# TOWARDS DISTINGUISHING BETWEEN CYBER-ATTACKS AND FAULTS IN CYBER-PHYSICAL SYSTEMS

By

Aaron William Werth

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

May, 2014

Nashville, Tennessee

Approved:

Gabor Karsai, Ph.D.

Gautam Biswas, Ph.D.

To my parents
and
my grandparents

ACKNOWLEDGEMENT

This work was funded by the NSF TRUST (Team for Research in Ubiquitous Secure Technology) Project. This project is supported by the National Science Foundation (NSF award number CCF-0424422).

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

# CHAPTER I

# INTRODUCTION

In recent years, there has been increased interest in a new area of research: the security of Supervisory Control and Data Acquisition (SCADA) systems. SCADA systems are elaborate computer systems which have two main functions as their name implies: (1) they manage and control critical infrastructure, and (2) they also collect data from critical infrastructure for analysis and potential response. Examples of these infrastructures include the power grid, chemical plants, and oil refineries [9]. Modern SCADA systems have new vulnerabilities. The major issue today is that malicious software, or malware, may be able to access these systems via the Internet since SCADA systems are now more connected and open to outside networks. Therefore, exposure to malware can affect the operation of SCADA systems [3], [4]. This exposure is especially problematic because a SCADA system, a type of cyber-physical system (CPS), manages critical infrastructure and affects the physical world. A CPS is defined as a system in which there is close interaction between computing devices and the physical world [30]. Because of this close interaction, malware could alter the behavior of these systems such that (1) they could cause the operation of the plant to run inefficiently meaning increased operating costs, or (2) in a worst case scenario, they endanger public safety [1].

If a cyber-attack occurs on a SCADA system, it is possible that the behavior of the system can be modified to deviate from normal operation. Normal operation would be defined as the way that the SCADA system was designed to operate in order to meet certain objectives. These objectives include safety and also the production of goods or services. It is especially problematic for the SCADA System if safety is compromised by a cyber-attack. Because SCADA systems manage critical infrastructure, these attacks can have a large physical impact in terms of the destruction that they can cause [1]. For example, pressure in a tank could build up to dangerous levels if the SCADA system is hacked by an intruder. This increase in pressure, if it is substantial, may lead to an explosion. Also from an economic standpoint, it is important to understand that this critical infrastructure can take the form of industrial plants, which means that revenue could be at stake if the SCADA system is tampered with. This loss of revenue could be due to the industrial processes becoming more inefficient. An example of this may be a chemical

plant. A certain chemical reaction may have to take place in the chemical processes of this plant to produce a certain product. Therefore, the right combinations of inputs with a certain ratio must be applied to have the optimal yield of the product. Therefore, it can be seen from these examples that economic consequences or issues with safety may occur if SCADA systems that manage these processes are compromised [1], [2].

The reason that research in the security of SCADA systems has become important is that recent incidents have in fact occurred in which SCADA systems were compromised. Because these types of incidents have a great impact on society, they have been brought to the attention of governments and research communities. Some well-known examples that have been studied are Stuxnet and the Slammer worm [41]. The Stuxnet worm was an elaborate piece of malware that took advantage of vulnerabilities of the Windows operating system in order to damage centrifuges in a nuclear facility in Iran. This led to economic loss since these centrifuges had to be replaced [31]. The Slammer worm attacked and disabled the network of Ohio's Besse nuclear power plant. As a result, the monitoring system used to ensure safety of the system was unable to function, even though there may have been a firewall to protect the system. In this particular case, this attack did not cause loss of life or other harm besides economic loss, but it did highlight the need for increased security for these systems [8].

## Developments in SCADA Security

To counter this threat against SCADA systems, certain measures have been studied and created over the course of time. Previously, a great body of literature has been developed that deals with computer and network security in general [2]. Some of the methods in network security have been applied to SCADA systems in recent work. For instance, one focus of recent research has been in prevention or access control by some of the standard techniques used in network and computer security. Access control simply means that the computer network is able to prevent certain access that is unwanted and allow other access that would be considered legitimate. This can be achieved through whitelisting and blacklisting. Whitelisting involves comparing applications that attempt to access a resource with a list of approved sources. Blacklisting, on the other hand, involves rejecting access to applications that are considered malicious. [7] This type of prevention is typically done using a firewall. Firewalls have been found to be useful for protecting the network associated with the SCADA system in addition to

protecting general computer networks [2]. However, protection may not always be as effective as desired. It is possible that malicious software can still bypass this form of security using zero-day attacks as in the case of Stuxnet. Zero-day attacks are attacks which take advantage of vulnerabilities of hardware and software that have not been discovered by the users of the system. These attacks have not been officially discovered by the vendors of this hardware or software associated with the SCADA system [2]. Since these attacks are unknown to the users of the SCADA system, these systems may not be truly secure, meaning a malicious entity can tamper with the system in potentially destructive ways. It is also important to be able to detect if an attack is occurring in the system in addition to protecting the system against intrusions [2]. Methods of detection have been used in traditional IT systems, such as intrusion detection systems (IDS) [28]. These methods are fairly effective. However, even with existing IDS technology, it should be noted that SCADA system security has a major difference from general computer security in that the physical system is affected in addition to the computer systems and networks. As a result of the new awareness, researchers have been increasingly studying the effects of cyber-attacks on physical systems and how these attacks can be detected for practical purposes. For example, the plant operator would want to be alerted that an attack has occurred and he would want to respond to it appropriately to prevent or reduce any damage or loss, whether it is physical or economic [1], [2]. Response in addition to detection was one of the objectives in the paper by Cardinas et al [2]. Cardinas discussed an approach of observing the physical system and then determining whether there was abnormal behavior. If the plant was found to have abnormal behavior, the supervisory system would attempt to place the plant in a safe state. The system considered was the Tennessee Eastman Process Control System (TEP-CS), which is a widely used system in the literature to study SCADA systems. The reason that it was studied is that it allows for situations where there could be economic loss or violations of safety. As in this current work, it is not the desire of that work to study how specific vulnerabilities in software are exploited. What is desired is to use knowledge of the physical system to aid in detecting attacks. Observation of the physical system for abnormalities is of utmost importance since the physical system is where the potential for danger is the greatest.

