Unique requirements of the power grid motivate the work being done here. The massive scale of the power grid necessitates the use of a virtual environment which can match its scope. Other test beds such as DETER [19] will not scale well enough due their dependence on real hardware. OPNET-based simulators will not scale, because, unlike RINSE, they do not support multi-resolution simulation – a technique that allows high scalable frameworks. Certainly, these test beds provide essential services and are not to be discounted. However, they do present a gap that we hope to fill with VPST.

CHAPTER 3
BACKGROUND

This work positions itself at the intersection of two other interesting topics of research. One of these areas is network simulation. The other is the power grid. Network simulation demands attention to details such as fidelity, performance, and scaling, while the power grid requires insight into the interaction between control and electrical planes as well as the development of new technologies to address SCADA-specific challenges.

3.1 RINSE

Based on the Scalable Simulation Framework (SSF) [20], RINSE is a network simulator that serves as the basis for this research and has been through several iterations. RINSE is currently maintained by a team of developers working at University of Illinois under the TCIP project. Developers are working on a variety of projects including wireless communications, switch modeling, and intrusion prevention through game theory. RINSE has traditionally been used as a wireline network communications simulator to explore the behaviors of malicious behavior in the Internet – namely worms, botnets, and denial of service attacks. The scalability that RINSE allows and the similarity of the Internet to the SCADA infrastructure make it a good candidate for simulating SCADA traffic.

RINSE has a number of properties that make it amenable to large-scale simulation. SSF enables highly parallelizable models by partitioning graphs into submodels. These submodels, which compose the main model, are divided such that communication between them is kept to a minimum. Thus, these partitions can be run in separate processes on separate computers to harness the power of multi-processor environments. Also, RINSE supports multiple resolutions. That is, RINSE can calibrate the fidelity of a simulation to ensure that it runs in real time. It does this by providing a fluid model [21] for traffic, and allowing both full-fidelity traffic and fluid models to exist within the same simulation. These fluid models exist for various components of RINSE such as transport protocols, routers, and links. These models also exist for modeling network topologies, and by utilizing these models, RINSE achieves a significant speedup over using a full resolution model. By using fluid models for background traffic, we can simulate the effect of such traffic without actually modeling it exactly. This allows more extensive evaluation of the models and protocols of importance such as newly designed security devices.

RINSE also supports simulation speeds that are faster than real time. This means that the simulator is not dependent on wall-clock time to advance the simulator time. This is important when interfacing with real devices through emulation. Instead of synchronizing with the wall-clock, the simulator keeps track of its own timeline and, where possible, computes traffic ahead of when it will be needed. Combined with prioritizing emulated packets, RINSE can provide a large-scale simulation environment that is not slowed down by interacting with physical devices.

Addtionally, RINSE is modular in such a way that new protocols and new models can be developed. By implementing new protocols through extending the base *ProtocolMessage*, RINSE can support the integration of existing or nascent protocols. Likewise, the *ProtocolSession* class allows us to develop new layers in the protocol stack. In the case of DNP3, for instance, the relays are derived from the *ProtocolSession* class and communicate with the data aggregators through messages derived the *ProtocolMessage* class. The *ProtocolSession*s that comprise a host are indicated in a file by using the Domain Modeling Language (DML). DML describes a model as a tree of key-value pairs and a file containing a DML model is passed to the simulation engine at run time. These key-value pairs are then interpreted and used to derive the network topology, hosts, protocols, and traffic patterns. This allows RINSE to simulate any number of models without having to recompile, thus increasing the turn-around time between testing iterations of the same model.

At the end of a simulation, RINSE can present detailed statistics regarding such metrics as bandwidth usage and dropped packets. Alternatively, the flow of traffic can be analyzed visually through the network visualizer, a

GUI designed specifically for displaying RINSE simulations. RINSE can also be instrumented to keep track statistics on device-specific functionality, such as throughput of an algorithm. Additionally, traffic from RINSE can also be dumped to a *tcpdump* and analyzed in more depth by tools such as *tcpanaly* [22, 23], a tool developed by Vern Paxson.

3.2 Power Grid

The power grid is an incredibly large, incredibly complicated network that combines both physical and cyber devices. The power grid consists of three main parts: generation, transmission, and distribution. The generation occurs at power plants and can come from a variety of sources, such as nuclear, coal, wind, etc. The transmission lines are the high voltage lines that run across the country transporting elecrical energy from the generation facilities to the areas of consumption. Distribution occurs at a local level, serving as a step-down from the high-voltage transmission lines to the lower-voltage energy fit for consumers. All three of these domains have their own issues, security and otherwise. For instance, security at the generation sites involves armed guards, hardened firewalls, and the like. Their larger concern is safety though, as a large plant operating outside normal operating parameters poses a large risk to personel and machinery. In the distribution regime, a large concern regarding the new smart grid is that of securing private data against disclosure. The transmission section of the grid is smaller in scale than the distribution side, but it has its own unique concerns. Much of the control systems that regulate the transmission lines are now highly interconnected, both to themselves and to the internet. This is an issue that must be addressed and this research has been a step forward in evaluating potential solutions to potential problems.

Traditionally, the grid has used MODBUS over serial connections to communicate between devices. More recently, protocols such as DNP3 in the US and IEC 60870-5 in Europe have been used to communicate over the long distances between substations and control stations. These protocols are also used to communicate between substations and the RTUs in the field. In the substation itself, the trend is currently to use IEC 61850 to communicate. These current and future trends are discussed elsewhere [24].

CHAPTER 4

SIMULATING DNP3

When simulating a new protocol or device, there always exists the question of how accurately we must model the proposed design. However, there exists little question that we must model the design itself, and not force an incongruent model onto it. The option of tweaking parameters of a different model to estimate the new model rarely works because there are often fundamental features that do not lend themselves to be portrayed by a different model. For instance, we could hypothetically model the power grid as a purely TCP/IP oriented network, with parameters such as poll interval, packet size, latency, bandwidth, etc., set to portray the parameters present in the power grid. However, this would completely ignore the idiosyncracies of the grid itself. For instance, if there is a vulnerability in the protocol specification or implementation, it would be agnostic to the network layout and configuration. In this manner, we can say that if we care at all about the quirks and security assumptions of a protocol, then we must model that protocol as accurately as the scale of the network permits.

4.1 DNP3 Overview

DNP v3.0 [25] is a communications protocol that is used to communicate among power grid equipment. Designed to provide interoperability and an open standard to device manufacturers, DNP3 has gained prominence in the United States electrical grid. Versions 1.0 and 2.0 were never released to the public; hence, DNP v3.0 is often referred to just DNP. So as to not confuse this protocol with previous implementations, this paper will refer to it as DNP3. DNP3 is designed to operate in environments with a high electronics density. These environments can be fairly noisy, and therefore, DNP3 is designed to be as robust as possible in regards to detecting and recovering from error. Extensive use of CRC bytes are used to detect when bits have been flipped and a small frame size (292 bytes) is used to localize errors and reduce the overhead imposing by resending frames.

The protocol was designed as a stack of three layers: the data link layer, the pseudo-transport function and the application layer. The general hierarchy can be seen in Figure 4.1. The physical medium is generally either Ethernet or RS-485. Since these standards are common, DNP3 can be run over existing networks or networks can be built from the ground up to support DNP3 SCADA networks. Proving framing information and reliability is the Data Link Layer which can either be run directly on the physical medium as in Figure 4.1 or it can be encapsulated by other protocols as seen in Figure 4.2. On top of the Data Link Layer is the Pseudo-Transport Layer. This simple layer is used to support fragmentation. Finally, the top layer is the Application Layer which acts on behalf of the user for requesting/confirming/sending/receiving requests and data.

4.1.1 Data Link Layer

As with any data link layer, the DNP3 Data Link Layer is responsible for point to point communication. Essentially, what is contained in this layer is addressing. More specifically, the individual fields for this layer can be seen in the top portion of Figure 4.3. All DNP3 messages start with two bytes $0x05$ $0x64$ to indicate the beginning of a frame. Next is one byte for the length of the packet. This length is the number of bytes in the message *disregarding* the CRC bytes.

Following the length is the control code. In the blown-up box of the top portion of Figure 4.3, we see the breakdown of this byte into bit fields:

- *DIR* (Direction): Indicates the direction of data travel.
- *PRI* (Primay): A *1* indicates that the frame is sent by the intiating party, where a *0* indicates the frame is sent by the responding party. This is distinct from *DIR* in the case of *report-by-exception.*
- *FCB* (Frame Count Bit): Indicates whether the frame is an even or odd frame.

- *FCV* (Frame Count Valid): Indicates if the frame counting is enabled.
- *Functions 3-0*: Provides information about the frame. If the frame is from the initiating party this can indicate either a send or a request. A responding party will use these bits to indicate either a confirm or a response.

Following the Control byte are two bytes that give the destination address of the frame (DST). Two bytes also idicate the source of the frame (SRC). The Data Link header concludes with 2 bytes of CRC that are computed against the first 8 bytes. Further information can be found in the DNP3 specification for the data link layer, volume 4 [26].

4.1.2 Psuedo-Transport Layer

This layer is responsible for the seggmentation of the Application Layer into lower level frames. It is a rather simple layer that consists of only one byte. As seen in the middle box in Figure 4.3, this byte is separated into 3 fields. The most significant bit, FIN, indicates whether this frame is the last in a sequence of frames that correspond to one application layer fragment. The second bit, *FIR*, indicates whether this frame is the first in a sequence of frames that correspond to one Application Layer fragment. For a message that can fit into one frame, both of these bits would be set.

The lower 6 bits act as the sequence number, which serves to detect dropped frames, out-of-order frames, and other such errors. Once the counter reaches $0x3F$, it simply resets to 0. Further information can be found in the DNP3 specification for the transport function, volume 3 [27].

4.1.3 Application Layer

The bottom box of Figure 4.3 shows the Application layer. The Application Layer is comprised of two sections – the Application Protocol Control Information (APCI) and the Application Service Data Unit (ASDU). First, we will describe the APCI. The APCI can either be a response header or a request header depending on the purpose of the packet. Both types of headers contain Application Control and Function Code fields.

Like a Transport Control byte, the Application Control byte contains two bits indicating whether the fragment is the first, final, or both fragment in a user level message. Unlike the Transport Control byte, the Application Control byte also contains a bit indicating whether the receiver ought to respond to the message or not. The lowest 5 bits are then used for a sequence number.

Following the Control byte is the Function Code byte. This byte indicates the purpose of the message. The most common function codes are *confirm, read, write, select, operate, response,* and *unsolicted response.* Their exact meanings will not be discussed, but their functionality is fairly straightforward. Perhaps the only two that warrant explanation are *select* and *operate*, which act as a two-step process. First, a *select* command tells the slave which control point will be changed. Then, the *operate* command changes the point, which may control a characteristic such as a threshold value or a fidelity requirement.

A response header also contains Internal Indications which are spread across two bytes as sixteen separate bit fields. Their values will not be described here, but their general purpose is to indicate device status or provide an error message.

The application layer contains the function code for either requests or responses. Following this identification code in a response is any data that the master may have requested. Information can be found in the DNP3 specification, volume 2 [28].

4.1.4 Cyclic-Redundancy Check

Every 16 bytes are entered into a CRC and the next two bytes contain this CRC. DNP3 has its own CRC function of the form $x^{16} + x^{13} + x^{12} + x^{11} + x^{10} + x^{8} + x^{6} + x^{5} + x^{2} + 1$. A CRC is also performed on the Data Link layer header and the two bytes are placed directly at the end of this header, before the Psuedo Transport Layer. It is up to the vendor to decide how they want their device to compute CRCs. One way of doing so is to use a lookup table, whereby every byte has a pre-computed output. The other option is to use an accumulation method whereby each bit is right shifted into an accumulation register. Implementation details for the CRC can be found in a document [29] released by the DNP3 User's Group. This document contains code examples for implementing both of these approaches.

4.2 Attacking DNP3

As one of the primary protocols used to transmit information in the power grid, it is important that we model DNP3 as accurately as possible. One reason for this is that there exist deficiencies in the protocol that allow it to be compromised. For instance, one research group [30] has developed a taxonomy to describe attacks against DNP3. In their paper, they describe 28 generic attacks (of which there are 91 specific instances) which show how vulnerable the protocol is. Countermeasures for these holes could easily be tested in a simulation environment before becoming part of the standard. Another group has compiled a survey of SCADA-related attacks, and discusses techniques such as attack trees, fault trees, and risk analysis as it pertains to CIs [31]. Indeed, much research has been done on both SCADA security gaps [13, 32, 33, 34] and their countermeasures, including data set security [35], SCADA-specific Itrustion Prevention Systems [36], and encapsulating DNP3 in another protocol such as SSL/TLS or IPSec [37]. Clearly, the power grid is susceptible to attack and while the exact nature of these vulnerabilities is out of the scope of this paper, this knowledge provides the impetus to model DNP3 accurately enough to reproduce the vulnerabilities in our virtual test bed.

4.3 Modeling DNP3

No matter the actual implementation of the protocol in the simulator, the protocol must be able to handle all three layers of the stack (data link, transport, and application). Inside the simulator, we treat the three layers as a combined payload to be transported by TCP/IP. Inside the simulator, packets are routed using the IP header as opposed to the data-link layer header. However, when dealing with emulated packets, the data-link source and destination fields are used to direct packets to and from the proper hosts. More information about this can be found in Section 5.3, which discusses how translation is done between the virtual hosts' IDs and their emulated DNP3 adresses.

Dealing with emulated packets is an important concern for our use of this virtual DNP3 model. Being compliant with physical devices enables many potential use cases. Without external communications, the RINSE model would provide limited usefulness. It would provide background traffic, metrics regarding correctness and scaling of technologies, and insight into large scale SCADA networks. However, by being interoperable with physical equipment, more use cases are available which involve a control station. This provides tremendous benefit and some of these use cases are further outlined in Section 9.1.

4.3.1 Approach

Instead of using a full-fledged implementation of the DNP3 stack, we model our own slightly abstract view due to a few different reasons. The main reason for this is scalability. With the potential of modelling hundreds of thousands of relays, it would be nearly impossible to model the full functionality of DNP3. Enabling the functionality to respond to the entire set of DNP3 requests would require more computational power than can be afforded per relay. Instead, by focusing on supporting two classes of reads, with only a few object types, and one type of command, we can greatly simplify the control flow to enable quick computation and low-latency replies. However, if requested by a collaborator, the structure to extend the models to support extra function codes does exist.

Instead of using the DNP3 stack directly, we treat it as a TCP/IP payload. Historically, DNP3 has been used on its own for data link and transport functionality. More recently, however, DNP3 has been encapsulated by TCP/IP so as to take advantage of the routability offered by Ethernet-based technology. As such, it is hardly a stretch to model DNP3 as a TCP/IP payload within RINSE. Likewise, communication between RINSE and the control station also uses DNP3 on top of TCP/IP.

It is obvious why we must explicitly model the transport and application layers as they provide the core functionality of DNP3. However, what is less clear is why we must model the data link layer. Since the IP layer of our simulator provides routing, it would seem that we do not necessarily have to rely on the DNP3 data link layer to route information from one device to another. Some reasons that we shoose to include this layer are that there may be unknown interactions between layers. For instance, if an adversary tampers with a field in the this layer, the application layer may not function properly. As seen in Section 4.2, there are attacks that directly attack the data link layer. Without modeling this layer, we would not know whether defenses properly address the issues. Similarly, there may be vulnerabilities that exist solely at the data link layer, and without modeling it, we may never uncover these faults. Lastly, since our simulator has the capability to emulate nodes (i.e., representing a real host as a stub in the simulation), we must support communication with real hosts. All the pieces of physical equipment in our lab require a well-formed packet in order to function properly. If we fail to deliver that, then our simulation is a failure.

In addition to this, DNP3 is fairly flexible. For instance, it can run on a multitude of physical layers, and in the real world, RS-458 and Ethernet are the two most common. We chose to implement DNP3 as a payload to be encapsulated by TCP/IP. This allowed us to route packets more easily and communicate with the control server. Another such decision that is left up to the vendor is how to implement the CRC function. We chose to implement it as a shift/accumlator, since this method has a constant calcuation time. The table lookup method, on the other hand, is faster in the best case scenario, but endures a penalty if the table is ever evicted from the cache. The RINSE implementation for this function was based on an algorithm released by the DNP3 User's Group [29], with some modifications to fit within the RINSE framework.

4.3.2 Drawbacks

Of course, with many of the design decisions we have made, there are bound to be some drawbacks. Also, modeling the three layers as one layer can speed up simulation time, but it also means that DNP3 cannot be used on its own to provide any of its functionality. Currently, this is not a problem, but if there is a need to model DNP3 directly on top of the physical link, it would require a reworking of the DNP3 implementation. Additionally, it would require a rewrite to the way RINSE handles routing as it currently routes based on IP address. In illumnating these drawbacks, it is our hope that we have further illustrated the tradeoffs of our design decisions. Where possible, speed and scalability have been optimized over other characteristics.

4.4 Figures

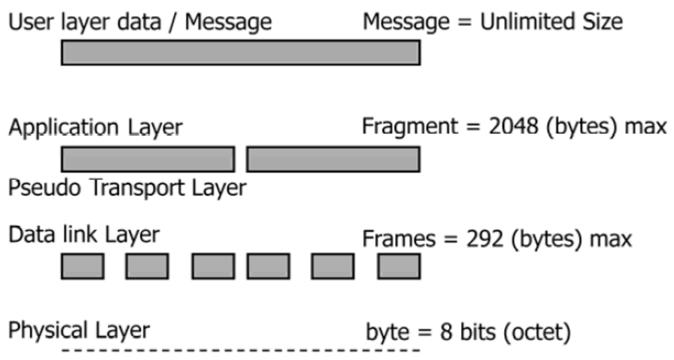


Figure 4.1: DNP3 Protocol Stack [38]

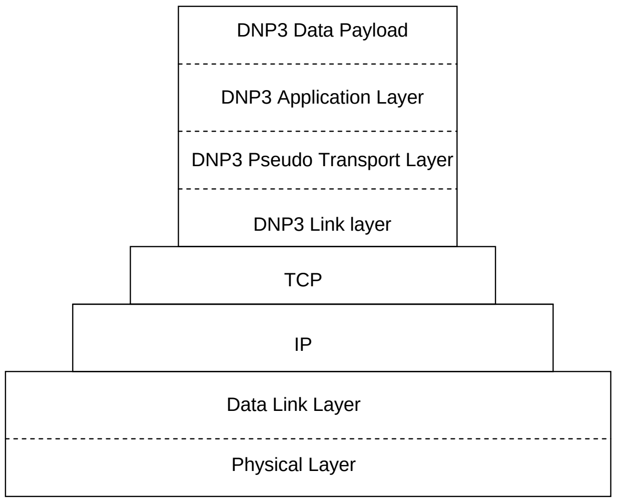


Figure 4.2: DNP3 Protocol Stack Encapsulated by TCP/IP

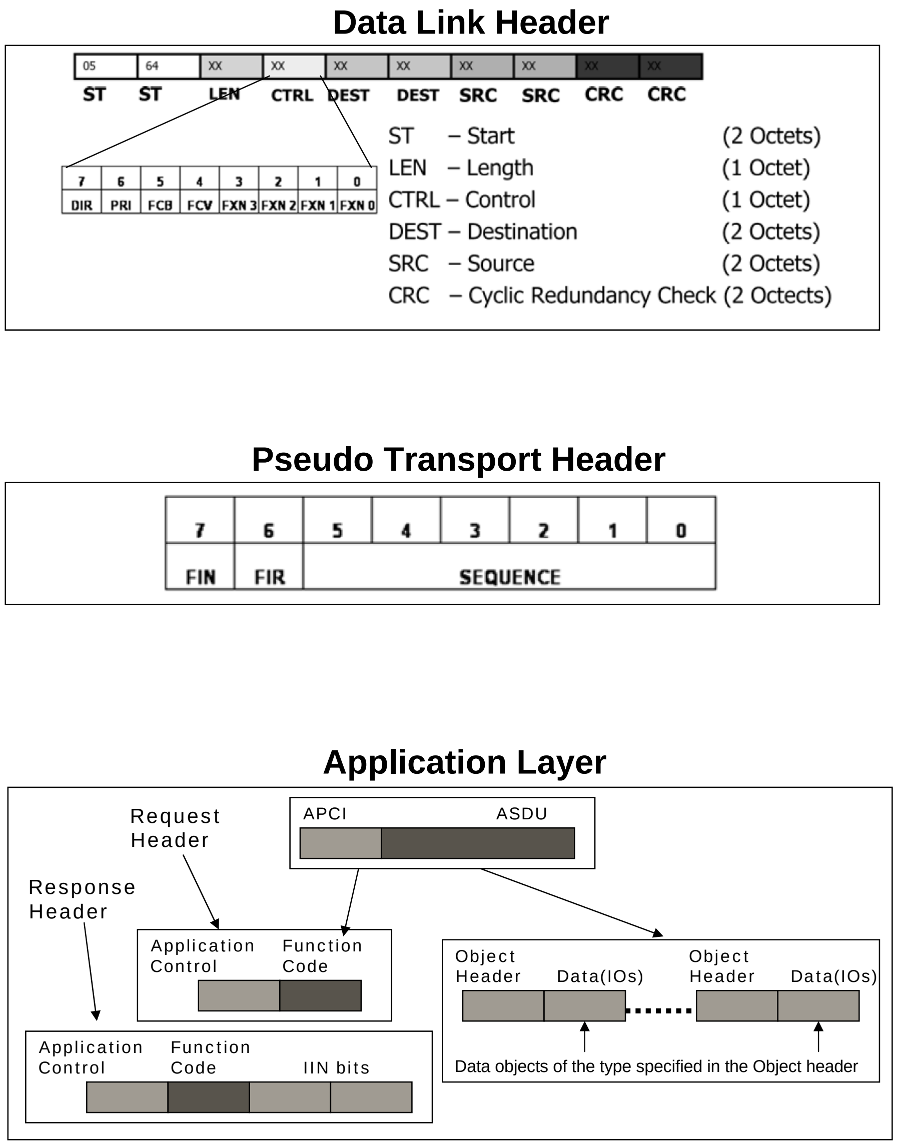


Figure 4.3: Three Layers of DNP3 [38]

CHAPTER 5

VIRTUALIZING HOSTS

A *host* is the term we give to nodes in the simulator that have some sort of computational power. That is, a host models some real world device. Traditionally, these hosts have modeled routers, servers, and clients. In this project, hosts have been extended to model the various computational entities in a power system. Namely, we have chosen to simulate data aggregators and relays. On the other hand, we have chosen *not* to simulate control stations, due to a few factors including their complex structure, their proprietary nature, and heavy customizability. Instead, we rely on our real control station, which provides the added benefit of allowing a human to view the simulated network as part of a real network or as a real network in itself.

5.1 Virtual Data Aggregators

The virtual data aggregators have a control flow similar to that of the virtual relays. However, since they require data from the relays instead of PowerWorld, their polling patterns are somewhat different. Instead of sending out one data request per window, it sends out n requests per window, where n is the number of relays reporting to it. It does this in a round-robin fashion, cycling from relay 0 to relay $n - 1$.

The control flow for the master thread, seen in Figure 5.2, describes how the virtual data aggregator polls the relays for which it is responsible. First, the data aggregator waits t seconds before beginning the round of requests. Starting with relay 0, the data aggregator prepares a request, and waits the appropriate amount of time that its physical analogue would take to produce the request. Then it sends the request and waits for the reply. Once it receives the response, the data aggregator then begins processing the command. Processing the command depends on which requests were requested and how the response is formatted. In the basic scenario, the data supplied by the response is simply entered as a corresponding entry into a table, which can be queried by the control station. Once the response is processed, the data aggregator moves on to the next relay and starts the request flow again. Once the data aggregator has polled all of the relays in its list, it waits until the next polling period and resets the current relay to 0.

The data aggregator also acts as source of data for the control station. This responsibility is handled by the virtual data aggregator's slave thread, as illustrated by Figure 5.3. The slave thread essentially acts as a server, remaining idle until it receives an incoming request from the control station. When it receives a request, it begins to process it. The data aggregator then makes a decision based on the function code in the request. If the function corresponds to a read, then the data aggregator will provide the requested data out of its table. If the function is some sort of command, then the data aggregator passes the command to the correct relay and waits for a response. Once either the relay responds or the data is ready (depending on which function code was sent to the data aggregator), then the data aggregator prepares a response to the control station. Then the data aggregator waits for an appropriate delay, corresponding to the delay its physical analogue would take for preparing the packet. Finally, the data aggregator sends the response and goes back into a waiting state.

5.2 Virtual Relays

Relays are responsible for control of physical lines as well as gathering data pertaining to their operation. Relays must determine various characteristics of these lines, such as phase angle, voltage, real and reactive power, and other such values. Additionally, relays must provide information about their own operation such as status values, counters, and synchronization efforts. It is unrealistic to support all of these features in a virtual relay, as that would severely hamper scalability. In particular, this would introduce extra computation, which would result in unfavorable latency. As such, it is important to determine which subset of features must be supported in order to provide the largest functional coverage at the lowest computational cost.

The functionality that covers a large majority of typical requests is as follows:

- *Read Class X data* – X can be 0, 1, 2, or 3. Different values of X represent different priorities, where 1 refers to data that changes the most often, and 3 the least. Class 0 data includes all data points. This covers such data points as status, voltage, power, phase angles, and many others. Since PowerWorld provides some of these values either directly or indirectly (through simple mathematical operations), this sort of data can be provided to the data aggregators at the substation level.

- *Turn breaker on/off* – A request sent to the relay to turn on or off can then be passed on to the state server.