Abnormalities in the physical system are typically detected by means of a comparison between an ideal system and the system under observation. This can be done using a fault

diagnosis method that incorporates model-based diagnosis, which is one of several fault diagnosis techniques that may be used [2], [3]. Model-based diagnosis compares the mathematical model of the ideal system with sensor data (data from the physical system) received by a supervisory system. The model of the nominal system is derived by first principles or empirical methods [13]. The comparison is useful to detect an anomaly. In fact, this work uses a basic form of model-based diagnosis to determine if there is an anomaly in the behavior of the physical system. But it should be noted that, while both of these papers [2], [3] focus on detection of attacks on the physical system, there appears to be somewhat of a weakness with their approaches for detecting attacks. To understand the weakness more clearly, it is helpful to realize that these methods of detection for attacks, by their very nature, would also be suitable for fault detection. This means that the attacks that can be observed in the physical system may be considered indistinguishable from faults. This would certainly be true with the experiments of these aforementioned papers in how the experiments are setup if faults were also introduced in the simulation. The faults in that case would be treated as attacks. In fact, Amin et al. [3] seem to lump faults and attacks together. In another paper [4], Amin et al. explicitly mentions that there is a difficulty in isolating faults from attacks. This inability to make a distinction between an attack and a fault may prove to be problematic since the system or the plant operator may need to respond differently depending on the situation.

## Contributions

Given more information besides the sensor readings from the physical system, it may be hypothesized that it is possible to distinguish between faults and attacks in certain cases, even if there is not enough information to do so in all cases. Simply observing the physical system alone will probably not allow for this distinction to be made. It should be noted that what is meant by "observing the physical system" is that sensor data and other useful information are being sent to a supervisory system for analysis. The reason that faults and attacks are indistinguishable is that a fault can easily resemble an attack in many cases. For instance, it is quite possible that a fault may have an effect that is similar to false data being sent by a malicious entity as seen from sensor data of the physical system, which is sent to the supervisory system. The ability to distinguish the faults and attacks may be useful so that the SCADA system or the plant operators can respond to the situation in an appropriate way, depending on the nature of the anomaly,

whether it is a fault or attack. What may be something worthwhile from a research perspective is to try to understand network behavior as well in order to aid in understanding the nature of the anomaly. A cyber-attack would most likely have a certain effect on the network. Therefore, it would be reasonable to study the resulting effects on the network in addition to the physical system. Possible effects that could occur are as follows: There could be increased traffic on the network, which is something that is discussed in the literature concerning certain types of attacks. For instance, the flooding of packets in a network is a type of attack that can increase network activity. In an extreme scenario, this could lead to loss of communication, which in turn could lead to possibly catastrophic events if a networked control system is the target of this attack. Therefore, it is important to study the methods for detecting these types of attacks. It may be useful to understand how certain network behavior could influence the behavior of the plant. There has been work done to create simulations that would allow the effects of a DoS (Denial of Service) attack on the physical system behavior to be studied [30]. Two major types of attacks that have been studied which can influence the behavior of the physical system in possibly harmful ways are DoS attacks and Deception attacks [1], [2], [30].

The major contribution of this work is to create a methodology or framework to distinguish faults and attacks despite the apparent similarities in how they manifest themselves in the physical system. Faults and attacks may not have unique enough signatures to allow for them to be distinguished from each other adequately in the physical system alone, i.e. using sensor readings of the physical system. A "signature" is a pattern or set of symptoms that characterizes the fault or attack. Therefore, more information is needed. It is worth considering that attacks may have a certain signature that can be detected on the network as well. On the other hand, physical faults with a similar signature as seen by the sensors may not have such a signature on the network. For faults associated with the network, however, the network may not even send any data. This might be seen in no traffic being detected. Therefore, these scenarios seem to have fairly unique signatures associated with them if data from both the network and the physical system are included. To achieve the ability to distinguish an attack from a fault or to simply increase the accuracy in doing so, more information associated with the SCADA system must be collected than what was done in previous work.

The rest of the thesis is organized as follows: Chapter 2 provides the background and covers important terms, definitions, and concepts in this work. Chapter 3 deals with the experiments. It describes the problem formulation and implementation of the experiments. Chapter 4 covers the results of the experiments and includes a discussion and an analysis. Chapter 5 presents the conclusions that were made based on all of the work completed.

# CHAPTER II

# BACKGROUND

In this work, there are several technologies studied and many concepts used. Among these are (1) SCADA systems, (2) Control Systems, (3) more specifically, Networked Control Systems, (4) Network Security, (5) Faults and Fault Diagnosis techniques, (6) Attacks, (7) Intrusion Detection Systems, and (8) Machine Learning techniques for classification. Some of these were briefly discussed in the introduction. The purpose of this chapter is to explain these important concepts as well as others that are applicable to this work.

## SCADA Systems

SCADA systems are essentially computer systems that manage the control systems used in national infrastructure or industrial plants. In recent years, these systems have been found to be more likely to experience cyber-attacks for several reasons: (i) They are increasingly using technology similar to that of traditional IT systems; (ii) they may be connected to the internet; (iii) and they tend not to use the proprietary protocols as in the past. Instead, they use more common protocols. The main reason these systems are made to use this technology is to be more cost-effective. [3] SCADA systems typically have a main supervisory computer system that is connected through a network to other nodes. In computer networks, a node is a device that can communicate with other nodes on the network [15]. These nodes may be microcontrollers that control certain aspects of the industrial process associated with the SCADA system. These devices for SCADA systems in particular are called Remote Terminal Units (RTU), Intelligent Electronic Device (IED), etc. These microcontrollers may be connected to actuators or sensors [10].