The primary function of the virtual relays as modeled by this project is to respond to commands issued by the virtual data aggregators. They *do not* detect events and issue unsolicited responses, as their physical analogues are capable of. This difference is described in detail in Section 13.1

The control flow for virtual relays can be seen in Figure 5.1. When not processing a request, the relays are sitting idle. During this time, they are waiting for a data aggregator to issue a request. Once received, the relay determines what type of request this is; currently, the relays support a limited subset of what their physical analogues support. The relay will thus determine if it can answer the request, and if it is either a data read request or a command, it will perform the required actions. If the request is a read, then the relay will retrieve data from the state server through its own designated shared memory. If the data aggregator request is a command, the relay takes a corresponding action by writing to shared memory. The state server is then responsible for forwarding this command to PowerWorld. Once the appropriate action is taken, the relay prepares a response. The response contains either the data or a confirmation, depending on whether the request was a data request or a command, respectively. Once sent, the relay re-enters the idle mode, and waits for the next request.

5.3 State Server

The state server acts as a single point of contact between the RINSE simulation environment and the PowerWorld simulation environment. The same procedure that generates the DML file also generates a mapping between PowerWorld entities and the virtual relays that are assigned to monitoring them. Using this mapping, the state server generates requests according to the PowerWorld API. The requests are split into four different groups – lines, generators, loads, and shunts. After PowerWorld processes these commands, it sends back a response with the appropriate values.

PowerWorld was designed with a server thread which can serve data to external applications. This has been used in the past for a MODBUS converter, but no previous implementation for DNP3 has been developed. Hence, we have created this state server, whose control flow can be seen in Figure 5.4. This figure can best be understood as four similar tasks being carried out in succession. Polling periods start by preparing a request for shunt values from PowerWorld. After waiting for a response, it processes the response and moves on to requesting generator values. The state server then repeats the process for load and line values. Finally, the state server waits t seconds before starting the process over again.

Once the data is transferred from PowerWorld to RINSE, it is up to the state server to partition the data. Depending on which values were requested, the amount of data a relay will receive can vary (typically, the values available are real power usage, reactive power usage and status, which are represented by 5 to 9 bytes). Thus, the state server splits up the response into appropriately sized portions and shares them with the relays through shared memory.

This choice of using shared memory as opposed to explicitly passing a message with the data was made to better represent the real world. Even though the state server has no real world counterpart, its sole role is to deliver data from PowerWorld to the virtual relays. As such, its latency should be no greater than the time it takes for a relay to measure the line it is connected to – essentially zero. Transmitting this information over the routing network would introduce latency into the system that has no real world analogue.

5.4 Figures

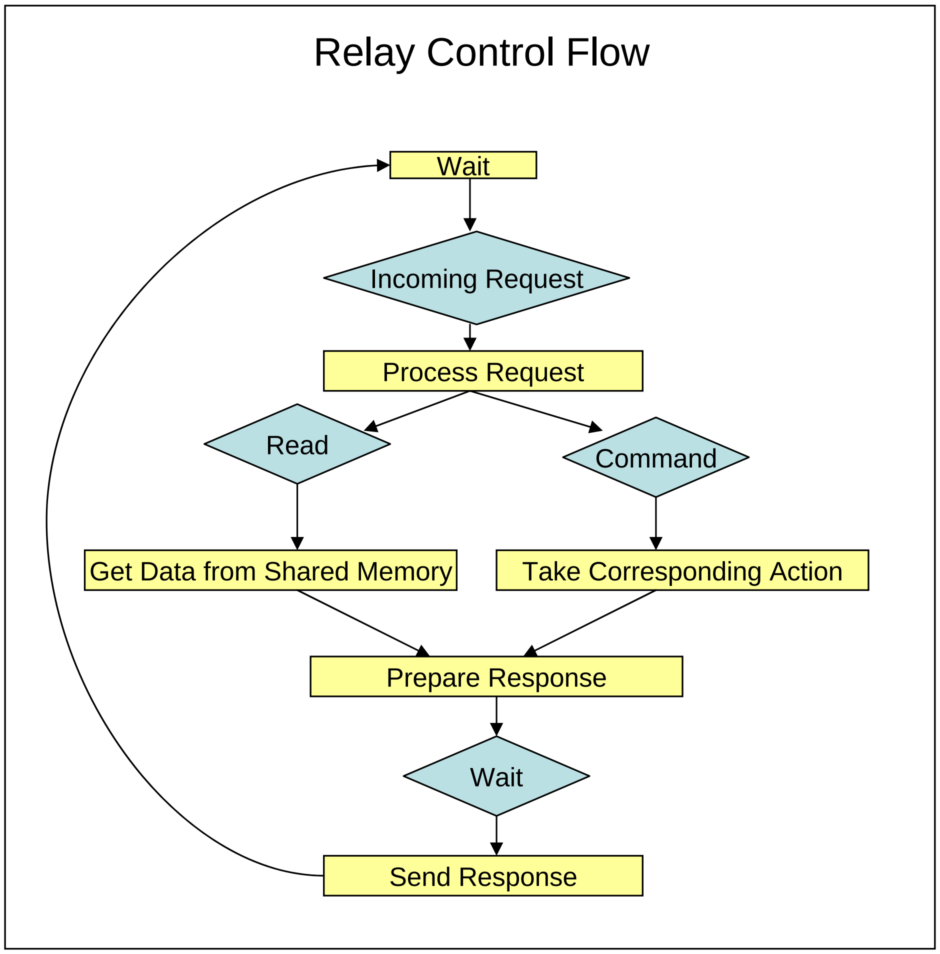


Figure 5.1: Virtual Relay Control Flow

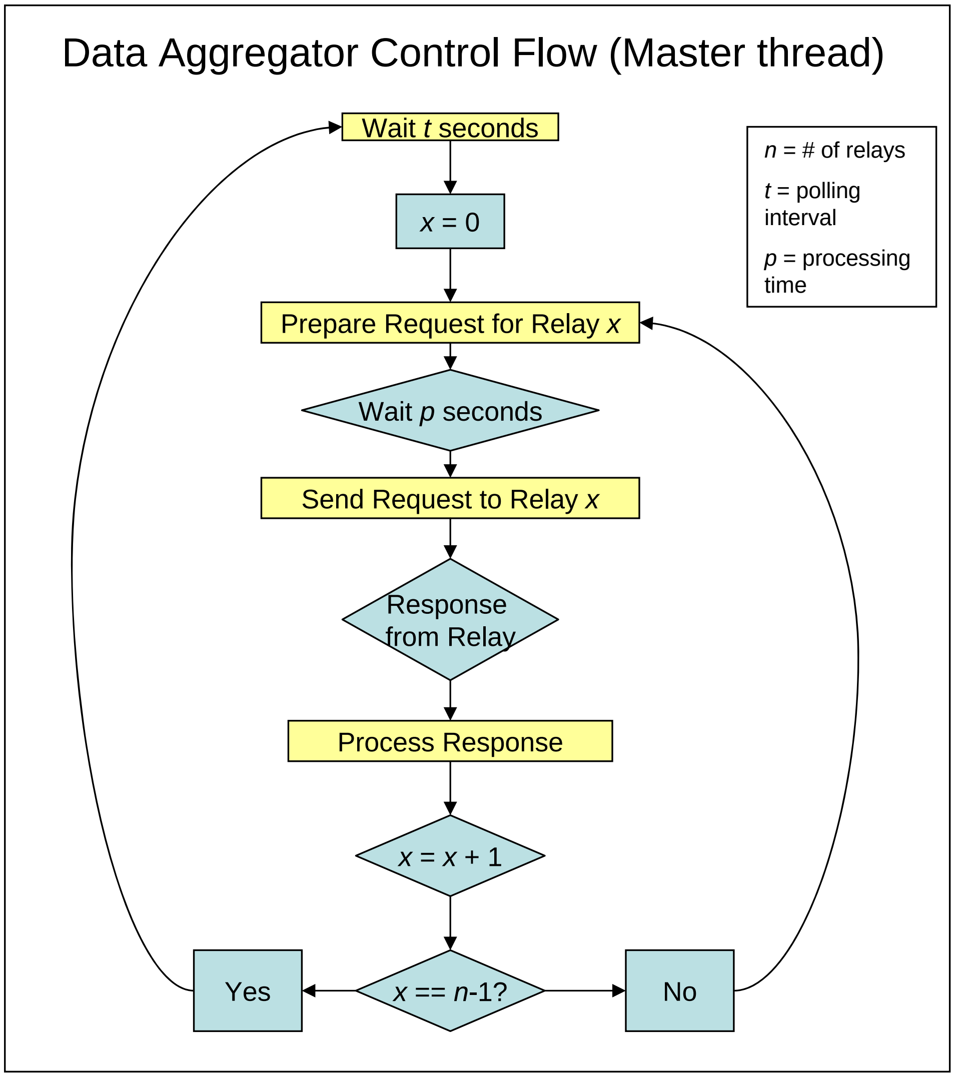


Figure 5.2: Data Aggregator Control Flow (master thread)

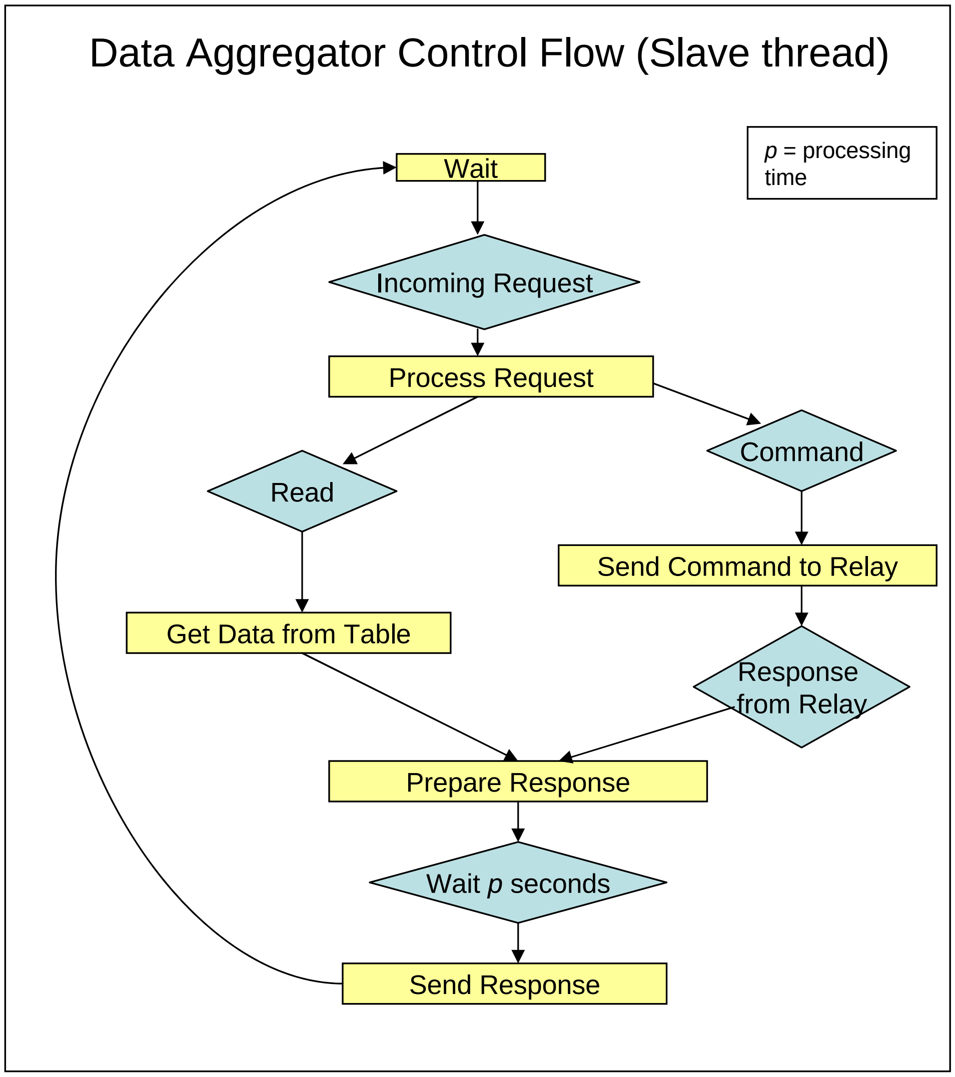


Figure 5.3: Data Aggregator Control Flow (slave thread)

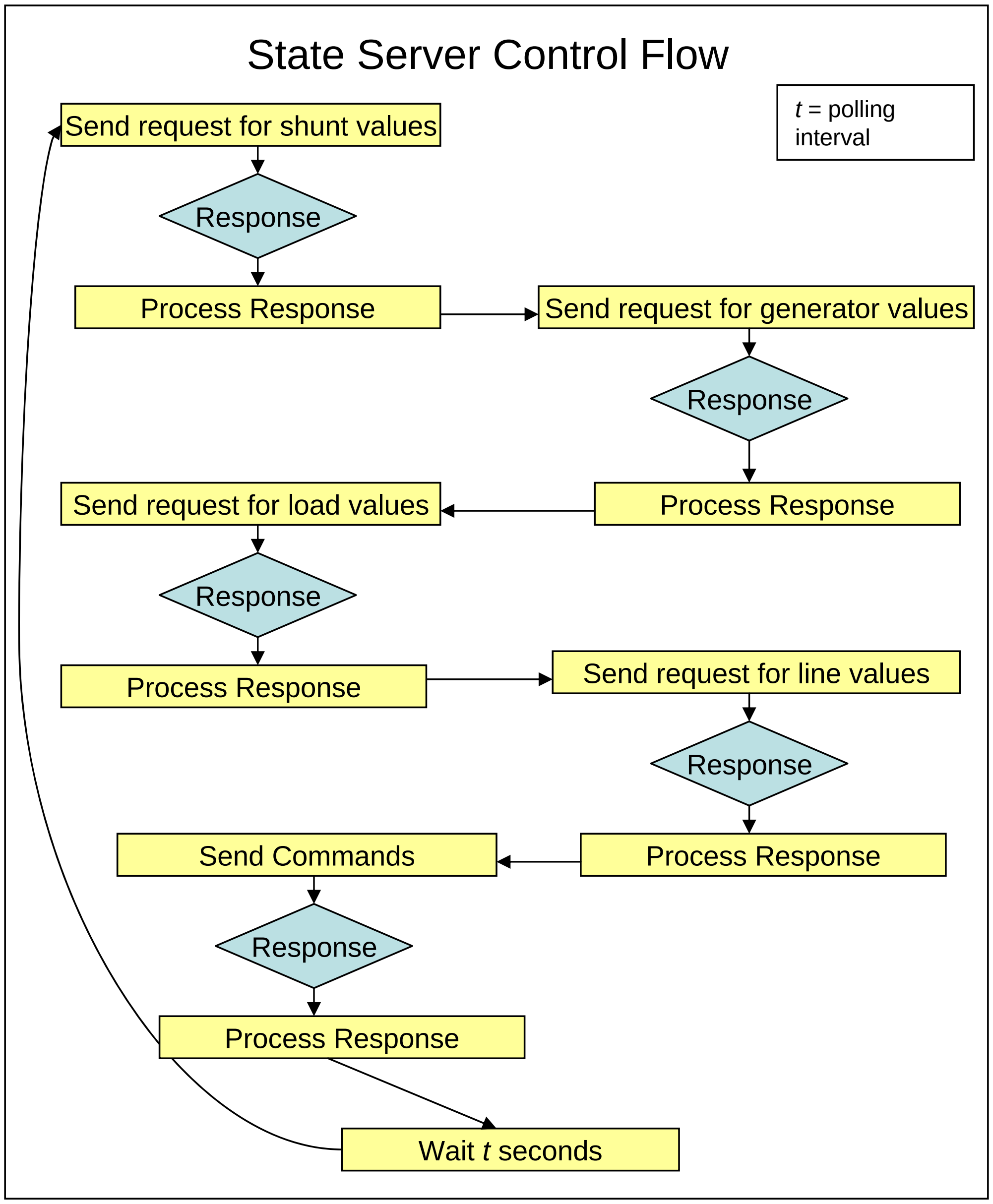


Figure 5.4: State Server Control Flow

CHAPTER 6

LOCAL LAB ENVIRONMENT

With virtual relays, data aggregators, and a protocol, we can now combine these to produce a usable system. Figure 6.1 shows how this system comes together. Our laboratory contains equipment donated from generous donors. Due to their generosity, we have quite a few SCADA devices, software, and support. The lab contains a variety of electronics: SCADA devices, workstations, and consumer-side equipment. While some of this may be integrated in the future, VPST currently utilizes two SEL-421 relays, an SEL-3351 data aggregator, and a workstation running the OSI Monarch OpenView EMS software suite [39]. The relays are each attached to an SEL-AMS. The Adaptive Multichannel Source (AMS) is intended to support lab testing of SEL devices, and provide sinusoidal waveforms which feed into the relays.

Our communications simulator, RINSE, provides the SCADA simulation for our local lab environment. Its components have been described in Chapters 4 and 5. Using these components, we can generate any topology we wish. Shown in Figure 6.1 is a simple topology that includes a virtual control station proxy, multiple substations with one data aggregator and multiple relays per substation, and state server that communicates with a virtual PowerWorld proxy. The figure shows boxes labeled as substations, which currently consist only of a data aggregator. As more devices are modeled and added to the substation domain, this model may or may not change. The PowerWorld proxy is directed through emulation to a computer running PowerWorld which runs a steady-state power simulation. PowerWorld has an API that can interface with external devices using its proprietary protocol. There is a server package that uses this API to service MODBUS responses to the EMS as though there were MODBUS relays responding. In this figure, we do not show PowerWorld interfacing with anything besides RINSE, although the capability is there, and may certainly be beneficial to certain use cases.

In order to provide emulation support, we must also utilize proxies hosted on physical machines. They are not shown in Figure 6.1 as they would only serve to clutter the illustration. However, our setup would not work without them. The purpose of these machines is to redirect traffic from themselves into RINSE, acting like a trampoline. Since RINSE operates on its own virtual private network, in order to direct traffic towards one of the virtual nodes, the traffic must orginate from within that virtual private network. In order to provide this functionality, RINSE comes equipped with a gateway that supports OpenVPN connections. From a workstation, we connect using an OpenVPN client to the OpenVPN server on RINSE's host machine. This allows a physical machine to be in the same private address space as the virtual machines. Once this connection is complete, we can redirect this proxy machine's incoming traffic to any destination inside RINSE. We also can redirect any traffic arrive on the private network address to physical computers. In this manner, the State Server can communicate with PowerWorld in both directions. By redirecting different ports to different addresses, we can allow the Control Station to poll any device in RINSE so long as it is set up to redirect on the proxy machine.

The lab is on a switched network, which provides two useful capabilities. The first is that we can use WireShark to look at traffic between all of these hosts. This allows us to troubleshoot communications between of the devices which is important since the relays only support maintentance through *telnet*. For instance, if we notice that the control station is no long updating, we can pinpoint where the break in communication is. When working with physical proxy machines, the ability to trace packets throughout the network also assists in determining if a link has been accidentally disabled. In addition, it also allows us to inspect the packets themselves. This has proved helpful for timing requirements as well as modeling both DNP3 and the virtual hosts.

6.1 Figure

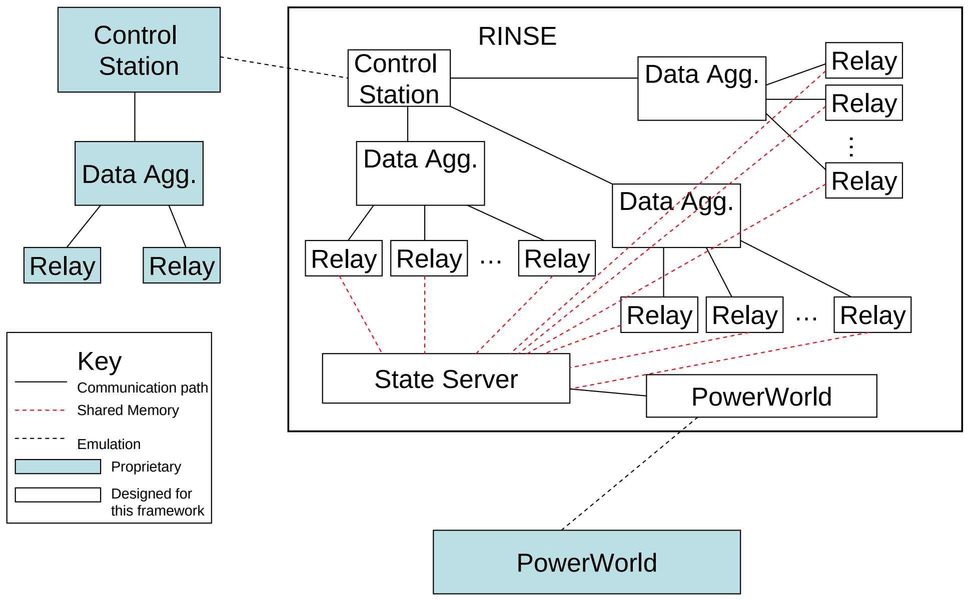


Figure 6.1: Simulation Framework

CHAPTER 7

DEVELOPMENT AND VALIDATION

When building a system such as this, fidelity is an important concern. That is, the system must perform in the same manner as the real-world analogue. There are varying degrees of fidelity, and development of this system has gone through phases addressing different levels of fidelty. First, the system must interact with physical equipment in such a way that the physical equipment does not realize that it is operating inside a partially virtualized environment. This is termed interopability. Second, the virtual hosts must accurately reflect such characteristics as response times and accuracy of data. This can be acheived by tuning the corresponding parameters. Third, verification must be run on the virtual system itself to ensure that no race conditions exist in the modeling code. Finally, since SCADA network information is closely guarded, an expert must be brought in to verify that the virtual network reflects a real-world network.

7.1 Interoperability

The first goal in creating a virtual host is allowing it to communicate with an external device. Reaching this goal will take several steps. First, we create a new virtual host which receives a transmission and responds with a hardcoded message. This hardcoded message can be devised by examining traffic between real world devices. For instance, when devising responses for virtual data aggregators, we look at requests from the real control station to the real data aggregator and copy the response. With this model, it allows us to test the communication channel. The communication channel, which is made up of the virtual networking stack, a virtual proxy, a physical proxy, a physical routing network, and a physical end-host, can have some snags that need to be worked out before working with a more dynamic response.

Once the physical end-host receives and successfully decodes the hardcoded message, we consider this stage a success.

The next step is to generate responses dynamically. In order to do this, there are two steps. The first step is to create a virtual analogue of the physical protocol. In this case, the protocol is DNP3.0 encapsulated by TCP/IP. Within RINSE, this is represented as a *ProtocolMessage* with a corresponding *ProtocolSession*. The *ProtocolMessage* class contains member data to correspond to each of the fields specified in Section 4.1. When communicating with a physical host, the class structure must be converted to a byte stream and there are functions that perform this for both directions. Additionally, there is a helper function to compute and interpose the CRCs when converting to a byte stream.

After the protocol was implemented, the next step was to produce the dynamically generated message with static data. To do this, we extract the DNP3.0 request, convert it to a DNP3.0 *ProtocolMessage*, and process it. The control flow has been discussed in Section 5.1.

Once we ensured that the virtual data aggregator was able to correctly generate a response with static data, it was time to generate a response with dynamic data. This is when work on the state server began. In order for the control station to observe a changing environment, responses from the data aggregators must reflect a changing environment. Therefore, each of the relays must also derive its state from a changing environment. Each could poll PowerWorld independently but this would not scale well. Instead, we chose to implement a single point of contact – namely the state server. In this manner, we have eliminated costly overhead by compressing multiple requests and responses into one.

This design model can serve as a model for integretating different virtual hosts into this framework. Perhaps it can also aid in developing other frameworks for both the RINSE platform and others.

7.2 Tuning Parameters

Once we have a generalized virtual model, it can be tuned to represent any number of physical analogues. By tuning parameters such as polling interval, polling pattern, and response time, we can model various types of data aggregator computers. New features could also be added by specifying them in the DML without designing brand new virtual data aggregators. Virtual relays do not have any polling interval (the data is already available upon receiving a request); however, they can be matched to their physical analogues by changing their response time. Were a virtual MODBUS protocol created, another parameter in DML could be specified for protocol support. If a new physical device cannot be modeled by changing some of these parameters, then it may be a wise idea to investigate either altering the model or creating a derived class or an entirely new class of device.

As for what we have done in the lab, we have analyzed physical data aggregators and physical relays by observing their polling patterns and response times. Then, we have tuned the parameters such that these match as closely as possible. We also check to ensure that the model size does not impact the response time for a relay. We set *relay_proccessing_time* = 2 ms for multiple model sizes and compare the results, which can be seen in Table 7.1. As the table shows, response times do not depend on the model size. This is key to operating in a large-scale simulation; if the response times changed based on the size of the model, then evaluations of new technologies against scaling environments could not be compared between model sizes.

7.3 Expert Analysis

Due to the criticality of the power grid, it is difficult to acquire network topologies. It is also difficult to acquire the designs for SCADA devices from their vendors. In addition to creating these models based upon observing the lab devices, some power experts in TCIP have offered their expertise. Thanks to them, the virtual models in this project have been through multiple iterations. Aspects such as polling patterns and the parameterization of virtual models appear to be sufficient. Network topologies can be managed through DML files and do not require RINSE development access, and therefore are not at the core of this analysis. The topologies described in previous sections are simplified, but accurate enough to present a good picture. The two suggestions that ought to be tackled next are expanded functionality of the meters and extra virtual models for other devices. This ought to be done on a case-by-case basis, though, as model devlopment should be guided toward a specific goal and not just for the sake of having extra models. The most important thing, though, is that the models contain enough parameters to allow an experimenter to model a device without delving into the RINSE source code.