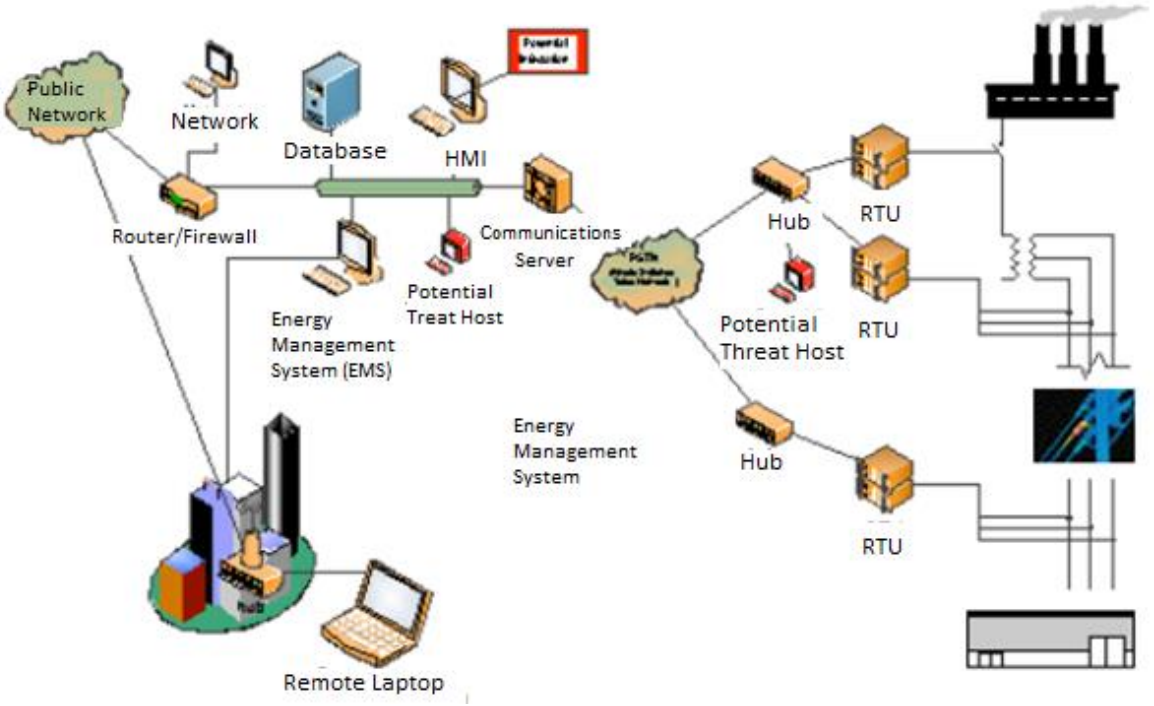


**Figure 2.1: Electric Power SCADA System, which may also be similar to other SCADA systems [40] (the labels, as shown here, were updated by the author for this paper)**

## Control Systems

A SCADA system is an elaborate form of a control system that typically has a hierarchy associated with it and uses lower-level control systems to manage certain aspects of the critical infrastructure. The SCADA system itself is the higher-level supervisory system in the hierarchy and manages the lower-level control systems. Each of these control systems would manage a particular aspect of the industrial process [3], [9]. A control system incorporates a plant, which is the physical system, and a controller, which regulates the plant's behavior. The behavior of the plant can be described in terms of physical quantities associated with the plant that may change over time. The physical quantities associated with the plant that must be adjusted by the controller are called the manipulated variables in the plant and are inputs to the plant. These variables are adjusted so that a certain objective is meant for the plant. This objective may involve these variables being adjusted so that other variables related to them reach a certain value known as the set-point. These other variables are the outputs of the plant, some of which may be referred to as the measured variables. These measured variables may also be made to track a certain function of time instead of a constant set-point. This function is known as the

reference signal [14], [16]. Sensors are the devices directly involved with the plant so that the measured variables can be known. This data, of course, must be sent to the controller. The controller will receive this data as input. Based on the control objective, the controller will send commands to an actuator that is directly involved with the plant. An actuator is a device that manipulates a physical system and receives the commands from the controller. The following figure illustrates a single-input/single-output feedback control system, which is used for this work in the interest of simplicity.

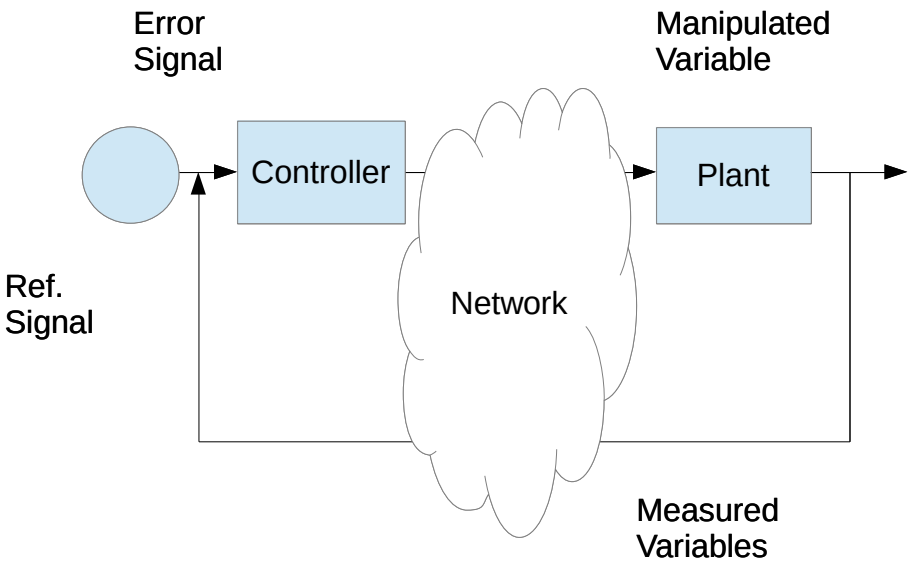


**Figure 2.2: Control System with Blocks that Represent the Controller and the Plant.**

The cloud in the diagram represents a network through which the manipulated variables from the controller are sent to the plant. These manipulated variables are the commands to the actuators. Also, the measured variables (the sensor data) from the plant must pass through this network in order to be received by the controller. This will be discussed in later sections in the chapter. The controller, of course, sends these commands to the plant based on what it receives from the plant in order to meet a certain control objective. The system is designed with a specific control objective to manage the plant or process. One example of a control objective is for the level of water in a tank to maintain a certain set-point. The actuator would be the valve to allow water into the tank. This valve can be opened or closed by the controller. Or the valve can

take on a range of positions that is neither fully closed nor fully opened. The goal of the controller would be to have the water reach that set-point by adjusting the valve.

As was mentioned, a feedback control system is used. This involves measuring the output and adjusting the manipulated variable so as to influence the plant with the goal of causing the measured variable to track with the reference signal.

**Controller**

For this work, a PI controller was used. One of the reasons that it was used is that this type of controller is in other related work [2], [3]. The PI controller implements a specific control law, which is expressed as a transfer function that describes the controller. The input of this transfer function is the error signal. The output of this transfer function is the manipulated variable, which is fed as the input to the plant. The error is defined as the difference between the reference signal and the measured variable, which is the output from the plant. For this work, it is important to express this with discrete time, which is what digital systems use. The controller is assumed to be implemented on a digital microcontroller. The control law can be expressed by the following transfer function, where the input to this function is the error signal:

$$P + IT_s \frac{1}{z-1} \tag{2.1}$$

Where P is the proportional parameter; I is the integral constant; and $T_s$ is the sampling period.
These equations are in the Euler form based on the discrete PI controller block used in MATLAB Simulink [43].
This equation for the transfer function must be implemented in C++ as a simple algorithm. The following pseudo-code shows an algorithm that would implement the above transfer function.

```
TransferFunc(ref_signal, measured_value) //called every 10ms
    {
        error = ref_signal – measured_value;
        integral = integral + error * dt;
                // integral = 0 for initialization
                // dt = sampling period in secs

        output = Kp*error + Ki*integral
    }
```

**Figure 2.3: C style Pseudo-Code to Show a Basic Implementation of the Transfer Function.**

## Plant

The plant is the physical system, and it must be regulated so that it meets the control objective. It can be mechanical, electric, hydraulic, or pneumatic, etc. and its behavior is generally governed by laws of physics, where the relationship of physical quantities associated with the plant can be described in terms of equations. There are many ways that a plant's behavior can be described through mathematics. Creating these mathematical descriptions of the plant is a form of modeling. [34] In the case of this work, a linear discrete-time state space system will be used which has equations for the simulations that are in continuous form. The continuous-time equations are also used for the mode [33]. These are defined as follows:

This is the continuous-time model of the dynamic system:

$$\begin{cases} \dot{\bar{x}} = A\bar{x} + B\bar{u} \\ \bar{y} = C\bar{x} + D\bar{u} \end{cases} \tag{2.2}$$

where $\bar{x}$ is the vector of the state variable, $\bar{y}$ is the vector of the output, $\bar{u}$ is the vector of the inputs.

Discrete-time Model of the Equations:

$$\begin{aligned} \bar{x}[n+1] &= A\bar{x}[k] + B\bar{u}[k] \\ \bar{y}[k] &= C\bar{x}[k] + D\bar{u}[k] \end{aligned} \tag{2.3}$$

**Networked Control Systems**

For some types of SCADA systems, the control system used has a network which facilitates communication between the controller and the plant (This is illustrated in Figure 2.1). These control systems are called networked control systems. The controller and plant are essentially the nodes in the network. The network, of course, has a delay and limited bandwidth associated with it. Delay is defined as the time that it takes for information to travel from one point of the network to another. It should be noted that the delay can vary in its duration. Bandwidth is defined as the amount of information, usually expressed in bits per second (bps) that can travel though the network in a given amount of time. It can be assumed that a networked control system or a SCADA system has a regular traffic pattern in normal operating conditions for the following reasons: (1) the topology remains unchanged. (2) The nodes use the networks in ways that remain constant or have the same basic pattern according to their original design or specification [31]. Deviation from this normal pattern would therefore be indicative of an abnormal event or circumstance. A network facilitates the communication between the nodes. Communication is done through certain protocols. The information is carried by means of packets. A protocol is a specification on how communication will take place so that information can be sent from one node in the network to another. It also refers to the service that allows for this communication according to a certain set of rules associated with the protocol. Two main network protocols are UDP (User Datagram Protocol) and TCP (Transmission Control Protocol). UDP is a connectionless protocol. This means that packets that are dropped on the way to their destination are not resent. In contrast to this, TCP is another protocol which ensures the delivery of the packet through an elaborate method that involves a three-way hand shake for the set-up of the connection. Also, with this particular protocol, the recipient acknowledges to the sender that it has received the data as communication takes place. For this work, UDP is used [15]. The UDP protocol was chosen for two main reasons: (1) It is simpler to implement than the TCP protocol. (2) UDP may be more suitable for real-time systems. A networked control system is a real-time system, in that the system must respond to sensor data in a reasonable amount of time. A slight advantage of UDP over TCP is that UDP will send the newest data from the plant to the controller without much overhead or a complicated process, regardless if some of the datagrams are dropped due to abnormal events on the network. Therefore, the data is more relevant because the newest data arrives at the destination without trying to send somewhat older packets, which

TCP may attempt to do [15]. However, it should be noted that UDP does have some disadvantages, which may cause this protocol to be unreliable in some cases: (1) it can drop messages (2) it does not guarantee in-order delivery of packets. TCP on the other hand will ensure that all data is sent, meaning that if some packets are dropped, TCP will resend them, and it does allow for an in-order delivery. But these data would not be the newest data for the system. There is, however, a time to live associated with TCP. Time to live is defined as the length of time for when transmission or retransmission is allowed to take place. Beyond this period, no retransmission is allowed to occur [15]. Therefore, it is possible to avoid the problem that could occur in TCP where retransmission could occur indefinitely.

**Network Security**

In this work, it is important to have a basic understanding of network security since cyber-attacks are a possible circumstance that an NCS can experience. There are a few concepts that should be understood that are relevant to this work which will be discussed.

Network security has several main components: (1) confidentiality, (2) integrity, and (3) availability [31]. Confidentiality involves hiding information. This component is not dealt with in this work. Integrity basically means that data has not been tampered with or that there is no deceptive data. There are a few types of integrity: Data integrity means that the content of the packets are not modified in any way. Integrity also includes authentication. Authentication means that the process or device that desires a resource is who or what it claims to be. In this work authentication is not dealt with, but it is still important to understand. Data integrity is dealt with in this work. Also, availability involves resources being available for use in a timely manner. All of these aspects of network security must be maintained in order for the network to function properly [21]. In the case of this work, the focus is on attacks on integrity and availability as far as network security is concerned. These two aspects of security are especially important for SCADA systems because unwanted and potentially dangerous behavior can occur on the physical system if these components are violated. Therefore, this work focuses on certain attacks that are meant to attack integrity and availability of the network such that they would influence the behavior of the NCS.