7.4 Table

Table 7.1: Average Response Time For Varying Model Sizes

Size of Model (# relays)	Average Response Time of Read Requests (ms)
10	2.02820
50	2.02818
100	2.02822
200	2.02817

CHAPTER 8

SAMPLE WORKFLOW

In this section, we will describe a typical workflow for testing a new technology within the local test bed. A third party who is interested in utilizing our test bed will come to us to present their problem. From there, we will discuss with them what they require, what we can provide, and how to ensure the largest overlap between the two. A test plan will be written, detailing the procedure required to implement and test the new technology within the VPST framework. Preferably, desirable outcomes will also be discussed and agreed upon before testing commencing. A sample workflow beginning with the power flow model creation is as follows.

8.1 Designing the Power Grid Layout

The first step in the process is to design a power network that exhbitits traits that might be interesting. Someone who is knowledgable of a typical power grid network will design this, as their knowledge should translate into a realistic representation. One such power network can be seen in Figure 8.1. This figure shows a power transmission network system for a large city, as displayed by PowerWorld. PowerWorld itself offers many tools for designing, interacting, and assessing power flow models. For the purpose of VPST, the important features that it offers are the abilities to manually create and export a design as well as support a real-time simulation. In order to interact with the real-time simulation, a tool has been created to serve the current snapshot's power to client applications. This feature is not available in the commercial release of PowerWorld at the time of this writing.

8.2 Setting up the Lab

From the PowerWorld simulation, key information will be extracted. Typically, these are the IDs of lines, buses, generators, shunts, and loads. From here, we generate the DML, set up proxies, and the EMS configuration. The corresponding RINSE model is shown in Figure 8.2. From the PowerWorld view, 36 buses and 196 distinct connections were found. These distinct connections have been categorized into lines, loads, generators, and shunts and are then represented by relays in the DML. As can be seen in the zoomed-in box in Figure 8.2, the relays are then annotated to provide a reference point in the PowerWorld model. Likewise, the corresponding EMS configuration in Figure 8.3 shows a list of RTUs (i.e., data aggregators) that the EMS is aware of and is able to poll for data. Currently, configuring the EMS is done by hand, but work is being done to automate this process as well.

This portion of the workflow is set up to be as automatic as possible. There are scripts to extract the information from PowerWorld, to create the DML model and its corresponding routing information, to log in the proxy machines and set up the redirects, and we are currently developing a script to produce the control station configuration file. Once this portion of the project is complete, it may even be possible to reproduce the entire lab setup in another location, provided similar resources exist there as well.

8.3 Integrate New Technology

In order to integrate a new technology into the framework, there exist two options. The first option is to provide a standalone device or model that can be accessed through a network device. By doing this, the actual technology is tested, as opposed to a recreation. The way in which this is done is similar to how the State Server receives its values from PowerWorld (see Section 5.3). There would exist a virtual proxy of the device inside RINSE and a physical proxy outside of RINSE. Traffic directed to the virtual proxy will be forwarded, through emulation, to the physical proxy. From there, traffic is redirected to a device hosting the new technology. Responses follow a similar route through the physical proxy, through the virtual proxy, and to the target host inside the simulator.

The other option is to incorporate a model of the technology directly into RINSE. When using this option, the technology owner will need to work with a point of contact in the RINSE development team in order to develop an accurate model of their technology, whether that is a protocol or a bump-in-the-wire device or anything else. RINSE is written in C++ and as such can accept a C++ implementation of the new technology. The original implementation must be ported to C++, allowing some modifications to provide networking support inside RINSE. Further work is planned to make RINSE as modular as possible, to ease extensibility for cases such as this.

There are trade-offs to consider here, which must be evaluated on a case-by-case basis. For instance, in the C++ implementation case, there are issues of fidelity and time of development. It takes time to develop a model and ensure that it corresponds to the actual technology. There are also issues of ensuring that the C++ model is in lock-step with the current iteration of the product. In the emulation case, there may be issues of scalability, latency, and availability of measurement data. If time allows, implementing the technology directly inside of RINSE will usually allow for better results, as the full benefit of scalable simulation can be realized.

8.4 Run Experiment and Draw Conclusions

Once all of the connections are set up and the entire communication flow is shown to be working, it is time to test the technology in *in silico*. During the course of the experiment, it is possible to interact with the model and test how it reacts to certain commands. For instance, the EMS software can provide a place for an operator to sit and monitor the grid. Depending on the new technology, it may become easier or harder for the operator to understand what is happening in the system. For instance, new technology implemented in RINSE may simplify the operator's job so that fewer errors occur. On the other hand, it may accidentally drop communication under certain conditions. Likewise, PowerWorld can offer information about the performance of a new technology. For instance, if a command sent from the EMS never produces a change in PowerWorld, this can be seen quickly in PowerWorld and root-caused in RINSE. Once the experiment has run its course, it is time to examine the results. Either they will match what was expected, and it is time to move on to the next stage, or they will not match, and it is time to figure out why they do not match. Finally, RINSE itself can provide valuable information. RINSE tracks metrics such as bandwidth usage and dropped packets, and can be used to determine average latency for communications. RINSE also provides debugging functionality through detailed logging and *tcpdump* files that can be used to do post-mortem evaluation of an experiment.

From these results, we can draw conclusions as to whether the new technology positively or negatively affects the communication network. After understanding the implications of the current version, the researcher may enhance the design and begin the cycle again. The hope is that this cycle has a quicker turn-around than testing the software in isolation or going through the hardware development process. This can be realized through some important facets of RINSE – namely, network creation at run-time, easily configurable network topologies, and immediate feedback. By changing parameters such as background traffic, link bandwidths, link latencies, and topologies, RINSE provides a way to test a technology in a diverse set of configurable environments.

8.5 Figures

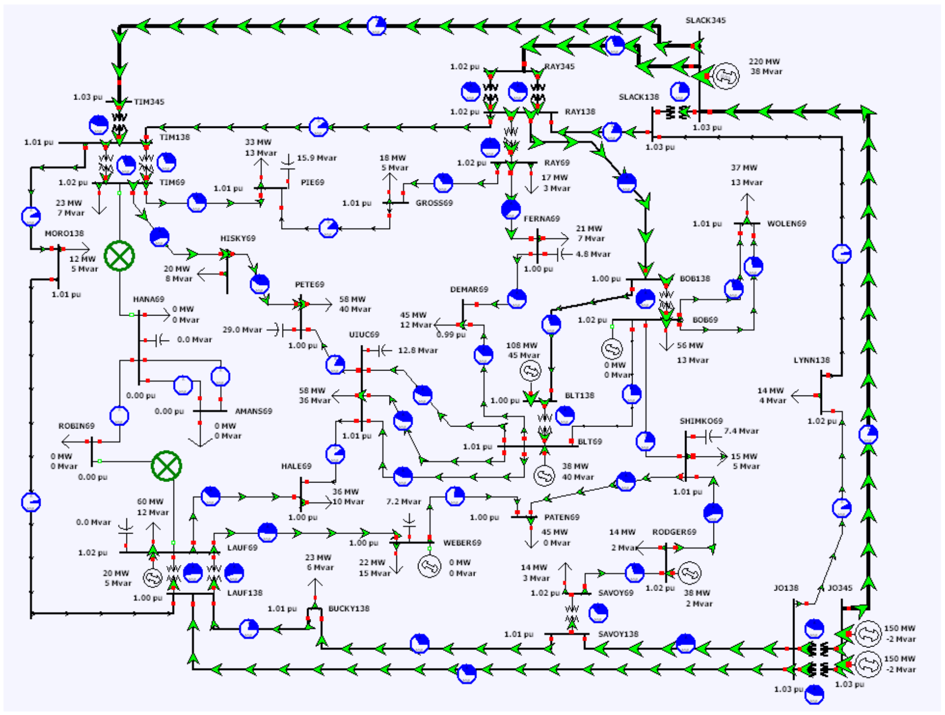


Figure 8.1: PowerWorld Sample Design

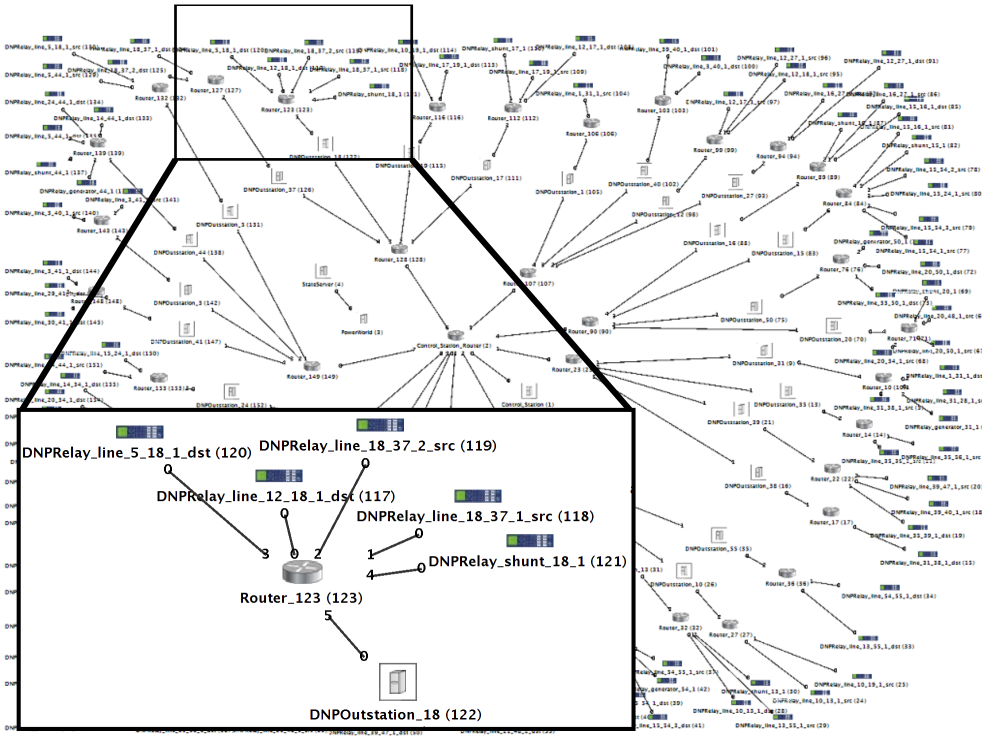
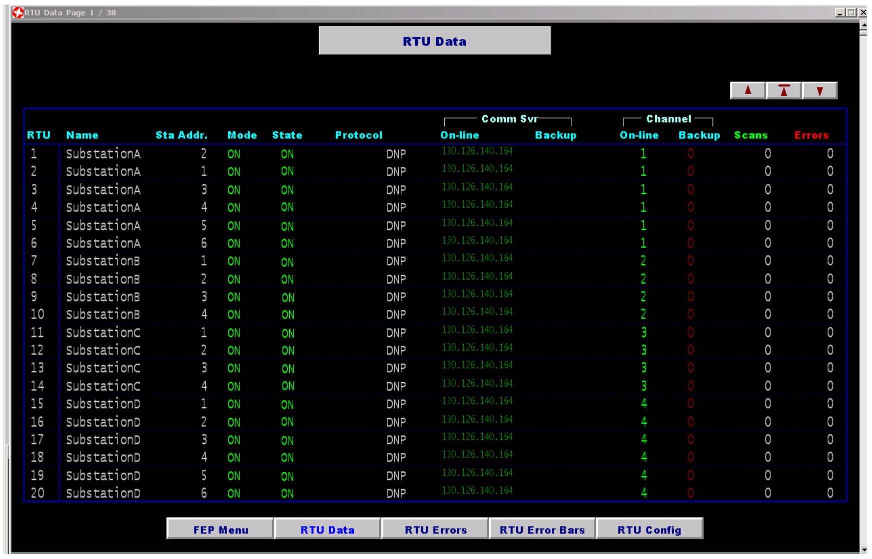

DNPRelay_line_5_18_1_dst (120)
DNPRelay_line_18_37_2_src (119)
DNPRelay_line_12_18_1_dst (117)
DNPRelay_line_18_37_1_src (118)
DNPRelay_shunt_18_1 (121)
Router_123 (123)
DNPOutstation_18 (122)

Figure 8.2: Corresponding RINSE View



RTU Data Page 1 / 38

RTU Data

						Comm Svr		Channel			
RTU	Name	Sta Addr.	Mode	State	Protocol	On-line	Backup	On-line	Backup	Scans	Errors
1	SubstationA	2	ON	ON	DNP	130.126.140.164		1	0	0	0
2	SubstationA	1	ON	ON	DNP	130.126.140.164		1	0	0	0
3	SubstationA	3	ON	ON	DNP	130.126.140.164		1	0	0	0
4	SubstationA	4	ON	ON	DNP	130.126.140.164		1	0	0	0
5	SubstationA	5	ON	ON	DNP	130.126.140.164		1	0	0	0
6	SubstationA	6	ON	ON	DNP	130.126.140.164		1	0	0	0
7	SubstationB	1	ON	ON	DNP	130.126.140.164		2	0	0	0
8	SubstationB	2	ON	ON	DNP	130.126.140.164		2	0	0	0
9	SubstationB	3	ON	ON	DNP	130.126.140.164		2	0	0	0
10	SubstationB	4	ON	ON	DNP	130.126.140.164		2	0	0	0
11	SubstationC	1	ON	ON	DNP	130.126.140.164		3	0	0	0
12	SubstationC	2	ON	ON	DNP	130.126.140.164		3	0	0	0
13	SubstationC	3	ON	ON	DNP	130.126.140.164		3	0	0	0
14	SubstationC	4	ON	ON	DNP	130.126.140.164		3	0	0	0
15	SubstationD	1	ON	ON	DNP	130.126.140.164		4	0	0	0
16	SubstationD	2	ON	ON	DNP	130.126.140.164		4	0	0	0
17	SubstationD	3	ON	ON	DNP	130.126.140.164		4	0	0	0
18	SubstationD	4	ON	ON	DNP	130.126.140.164		4	0	0	0
19	SubstationD	5	ON	ON	DNP	130.126.140.164		4	0	0	0
20	SubstationD	6	ON	ON	DNP	130.126.140.164		4	0	0	0

Figure 8.3: Corresponding EMS View

CHAPTER 9

VPST EXTENSION

From here on, the thesis will describe extensions to VPST that provide the functionality required to integrate VPST with remote test beds. First, we will discuss potential use cases that motivate this interconnection.