There are also three major goals of security: prevention, detection, and recovery. These were briefly discussed in Chapter 1 of this thesis. The focus of this work is on detection, but it is also important to do more than just detection. It is desired to diagnose the specific problem that the NCS experiences and classify it. Recovery may involve placing the system in a safe state or eliminating the problem altogether.

## Abnormal Situations Experienced by the Networked Control Systems

As was mentioned in Chapter 1, there are two main types of abnormal situations that the NCS can experience: (1) Faults and (2) Attacks. What follows is a discussion for each of them.

### Faults

Faults may occur during the operation of the NCS. Although they may be rare, it is not known how often they can occur as compared with attacks. Therefore, it is reasonable to understand the behavior of faults also. Faults are defined according to Gertler, using general terms, [13] as "deviations from the normal behavior of the plant or its instrumentation." These faults could be the result of damage to equipment [12]. It should be noted that these faults may impact the physical system such that it operates inefficiently or to the point of causing harm to equipment or people. It is also important to understand what kinds of faults can occur. Faults may be mechanical, but they may also involve errors in the network or malfunctions in software. Some of the more standard faults are those in the physical system, which may be either additive faults or multiplicative faults. Additive faults entail unknown inputs being added to the inputs of the plant or unknown biases added to the outputs of the plant. On the other hand, multiplicative faults in the system may involve the system as described by a matrix to have changes. This could be due to changes in the plant parameters due to component faults [13]. Attacks may also exhibit similar behavior to faults and may be detected by standard fault diagnosis techniques. One way to detect a fault/attack is to use a method similar to what is used in the work by Cardinas et al. [3] which focuses on sensors that give incorrect information deviating from their actual readings.

**Cyber-Attacks**

A cyber-attack, or simply an attack in this work, is an action which undermines the security of computer-systems and networks for malicious purposes [37]. There are a few main types of attacks that can occur on a network controller as described in the literature: one is the DoS (Denial of Service) attack, which has the goal of preventing communication on the network. This is an attack on availability. Another is a deception/integrity attack, which is defined as an attack in which a hostile entity sends intentionally incorrect information so as to manipulate the control system [2]. In the scope of this work, an attack is a cyber-attack and not a physical attack. A physical attack is one that would involve an enemy physically causing harm to the system. A physical attack, of course, might be considered almost totally indistinguishable from a fault from the standpoint of the supervisory system if the only data used comes from the physical system. It will be assumed that the physical security of the facility where the SCADA system performs its operations is sufficient to prevent any adversary from breaking in. Because the scope of these attacks is limited to cyber-attacks like other work in the literature of SCADA security, it should be understood that these attacks would require the use of networks and the compromising of nodes in order to go about their intended action of penetrating defenses of networks. Knowing this, it can be assumed that information from the network would help in detecting an attack. Or at least, it would show that what is occurring within the network is correlated with the physical behavior seen by the SCADA system. Also, attacks can be categorized as the targeted attacks and the non-targeted attacks. What differentiates the targeted attacks from the non-targeted attacks is that with the targeted attacks, the attacker has an understanding of the system and is therefore able to influence the system in a more harmful way. Non-targeted attacks do not require such knowledge. DoS attacks many times can be considered non-targeted since these attacks do not require detailed knowledge of the NCS and, in particular, the physical system. All that is necessary is to stop communication. Different types of DoS attacks are as follows based on how they work: compromising the nodes in the network, preventing the nodes from sending data, and flooding the network with packets [1].

For the scope of this work, it must be understood that the goal of the DoS attack would be to disable the control network. The control network connects various nodes associated with the process. Specifically, it links the controller to the plant. The DoS attack would thus hinder or completely block actuators from receiving commands from the controller. The DoS attack likewise impedes data sent from the sensor to the controller. As a result of this, actuators would not be able to respond to the plant correctly, which may lead to disastrous consequences. It should also be noted that the way the system responds to a DoS attack as far as the nodes are concerned is that the value in the last packet received in a node from another node is the one that is used. For instance, this might be the value for the sensor data contained in a packet that is last received by the controller. This was done it the work by Huang et al. [1].

**Two Paradigms of Detection of Cyber-Attacks for SCADA Systems:**

In order to go about the problem of distinguishing attacks from faults, it is helpful to look at two main areas of research: (1) Fault Diagnosis; (2) Intrusion Detection Systems (IDS).

Fault Diagnosis is a process of detecting the presence of faults and isolating the faults in the system to determine what components have faults. Fault Diagnosis may include identification, which involves finding a quantitative measure of the fault's magnitude [13] Fault Diagnosis is important because the techniques used in fault diagnosis may be applicable to detection of attacks. In fact, in some of the recent research, techniques from fault diagnosis are used [3]. The reason that Fault Diagnosis is important to understand is that a cyber-attack may cause the system to deviate from normal behavior in the physical system. This ability to detect when such a deviation occurs is very important for SCADA system security.

Also, there is research concerning IDS, which are defined as systems which are used to determine if a computer system or network is compromised by malware so that the threat can be dealt with. This is done with the hope that the system would be able to return to its normal functioning state if the threat is detected and eliminated [24]. This research involves detection of anomalous network behavior. These techniques can also be used for SCADA systems. There is a paper that shows how these systems can be adapted for SCADA systems [35].

**Fault Diagnosis**

In the literature of fault diagnosis, there have been many techniques that have been developed. [13] Some that are of particular interest are those that use model-based diagnosis, which as mentioned before, involves a simulation of the plant as the real plant goes about its operation and a comparison between the simulation and the real plant. Model-based diagnosis generally involves a residual generator for diagnosis. The residual generator allows certain attributes of the signal or signature to be enhanced so that isolation can be performed in addition to detection. In the case of this work, it is quite possible to use a residual generator, but as far as model-based diagnosis is concerned for this work, only detection is necessary. As for distinguishing one signature from another even in the physical system, possible methods could involve collecting some of the data from the physical system to be used as inputs to a machine learning algorithm [16].