9.1 Use Cases

9.1.1 Training and Human-in-the-loop Event Analysis

Early detection and prevention technologies are required, as was learned from the massive mid-western blackout in August 2003 [4], occurring largely because diverse failures caused a lack of situational awareness on the part of the operators. Had there been different detection mechanisms in place, the evolving disaster might have been recognized sooner, and corrected. Electrical system data is continuously read and stored which allows system state to be replayed on the testbed, presenting operators with the same data as existed at the time of the blackout (or some other notable event). However, the testbed can provide real or simulated versions of new technology. This allows operators to react to what they *now* see (owing to the simulation), which may be different from the original incident. As different control decisions are made (e.g. open a breaker) the testbed shifts from playing back observed state information from the original event to generating a new state trajectory, based on the new control decisions. By integrating with real device/emulation testbeds, VPST has two significant benefits over a traditional power system training simulator: (1) actual hardware is in the loop, (2) the communications system is modeled, allowing a more faithful interaction.

This use case requires the testbed to ensure the following: secure connectivity for protecting sensitive information such as strategic decisions; reproducibility for event replay and analysis; scalability for large-scale power grid experiments; and flexibility and fidelity for easy construction of realistic scenarios from the operators' point of view.

9.1.2 Analysis of Incremental Deployment

Securing the power grid requires an overhaul of the existing infrastructure. However, the size and scope of the grid necessitates that the change be gradual, not instantaneous. When old and new technologies coexist, there is the possibility that unforeseen interactions may occur. For instance, current communications make heavy use of the DNP3 protocol [25], yet this protocol provides little in the way of security. Effort has been put into creating an extension to DNP3—DNP3 Secure Authentication (DNP3SA)[17] that addresses authentication concerns with DNP3. As this is deployed, the legacy support must be maintained. DNP3SA is just one example of many. We may also want to analyze strategies for incremental deployment of new technologies, tested against a multitude of network conditions (e.g. lossy networks, congested networks, insecure environments, while under attack, or with the inclusion of corrupted data packets) and collect related statistics (e.g. bandwidth usage, latency, dropped packets, success ratio for communications, overhead incurred, etc.).

When testing a new protocol, device, or policy, there are a few important requirements: reproducibility to ensure that any changes are a direct result of the new technology in question; high performance in that we must accurately model the scale of the real power grid; customizability in order to provide a quick turnover from one configuration to another; and high fidelity in order to guarantee that a new technology behaves the same in simulation as in the real world.

9.1.3 Attack Robustness Analysis

A third use case is analyzing the robustness of a design against an attack. For instance, the DETER testbed has a highly provisioned communications network that is often used to test new protocols against well-defined attack models. Also, Idaho National Labs (INL) has SCADA equipment for use in experiments. However, there is currently no safe way to launch an attack on a large-scale SCADA network – VPST provides such an avenue for testing the reliability of a SCADA network in the face of an attack. We can leverage the cyber-attack capability of DETER, while integrating the real power equipment that INL operates. VPST provides the crucial third testbed that can simulate the SCADA network at full-scale with the added benefit of causing no harm to the actual power grid.

The main requirements for such an experiment are: secure connectivity to guarantee that attacks are contained to the network under duress; reproducibility to allow repeated attacks against various defenses; and fidelity to ensure the system under attack behaves reliably. Additionally, this use case actualizes an attack, so it requires the ability to abstract attacks in order to prevent WAN links from saturating (e.g., during a ping flood).

CHAPTER 10

INTER-TESTBED CONNECTION REQUIREMENTS

10.1 Secure Connectivity

Since the integrated testbed is targeted for SCADA network security analysis, security of the testbed itself is an absolute requirement. The testbed may face threats from external cyber-attacks as well as internal malicious code, which could intentionally or accidentally gain unauthorized access to secret assets. A good security policy should enable several layers of protection including transmission security, authentication and access control, traffic isolation, intrusion detection, logging and reporting. However, existing security technologies are generally not implemented in real SCADA systems because real SCADA devices usually have limited processing capabilities, operate in real time, and are typically not designed with cyber-security in mind.

To ensure a secure cross-testbed experimental environment, access control policies and strong authentication should be enabled at all access points; the access model should deny access to any party not explicitly allowed; and only ports and services required for operations should be allowed. Open PCS Security Architecture for Interoperable Design (OPSAID)[40] is a framework for accelerating development and adoption of a wide range of security functionality in control systems using IP networks. Its technical approach is to use existing open-source technologies whenever possible. This approach could be augmented with additional layers of protection to cover the full scope of secure connectivity.

We have analyzed the current testbed situation and have implemented various measures to ensure the secure connectivity required by VPST. First, we have segregated the virtual nodes onto a private network. This prohibits incidental traffic emanating from our network and interfering with the outside world unless specifically redirected to do so. Also, communication between all components utilizes proven secure protocols. The local systems all communicate using OpenVPN (using *SSL* for encryption). The remote testbeds connect using IPSec, which provides basic security guarantees.

10.2 Performance

For a testbed to be useful, it must provide timely results. In order for this to occur, certain characteristics must hold true for both the inter-testbed connections (ITC) as well as the performance of a single testbed. When connecting two or more testbeds, precautions must be taken in order to provide guarantees about performance, primarily concerning the latency of information transfer between testbeds. To allow for efficient scaling, our ITCs handle multiple connections by having a single point of contact and then distributing the workload to other components. Instead of having an individual connection for each emulated host, we aggregate the data and use a tagging mechanism to differentiate between the hosts using "super nodes" as discussed in Section 11.3.2.

Connecting with a simulated environment has its own set of issues, namely the interaction between emulation and simulation. The communication simulator in VPST has the capability to absorb latency caused by emulated packets. Since emulated packets take precedence over simulated packets, real communication will always remain as close to real-time as possible. *Look-ahead* is the ability to predict the amount of simulated time that could be safely advanced in one process without causing errors in other concurrent processes. Where possible, VPST-C utilizes look-ahead algorithms in order to keep the simulation running as smoothly as possible.

The other source of latency is in the simulation control plane. The main source of information provided over this link ought to be known ahead of time and therefore can be transmitted prior to the start of simulation. By shifting the bulk of control messages outside the simulation itself, we can minimize the overhead incurred by control messages.

Another performance concern is that of scalability. One of VPST's contributions to the SCADA testbed community is that we provide a highly scalable network simulator. Therefore, we must ensure that the scale of networks we can simulate is sufficient to justify the interconnection of other testbeds with VPST. VPST is capable of simulating over a million devices and the electrical simulation can handle more than one hundred thousand buses [41]. The size of a power grid that can be simulated with this much capacity may be appreciated by comparison: a city the size of Madison, WI (about a quarter of a million people) has a grid with a couple hundred buses. In order to simulate a network of this scope in real-time, we make use of the the Trusted ILLIAC [42] as discussed in Section 11.2.

10.3 Resource Allocation

Flexible configuration has been addressed for standalone simulation/emulation testbeds in [19] and [43]. Further, an integrated testbed requires an accurate resource mapping among testbeds for balancing customizability and speed. VPST takes the decentralized approach, where interfaces to other testbeds are decomposed into modules for the ease of customization. Details are discussed in Section 11.3.

VPST intelligently partitions simulation models for balancing resources and minimizing communication overhead across multiple machines. Similarly, a good mapping may minimize the number of links across heterogeneous testbeds, though it is often hard to determine if a mapping is overloaded at the initial stage of an experiment, especially when human decisions are later involved. Therefore, techniques for overloaded link detection and dynamic resource mapping optimization are desired. Successive mappings are adjusted based on the feedback from the detection system and the prior mapping, until no overload is detected, or until all physical resources are depleted.

10.4 Reproducibility

The dynamics of the real SCADA network cover a wide range of conditions including, but not limited to, the size of the network, type of underlying physical medium, available bandwidth and time-varying traffic patterns. Therefore, precisely repeating the experimental conditions and reproducing entire or partial results is a property of a good testbed. Integrating with a simulation testbed enhances reproducibility, since the entire parameter space including both input and environment configuration can be fully controlled.

Reproducibility in the scope of inter-connected testbeds requires conducting experiments in a controlled and interactive manner, especially allowing human-in-the-loop decision, as discussed in Section 9.1.1. Experimenters are given the opportunity to tune certain model parameters online, such as link connectivity and event response mechanisms. VPST then progresses along a new experimental trajectory, which is recorded as tcpdump/libpcap traces for analysis and later reproduction.

Lastly, the testbed must be able to handle the unpredictability of long-distance communication. For instance, if VSPT receives traffic with remote origination, the simulation must be identical from one run to the next, regardless of inter-site latency. VSPT utilizes algorithms that deal with this on a local level, but it also must be addressed in the context of inter-operating testbeds.

10.5 Fidelity

To provide high fidelity, VPST-C must be as transparent as possible. That is, real-world equipment should not be able to tell that it is communicating with a virtual host. In order to present this appearance to a real control station, for instance, issues such as latency, realistic data patterns, and accurate virtual hosts are all important aspects of VPST. Latency has been discussed in Section 10.2, but realistic data and accurate hosts both fall under the auspice of fidelity requirements.

Realistic data patterns are created through the interactions of each of the layers modeled in VPST-C. Virtual hosts are responsible for creation of such data. These virtual hosts can be as abstract as a "router" with a simple MAC layer or as detailed as an "SEL 421 Relay" with a complete DNP3/MODBUS stack and high resolution Ethernet.

Fidelity is often a counterpoint to performance in that the more accurately a host is modeled, the more computation is required. As such, the trade-off needs to be considered individually for each project that requires interaction with VPST.

CHAPTER 11

VPST ARCHITECTURE

11.1 Base System Overview

As introduced in [2], VPST is divided into three main subsystems. Figure 11.2 shows the following subsystems and their interconnections:

VPST-E handles electrical simulation. The primary component here is PowerWorld which is capable of simulating large scale electrical networks at the bus level. We use this to model city-sized or larger power grids. Using a conversion module, PowerWorld can supply real-time input to our physical devices in VPST-R-local. Using the same conversion tool, emulation and the State Server in RINSE, we can provide data to VPST-C.

VPST-C handles network simulation based on RINSE [43], which provides a highly scalable virtual network that is used to model the cyber domain of the electrical grid. As discussed in previous sections, we have developed virtual SCADA-specific models which are able to communicate among themselves and real devices using DNP3 through emulation support. VPST-C can also interact with VPST-R-local through emulation, supplying a realistic SCADA environment to trusted third parties.

VPST-R-local represents all the real devices. Any software that is run rather than simulated resides on some real device in the VPST-R-local, and is represented inside of VPST-C by a device proxy. Devices in VPST-R-local are capable of interacting with VPST-E through a converter, and VPST-C through its emulation capability.

11.2 Enhanced VPST

As seen in Figure 11.2, VPST-R-remote represents a remote test bed communicating with VPST through an ITC installed at the remote site. We have enhanced VPST with the following significant augmentations:

ITC: In order to connect with a remote testbed, we extend the basic VPST architecture by use of ITCs, discussed further in 11.3. Figures 11.2 and 11.3 both show one remote testbed, but this can be scaled to handle more than one remote connection. The ITC is responsible for ensuring that two testbeds are able to efficiently and expressively communicate with each other so as to facilitate their interaction.

Trusted ILLIAC: Another enhancement to the VPST architecture we have introduced is the use of high performance computing clusters, such as the Trusted ILLIAC [42] and its 512 cores, to scale even further. VPST supports model partitioning to intelligently spread virtual nodes across the allocated processors so as to minimize inter-processor communication. This allows an almost-linear scaling of performance per core. The Local Controller is responsible for interpreting configuration files in order to partition the graph and allocate the Trusted ILLIAC resources.

Trace Files: We have also modified VPST to work with trace files. VPST now has the capability of dumping the traffic into a *tcpdump* file that can be replayed at a later time. First, this allows an external entity (e.g. a power company) to use our testbed to examine their network traces for whatever purpose they may have (e.g. testing a prevention policy). Secondly, this allows replaying experiments in order to examine how real-world equipment interacts under different configurations.

11.3 Inter-Testbed Connector Framework

Figure 11.3 shows the ITC architecture. The control plane and the data plane are separated because the control channel is often more sensitive to latency and also requires less bandwidth than the data channel. In addition, control signals likely require higher security level than traffic data does. When two testbeds are initially connecting to each other, the ITC in VPST-C is always identified as the master controller, which is responsible for making a high-level resource allocation plan based on the requests initiated from slave ITCs in other testbeds. Only the local ITC is allowed to communicate with the local controller so as not to violate the local security policy. The architecture design is decomposed into individual modules to allow customization and facilitate the extension of functionality.

11.3.1 Simulation Control Plane

ITC Controller, the hub of an ITC, exchanges control commands with a remote ITC and collects/distributes these control commands with the local control plane. *Secure connectivity* is ensured here through access control policies and authentication mechanisms.

Resource Allocator is responsible for managing *resource allocation* based on requests from remote testbeds and responses from local resources. Load balancing should be optimized here to improve performance and fidelity. The Resource Allocator is also responsible for checking the correctness of the topology mapping and detecting IP address conflicts across multiple testbeds. Since VPST-C is the central component of the system and provides the interconnection, IP uniqueness should be guaranteed by VPST-C. IP conflicts in the simulator are automatically resolved through its IP reassignment procedure in the case of simulation, and the resource allocator performs IP translation when real devices are used.

Resource Configurator is responsible for configuring network components such as hosts, links and traffic. Using the Domain Modeling Language (DML) in VPST-C makes the configuration process both flexible and *reproducible*. DML is a simple scripting language with a hierarchical attribute tree notation. Each node is syntactically nested with all its attributes and its child nodes to ease configuration for large-scale network experiments and, by accurately modeling the network stack, to support *high-fidelity* distributed experimentation.

Run-time Controller is responsible for controlling live experiments. This could include tasks like starting a DoS attack or altering the data polling pattern based on observed states at the remote site. However, overuse of dynamic adjustment may increase communication overhead, lowering *performance*. Therefore, a good practice may be to shift the majority of cross-testbed communication to initialization and cleanup stages.

Error Detector is responsible for detecting abnormalities that occur during a live experiment. These could include errors such as host failure, loss of synchronization, or known warnings due to certain experimental parameters/intermediate outputs exceeding a preset threshold. To ensure *fidelity*, the system will then take corresponding actions such as relocating extra hosts, generating local or cross-testbed alerts, writing events to local system logs, or terminating/restarting experiments. The error detector can either be triggered by an abnormal event or perform periodic checks to ensure a healthy experimental environment.

Data Plane Configurator is responsible for issuing controls to the data plane at the initialization, run-time and cleanup stages of an experiment. Controls include setting the distribution of incoming traffic, how to aggregate outgoing traffic, and specifying the types of reports to collect upon completion of an experiment.

11.3.2 Model Data Plane

Traffic Distributor is responsible for bridging traffic data across two testbeds based on the settings from the data plane configurator. The number of cross-testbed physical links are minimized for *performance* gain. Therefore, traffic data is often aggregated and the traffic distributor is used to forward traffic to the right destination. VPST-C has one type of node called a "super node", which handles real traffic from the emulation channel. It appends a new virtual IP header based on an IP translation table established at the initialization stage by the resource allocator, and then distributes packets as if the super node is a traffic generator. Upon leaving VPST-C, the virtual IP header is extracted so that a remote real testbed can correctly handle the packet.

Measurement Reporter is responsible for collecting statistics of the experiment ranging from a single value like average packet loss rate to detailed per-host status reports. Upon completion of an experiment, reports are sent to the remote testbeds as instructed by the control plane. This module is included in the data plane because in some cases, remote experimenters may need the entire trace files, which require large bandwidth. Meanwhile, the security level of the data channel must be raised accordingly if the reports contain sensitive data.

11.4 Figures

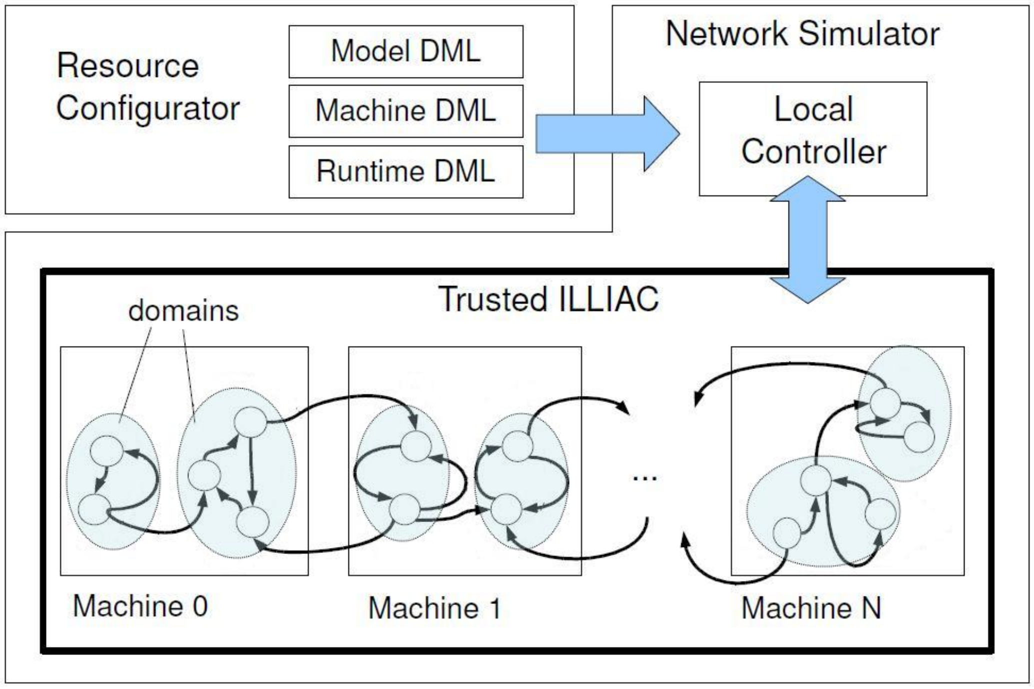


Figure 11.1: Trusted ILLIAC

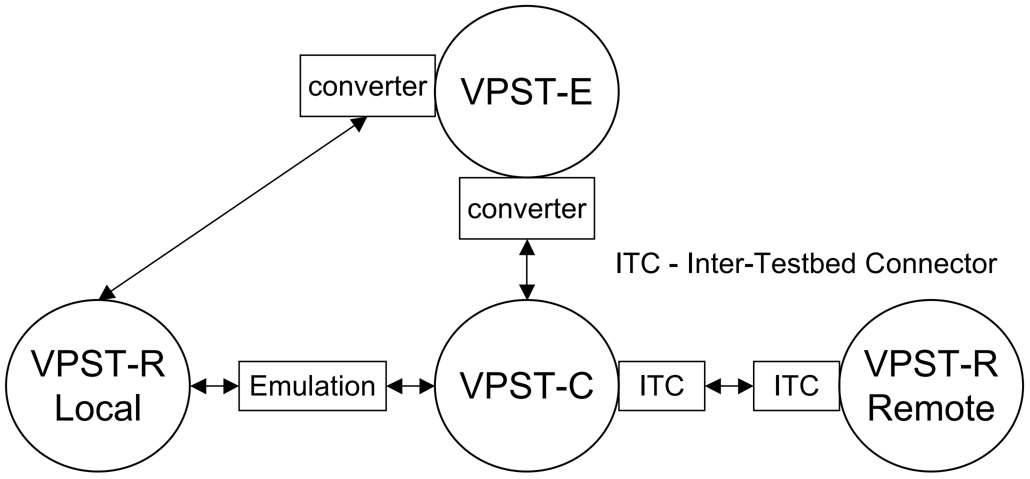


Figure 11.2: VPST System Diagram

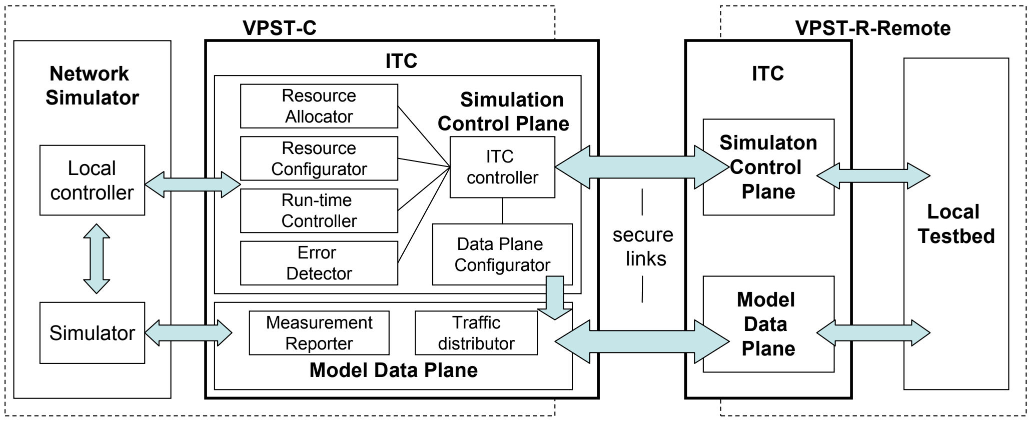


Figure 11.3: ITC Architecture Diagram

CHAPTER 12

RELATED WORK

The following are SCADA testbeds that we are aware of. SCADA research is incredibly important as the actions taken by and against the system can affect the lives of millions. It is always a good a idea to have many eyes looking over the same specifications and implementations to ensure that requirements are met regarding safety, security, and reliable energy. That being said, we feel that VPST fills a special need that other test beds do not.

DETER [19] is an Emulab-based security testbed with a shared infrastructure of several hundred experimental nodes. Many attacking/malware models and tools for traffic/topology generation and analysis are available through DETER. These resources must be leveraged in order to faithfully test the security of SCADA networks. DETER also supports remote access while providing assurance for isolation and containment of each experiment. However, everything in DETER is real, from the operating system to the network stack, which makes replicating a full-scale SCADA network infeasible.

National SCADA Test Bed Program (NSTB) [44] was jointly established at INL and Sandia National Laboratory (SNL). NSTB consists of 61 miles of cables, 7 outstations, and more than 300 monitoring points across the nation. Many equipment manufacturers and government agencies conduct tests on NSTB for finding tangible solutions to growing threats to the power grid. However, NSTB is not publicly available. Even if it were, it is still insufficient to explore the impact of various security technologies on a nationwide power grid. In addition, it lacks the flexibility to explore different architectures, since it is primarily a physical system.

Virtual Control System Environment Project (VCSE) [45] in SNL was designed to incorporate their existing tools, including simulated, emulated, and physical components to assess security vulnerabilities in SCADA systems. The main difference from VPST is that its OPNET-based simulation framework does not scale nearly as well as RINSE.

The VIKING Project [46] is a project sponsored by the European Communitys Seventh Framework Program. Their research is similar to the work being done here, as they are concentrating on a combined simulation/emulation framework with SCADA-specific models. However, their focus appears to be on analzying attacks – specifically, *denial of service*, *integrity*, and *phishing* attacks [47] and not appraising new technologies. Their use of Simulink by MathWorks [48] provides a much higher degree of fidelity, going so far as to simulate OS effects. This will, of course, provide greater fidelity, but it will not sufficiently scale to investigate networks that approach the size of an actual power control network.

CHAPTER 13

CONCLUDING REMARKS

13.1 Conclusion

In this thesis, we have discussed the development of virtual models for DNP3, data aggregators, and relays within the RINSE framework. These models are important as they provide a realistic environment in which experimentation on emerging power technologies can take place. The highly touted smart grid and its related technology are being integrated into the existing grid at an alarming rate. We must understand how they affect existing technologies and our existing infrastructure. By utilizing the models discussed above, we provide such an environment to test these technologies. In order to provide a realistic simulation, the virtual relays acquire data from PowerWorld, a power flow simulator, through a virtual host termed the State Server. This State Server captures snapshots of the electrical simulation and provides them to the virtual relays through shared memory. By enabling real time information to be passed through our SCADA simulation, RINSE provides a platform for testing new technologies in a scalable, high-fidelity manner. In Chapter 8, we go through a sample workflow where we can see the steps required to verify the functionality of a new technology.