**Techniques used in IDS**

Anomaly detection is one method that IDS techniques can use. The main idea behind anomaly detection is that if there is behavior that deviates from normal behavior, then it can be determined that an anomaly is occurring. This detected anomaly would be a possible sign of a malicious attack. Many times, a supervised learning algorithm can be trained to recognize normal behavior. These algorithms can also be trained to recognize malicious behavior. Many of these algorithms that are used involve training where certain features that are based on data from the network or computer system are used in the training process [15].

It is necessary in this work to combine these two paradigms: One way of doing this would be to take information from the physical system and extract some of this data to be used as features to train the machine learning algorithm. These features can then be combined with features from the network so that it is possible to establish uniqueness between circumstances that are different in nature, but may be similar from the perspective of the physical system only.

## Statistical Methods of Machine Learning

It is the desire of this work to be able to understand the situations that the NCS experiences so that the data can be analyzed to make distinctions between attacks and faults. One key feature of an attack is that it may influence the behavior of the network in addition to the physical system, and therefore the behavior in the physical system is correlated with the behavior on the network. What is noteworthy about a control network is that it is different from a standard corporate network in that the network exhibits regular behavior, meaning that a change in the network behavior could signal that a cyber-attack may be occurring. This is certainly true if the control network is separate from the corporate network. It can also be assumed true if there is a connection between the corporate network and the control network since in normal circumstances, the corporate network would not be allowed to disrupt the control network, or at least, the interaction would be hardly noticeable in terms of the network traffic. This may allow a distinction to be made, which a machine learning algorithm may be able to make if there are unique signatures associated with the different situations experienced by the NCS. These different situations may be said to fall under different classes or types of scenarios that the NCS experiences. For instance, three classes that would be appropriate for this work would be normal, fault, and attack. The machine learning algorithms that would classify the circumstance according to the data given are simply known as classifiers. One way to create a classifier for this work may be to use a naïve Bayes classifier [16], which could receive certain data from the system as features. The rationale for using Bayesian techniques would be that these techniques are also used for some intrusion detection systems. It is also the desire of this work to use a statistical approach to classification.

The algorithm for a naïve Bayes classifier is a supervised learning algorithm. It is fairly accurate, although it may not be as accurate as more advanced Bayesian classifiers. An advantage that this classifier has that the work in [36] discusses is that it is not very computationally expensive. This classifier involves Bayes rule as the name implies and relies on an independence condition. For this, the attributes are assumed or treated as if they are not related. The classifier must be trained since it is a supervised learning algorithm. For this algorithm, the means and the variances are calculated for the attributes of the training data for each class. [36].

Suppose that there are q classes, then the following equation is used for the classification process.

$$l = \arg\max_{k=1,\ldots,q} \hat{f}_k(x^*) \tag{2.4}$$

Where *l* is the class that the machine learning algorithm determines to best fit the testing data that it receives. Basically what the above formula means is that there is a set of functions calculated and the maximum is the one that is selected whose subscript is what *l* will equal.

For the Naïve Bayes's classier,

$$\hat{f}_k(x^*) = \frac{1}{Z} p(C) \prod_{i=0}^{n} p(F_i|C) \tag{2.5}$$

The above equation is based on Bayes' rule, where z is the evidence. $F_i$ refers to a feature i. P is the Gaussian distribution, which is as follows:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{2.6}$$

Where μ is the mean, and σ is standard deviation [37].

## Previous Work

There are several papers in the literature that are related to this work: The paper by Cardinas et al. is one of the first papers that point out one of the main differences between SCADA security and general network and computer security. The SCADA system is different in that the physical system is involved, and therefore the behavior of the physical system should be taken into account. In other work, it was mentioned that SCADA security is different because the networks and devices involved do not have the same computing capabilities as the computers used in general IT networks [2]. Litrico et al. discusses the isolation of faults and attacks and mentions the difficulties in doing so.

# CHAPTER III

# EXPERIMENTS

In order to go about testing the ideas of distinguishing faults from attacks, a certain networked control system is modeled, and an implementation is created for simulations. Also, the methods for classifying unknown situations are discussed.

## Problem Formulation

The scope of the problem encompasses a networked control system that is subject to various circumstances. Some of these circumstances include cyber-attacks that affect the physical system. They also include faults and normal operating conditions. It should be noted and emphasized that the cyber-attacks of interest are the attacks that have an impact on the physical system. This type of impact is important to study because of its potential to cause danger in the physical world or its potential to cause inefficient operation for the networked control system. The physical world includes the plant and its physical environment. Other cyber-attacks, which do not impact the physical system, are out of scope for this work. Those other types of cyber-attacks may be considered more or less benign in terms of the physical destruction that they cause but they may involve stealing information from the system. They could be an attack on confidentiality in other words. One of the objectives of this work is to be able to detect abnormal behavior in the system due to attacks or faults. Detecting abnormal behaviors is similar to other work that has been done previously [2], [3], [4]. Furthermore, this work goes beyond detecting abnormal behavior. If abnormal behavior is detected, then it is desired to understand the nature of the circumstances responsible for this behavior; that is, it is desired to know what is causing the abnormality. It would be useful to know this information so that a more appropriate response can be used according to the type of abnormal situation. Therefore, it is important to diagnose the system further as a primary goal of this work and gain an understanding of the limitations in doing so as well. There are several aspects of this problem that will be discussed in the following paragraphs.