In the second portion of the thesis, we have also shown that there is potential in connecting VPST with remote testbeds, in order to take advantage of their unique offerings. We have discussed many of the concerns regarding integrating multiple testbeds as well as the methods that we employ to alleviate said concerns. We have also presented other testbeds and concluded that VPST fills an important niche that other testbeds cannot.

13.2 Future Work

One area that needs to be looked into is *report-by-exception* polling. This can come in two flavors. The first is a *polled report-by-exception* which is a poll that only asks for data which has changed. This should be fairly easy to implement. Another form is *unsolicited report-by-exception.* This allows a relay to send a message if something has changed state, without being asked to do so. This type of reporting is also currently unimplemented, but would require greater effort, due to the nature in which the state server is implemented. Without having access to the actual grid, but relying on the advice of experts, it seems as though relays support *report-by-exception*, but often do not use it as resources may be better spent on real-time monitoring. In any case, this seems like a natural feature to implement.

A nontrivial amount of work had been put into formally verifying a precursor to the architecture described in Figure 6.1 using the Symbolic Model Verifier (SMV) [49]. However, it was put aside in order to focus development on the model itself. More work must be done to verify the framework, and formal verification could provide a useful and interesting way to do that.

The DNP Users Group has published test procedures for Levels 1 and 2, the simplest implementations. Future work ought to be done to ensure that the virtual models developed in RINSE pass these tests, or at least that sub-portions of these tests are deemed relevant to the demands of the experiment. It may be wise to introduce models that conform to varying levels of these tests, so as to provide certain guarantees regarding both fidelity and performance.

One area of concern mentioned by people who reviewed this work is that of clock skew among the relays. As implemented, the state server receives values in chunks. All data within a given chunk has occured during the same time slice. However, in the real grid, data from separate relays are never guanranteed to occur during the same time slice. This is the type of problem that synchrophasor techonology [50] is attempting to fix. However, the question of how, or even whether, to simulate this effect will require some examination. Again, this raises the question regarding fidelity versus performance again. In order to simulate this clock skew, the state server must query each data point separately and provide randomization in the timing as well. This overhead may impact the performance of the simulation, as well as the freshness of the data.

One of the next steps we would like to take is to develop a black-box implementation of the current ITC. This box would be installed in a remote testbed and would be responsible for seamlessly connecting a remote testbed to ours with as little manual configuration as possible. In addition to this, we believe there is benefit in extracting as much efficiency out of the WAN transmissions as possible (e.g., compression of data and intelligent use of control messages to reduce the amount of traffic that must pass over the link). For instance, a real flooding attack would not be tolerated over a WAN link since this will likely be the bottleneck.

REFERENCES

[1] UIUC, "Trustworthy Cyber Infrastructure for the Power Grid," 2010. [Online]. Available: http://tcip.iti.illinois.edu

[2] D. M. Nicol, C. M. Davis, and T. Overbye, "A Virtual Power System Testbed for Cyber-Security Decision Support," in *Proceedings of the 2009 INFORMS Simulation Society Workshop on Simulation: At the Interface of Modeling and Anaylsis*, 2009, pp. 62–66.

[3] D. Bergman, D. Jin, D. Nicol, and T. Yardley, "The Virtual Power System Testbed and Inter-Testbed Integration," in *Proceedings of the Conference on Cyber Security Experimentation and Test (CSET), USENIX Association*, 2009, pp. 1–6.

[4] Pacific Northwest National Laboratory (PNNL), "Looking Back at the August 2003 Blackout," Accessed February 2010. [Online]. Available: http://eioc.pnl.gov/research/2003blackout.stm

[5] G. Andersson et al., "Causes of the 2003 major grid blackouts in North America and Europe, and recommended means to improve system dynamic performance," *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 1922–1928, 2005.

[6] S. Corsi and C. Sabelli, "General Blackout in Italy Sunday September 28, 2003, h. 03: 28: 00," in *IEEE Power Engineering Society General Meeting, 2004*, 2004, pp. 1691–1702.

[7] E. Bompard, C. Gao, R. Napoli, A. Russo, M. Masera, and A. Stefanini, "Risk assessment of malicious attacks against power systems," *Trans. Sys. Man Cyber. Part A*, vol. 39, no. 5, pp. 1074–1085, 2009.

[8] J. D. Fernandez and A. E. Fernandez, "SCADA systems: Vulnerabilities and remediation," *J. Comput. Small Coll.*, vol. 20, no. 4, pp. 160–168, 2005.

[9] V. Igure, S. Laughter, and R. Williams, "Security issues in SCADA networks," *Computers & Security*, vol. 25, no. 7, pp. 498–506, 2006.

[10] J. Salmeron, K. Wood, R. Baldick et al., "Analysis of electric grid security under terrorist threat," *IEEE Transactions on Power Systems*, vol. 19, no. 2, pp. 905–912, 2004.

[11] Department of Energy Smart Grid Task Force, 2007. [Online]. Available: http://www.oe.energy.gov/smartgrid_taskforce.htm

[12] US National Institute of Standards and Technology (NIST), "Smart Grid Interoperability Standards Project," 2010. [Online]. Available: http://www.nist.gov/smartgrid/

[13] US National Institute of Standards and Technology (NIST), "Smart Grid Cybersecurity Strategy and Requirements," 2009. [Online]. Available: http://csrc.nist.gov/publications/drafts/nistir-7628draft-nistir-7628.pdf

[14] P. Tsang and S. Smith, "YASIR: A Low-Latency, High-Integrity Security Retrofit for Legacy SCADA Systems," in *Proceedings of The IFIP TC 11 23rd International Information Security Conference*, 2008, pp. 445–459.

[15] L. Piètre-Cambacédès and P. Sitbon, "Cryptographic Key Management for SCADA Systems-Issues and Perspectives," in *ISA '08: Proceedings of the 2008 International Conference on Information Security and Assurance (isa 2008)*, 2008, pp. 156–161.

[16] C. Bowen III, T. Buennemeyer, and R. Thomas, "A Plan for SCADA Security Employing Best Practices and Client Puzzles to Deter DoS Attacks," presented at Working Together: R&D Partnerships in Homeland Security, 2005.

[17] DNP Users Group, "DNP3 Specification, Secure Authentication, Supplement to Volume 2," March 2010. [Online]. Available: http://www.dnp.org/Modules/Library/Document.aspx

[18] M. Majdalawieh, F. Parisi-Presicce, and D. Wijesekera, "DNPSec: Distributed Network Protocol Version 3 (DNP3) Security Framework," *Advances in Computer, Information, and Systems Sciences, and Engineering: Proceedings of IETA 2005, TeNe 2005, and EIAE 2005*, pp. 227–234, 2006.

[19] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab, "Experience with DETER: A testbed for security research," in *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on*, 2006, p. 10.

[20] SSF, "Scalable simulation framework," 2004. [Online]. Available: http://www.ssfnet.org/

[21] D. Nicol, M. Goldsby, and M. Johnson, "Fluid-based simulation of communication networks using SSF," in *Proceedings of the 1999 European Simulation Symposium*, vol. 2, 1999.

[22] H. Liu, "SSFNET TCP Simulation Analysis by tcpanaly," *Relation*, vol. 10, no. 1.27, p. 8844, 2000.

[23] V. Paxson, "Measurements and Analysis of End-to-End Internet Dynamics," Ph.D. dissertation, University of California at Berkeley, Berkeley, CA, 1998.

[24] S. Mohagheghi, J. Stoupis, and Z. Wang, "Communication protocols and networks for power systems-current status and future trends," in *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, March 2009, pp. 1–9.

[25] DNP Users Group, "DNP: Distributed Network Protocol," 2010. [Online]. Available: http://www.dnp.org

[26] DNP Users Group, "DNP3 Specification Volume 4: Data Link Layer, DNP User's Group," December 2002. [Online]. Available: http://www.dnp.org/Modules/Library/Document.aspx

[27] DNP Users Group, "DNP3 Specification Volume 3: Transport Function, DNP User's Group," November 2002. [Online]. Available: http://www.dnp.org/Modules/Library/Document.aspx

[28] DNP Users Group, "DNP3 Specification Volume 2: Application Layer, DNP User's Group," October 2005. [Online]. Available: http://www.dnp.org/Modules/Library/Document.aspx

[29] DNP Users Group, "Guide to Calculate DNP CRC," December 2002. [Online]. Available: http://www.dnp.org/Modules/Library/Document.aspx

[30] S. East, J. Butts, M. Papa, and S. Shenoi, "A Taxonomy of Attacks on the DNP3 Protocol," in *Critical Infrastructure Protection III, IFIP Advances in Information and Communication Technology, Volume 311*, 2009, pp. 67–81.

[31] P. Ralston, J. Graham, and J. Hieb, "Cyber security risk assessment for SCADA and DCS networks," *ISA Transactions*, vol. 46, no. 4, pp. 583–594, 2007.

[32] S. Patel and Y. Yu, "Analysis of SCADA Security models," *International Management Review*, vol. 3, no. 2, 2007.

[33] A. Faruk, "Testing & Exploring Vulnerabilities of the Applications Implementing DNP3 Protocol," M.S. thesis, Kungliga Tekniska högskolan, 2008.

[34] S. Hong and S. Lee, "Challenges and pespectives in security measures for the SCADA system," in *Proc. 5th Myongji-Tsinghua University Joint Seminar on Prototection & Automation*, 2008.

[35] T. Mander et al., "Power system DNP3 data object security using data sets", Comput. Secur., pp. 1–14, 2009. doi:10.1016/j.cose.2009.10.001. [Online]. Available: http://www.sciencedirect.com/science/article/B6V8G-4XFGJ76-1/2/3992800656e1bb33ad74bfe2bff1c724.

[36] D. Rrushi and U. di Milano, "SCADA Intrusion Prevention System," in *Proceedings of 1st CI2RCO Critical Information Infrastructure Protection Conference*, 2006. [Online]. Available: http://perso.telecom-paristech.fr/ legrand/CI2RCO-conf/Article/scada_rrushi.pdf

[37] J. Graham and S. Patel, "Security Considerations in SCADA Communication Protocols," Intelligent Systems Research Laboratory, Tech. Rep. TR-ISRL-04-01, September 2004.

[38] "DNP v3.0 Guide: A Protocol Primer," 2008. [Online]. Available: http://www.scribd.com/doc/2542661/DNP-v3-0-Guide-A-Protocol-Primer

[39] "OpenView: Graphical User Interface," 2009. [Online]. Available: http://www.osii.com/pdf/scada-ui/OpenView_PS.pdf

[40] S. A. Hurd, J. E. Stamp, and A. R. Chavez, "Department of Energy Office of Electric Delivery and Reliability's National SCADA Testbed Program," *OPSAID Initial Design and Testing Report*, November 2007.

[41] PowerWorld Corporation, "PowerWorld Simulator," 2010. [Online]. Available: http://www.powerworld.com/

[42] W. Hwu, W. Sanders, R. Iyer, and K. Nahrstedt, "Trusted ILLIAC: A Configurable, Application-Aware, High-Performance Platform for Trustworthy Computing," June 2006. [Online]. Available: http://www.iti.illinois.edu/sites/www.iti.illinois.edu/files/docs/cri-snowbird-06-talk-final.pdf

[43] M. Liljenstam, J. Liu, D. Nicol, Y. Yuan, G. Yan, and C. Grier, "RINSE: The Real-Time Immersive Network Simulation Environment for Network Security Exercises," in *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation.* Washington, DC, USA: IEEE Computer Society, 2005, pp. 119–128.

[44] Idaho National Laboratory, "National SCADA Test Bed Program," 2009. [Online]. Available: http://www.inl.gov/scada/publications/index.shtml

[45] M. J. McDonald, G. N. Conrad, T. C. Service, and R. H. Cassidy, "Cyber Effects Analysis Using VCSE," Sandia National Laboratories, Tech. Rep. SAND2008-5954, September 2008.

[46] A. Giani, S. Sastry, K. Johansson, and H. Sandberg, "The VIKING Project: An Initiative on Resilient Control of Power Networks," in *Proceedings of ISRCS 2009: 2nd International Symposium on Resilient Control Systems*, 2009, pp. 31–35.

[47] A. Giani, G. Karsai, T. Roosta, A. Shah, B. Sinopoli, and J. Wiley, "A Testbed For Secure and Robust SCADA systems," *SIGBED Rev.*, vol. 5, no. 2, pp. 1–4, 2008.

[48] MathWorks, "Simulink," 2010. [Online]. Available: http://www.mathworks.com

[49] CMU Model Checking Group, "The Symbolic Model Verifier (SMV)," 2001. [Online]. Available: http://www-2.cs.cmu.edu/ modelcheck/smv.html

[50] E. Schweitzer and D. Whitehead, "Real-Time Power System Control Using Synchrophasors." Presented at 34th Annual Western Protective Relay Conference, Oct. 2007.