## The Networked Control System

The hypothetical networked control system (NCS) in this problem incorporates a plant and a controller, each of which is able to communicate to the other by means of a network in ordinary operating conditions. The communication is done using datagrams, or basic units of information that can be sent over the network [7], [8]. The plant sends datagrams over this network to the controller. These datagrams originate from the sensors associated with the plant and contain information concerning readings from those sensors. The controller, likewise, sends commands for the actuator associated with the plant by means of datagrams through the network. These datagrams sent by the controller to the plant are sent at regular intervals. Likewise, the datagrams sent by the plant to the controller are also sent at regular intervals. Therefore, under normal operating conditions, it can be assumed that the overall network traffic will be regular as far as its bandwidth utilization is concerned. Bandwidth can be defined as the amount of data that can be sent over the network in a given amount of time. Bandwidth utilization is how much of the bandwidth is being used and is typically represented as a percentage. The controller would enable the system to meet the control objectives, which can be defined as the desired result for the plant or the way that the plant is required to behave as it is manipulated by the controller [10]. As a control objective, it may be desired to have a variable associated with the plant to reach a certain setpoint or to have it track a time-varying reference signal. A diagram of the setup for the generic networked control system can be seen as follows:

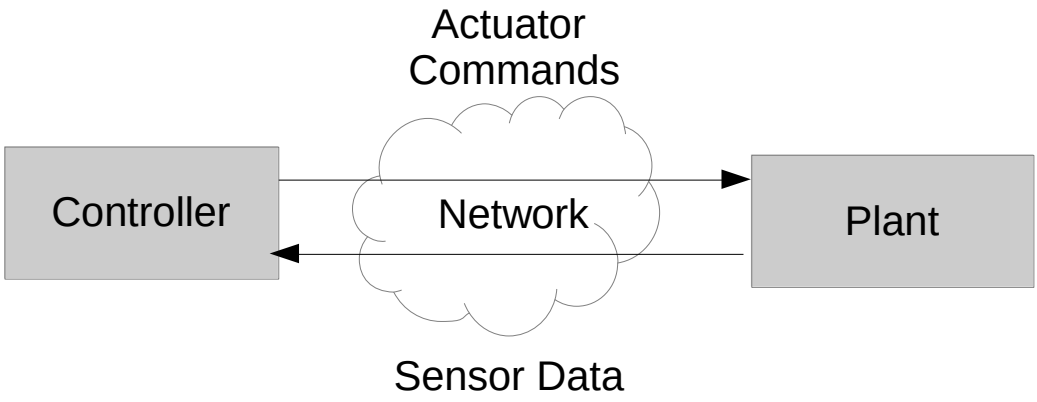


**Figure 3.1: Generic Networked Control System with a Controller, a Plant, and a Network Represented by a Cloud with Arrows that Represent Streams of Data**

In Figure 3.1, the boxes represent the plant and the controller. Also, in the figure is a cloud to represent the network. There are also arrows which indicate two streams of datagrams in the opposite directions between the controller and the plant. One stream is from the controller to the plant over the network. This stream contains commands for the actuators associated with the plant from the controller. The other stream is from the plant to the controller. The datagrams in this stream come from the plant's sensors and are sent to the controller so that the controller can respond to meet the control objective it was designed for. Within the sensor stream, the plant sends these datagrams at a fixed sampling rate, and the datagrams are time-stamped. There is also a delay incurred as the datagram travels through the network. In addition, the controller sends datagrams at a fixed sampling rate as well. Therefore, the network has regular behavior as far as the traffic is concerned.

## The Specific Problem Selected

It is important to note that Figure 3.1 shows a generic networked control system and not a concrete one. But, of course, a concrete one is best suited for the purposes of experimentation. More specifically, it is desired that a relatively simple networked control system be used for the experiments so that there is no unnecessary complexity beyond what is reasonable for a project such as this. These situations include normal operation, and different cyber-attacks and faults.

The specific networked control system chosen involved a fairly simple two-tank water system as the plant. This plant was regulated by a controller, and communication was facilitated by a simple network. It should be noted that this particular NCS will be described in more detail in the paragraphs that follow as well as the various situations that it can be subject to. This specific NCS was chosen as opposed to a much more complex one such as the Tennessee Eastman Process Control System (TEP-CS) even though the TEP-CS is widely used in the literature. Its widespread use is evidenced by the fact that many of the papers associated with SCADA security use this system [1], [2], [5]. Using an overly complex system would mean that the faults and attacks may become very complex to simulate and lead to a large number of simulations that may become intractable for this work.

**The Plant: The two-tank system**

The plant chosen for these experiments is a two-tank system for water. A model of this system can be seen in Figure 3.2. It should be noted that this two-tank system is similar to one used in the thesis by Zhou [9]:

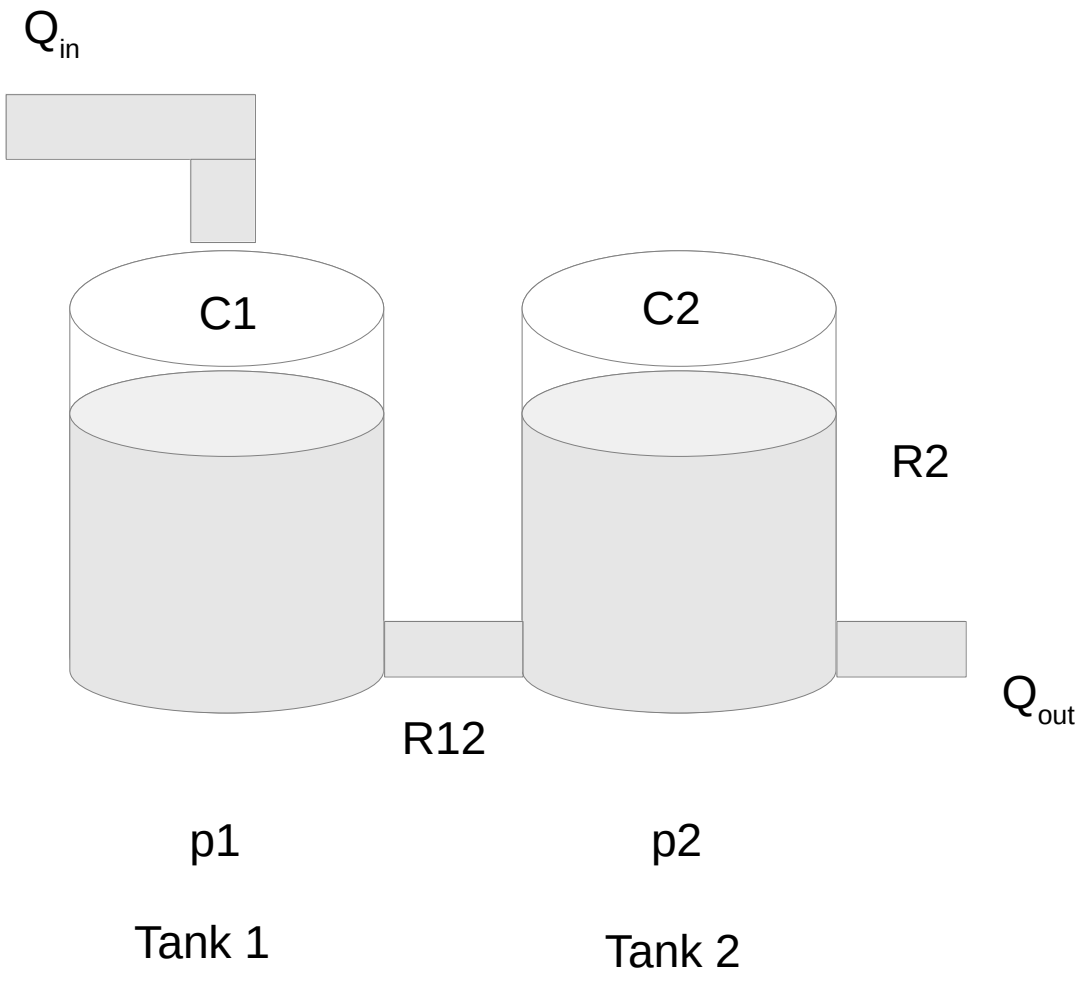


**Figure 3.2: Illustration of the Working Plant**

In Figure 3.2, there is a valve over the first tank, which opens to allow water to flow into the tank. This valve is the actuator for the plant that receives commands from the controller through the network. Also, at the bottom of the second tank is an outlet for the water. Between the two tanks is a pipe that links the tanks and allows water to flow from one to another. Initial conditions are such that both of the tanks start empty. There are two sensors associated with the plant, one for tank 1 and another for tank 2. These sensors measure the pressure of the water in the tanks.

The tanks have certain dimensions and quantities associated with them. These include the cross sectional area of the tanks (A), the mass density of water (ρ), gravitational acceleration (g). There are certain quantities associated with the tanks-P1 and P2-which are the pressures at the bottoms of the tanks 1 and 2 respectively. C1 and C2 are capacitances of the tanks. The R12 is the resistance of the pipe between the tanks. R2 is the resistance of the pipe attached to tank 2 that allows water to flow out. The pressures are the state variables for the state-space model of the system that will be derived. First of all it, it is necessary to establish the relationship between the pressure (P), the volumetric flow rate (Q), and the capacitance (C) for each tank:

$$\begin{cases} P_1 = Q_{C1}/C_1 \\ P_2 = Q_{C2}/C_2 \end{cases} \tag{3.1}$$

Where $P_1$ and $P_2$ are the pressures at the bottoms of the tanks 1 and 2 respectively, and $Q_{C1}$ and $Q_{C2}$ are the net flow rates of the tanks.

The net flow rate for each tank is the following:

$$\begin{cases} Q_{C1} = F_1 - Q_{R12} \\ Q_{C2} = Q_{R12} - Q_{R2} \end{cases} \tag{3.2}$$

For tank 1, water can flow into the tank ($Q_1$). Water can flow through the pipe between the tanks as well. $Q_{R12}$ is the volumetric flow of water out of tank 1 and into tank 2.

The flow rates for the two tubes associated with the plant are as follows:

$$\begin{cases} Q_{R12} = (P_1 - P_2)/R_{12} \\ Q_{out} = P_2/R_2 \end{cases} \tag{3.3}$$

The two-tank system is actually a non-linear system. The resistance is not a constant with respect to the pressures in the tanks in reality. However, in this work, the resistance is given as a constant to simplify an otherwise complicated system. Because it is a dynamic system, it is described by a set of differential equations, which are as follows:

$$\begin{bmatrix}\dot{P}_1\\ \dot{P}_2\end{bmatrix} = \begin{bmatrix}-\frac{1}{R_{12}C} & \frac{1}{R_{12}C}\\ \frac{1}{R_{12}C} & -\frac{1}{R_2C}-\frac{1}{R_2C}\end{bmatrix}\begin{bmatrix}P_1\\ P_2\end{bmatrix} + \begin{bmatrix}\frac{1}{C}\\ 0\end{bmatrix} Q_{in} \tag{3.4}$$

These equations are written in linear state space form, Therefore $P_1$ and $P_2$ are the state variables.

Also, for simulation purposes, it is necessary to use concrete values for the quantities associated with the model. The quantities chosen can be seen in the following table [42]:

| | Capacitance ($m^4s^2$/ kg) | Resistance ($N{\cdot}s/m^5$) |
|---|---|---|
| Tank C1 | $1.5708*10^{-6}$ | - |
| Tank C2 | $1.5708*10^{-6}$ | - |
| Tube R12 | - | $7.35*10^{7}$ |
| Tube R2 | - | $1.45*10^{8}$ |

**Table 3.1: Values for Resistances and Capacitances**

Using the previous state space-model derived, the following equations can be determined with actual numerical values for the parameters. $F_{in}$ is a variable that can range from 0 to 1.

$$\begin{bmatrix}\dot{P}_1\\ \dot{P}_2\end{bmatrix} = \begin{bmatrix}-0.0087 & 0.0087\\ 0.0087 & -0.0262\end{bmatrix}\begin{bmatrix}P_1\\ P_2\end{bmatrix} + \begin{bmatrix}636181\\ 0\end{bmatrix}F_{in} \tag{3.5}$$

The above equation (3.5) is expressed in continuous-time, and can be used directly in MATLAB Simulink for the behavior of the plant.

**The Controller**

The controller for this project is a PI controller. The setpoint for this controller varies as a function of time. The controller receives values from the sensor for tank 2 and responds to the plant to meet the control objective, which is to have the level of the water in tank 2 track a time varying reference signal. Specifically, in this work, it is the pressure of tank 2 that is to track with the reference signal. The PI controller is implemented as a discrete-time controller for the actual implementation. The sampling rate is set at 100Hz. This rate was chosen as an academic example, rather than a value that may be seen in an actual system.

The reference signal is a piece-wise function defined by a set of equations.

$$f(t) = \begin{cases}\frac{1}{160}(t-220) & : 220\\ 1 & : 380 < t < 460\\ \frac{1}{160}(620-t) & : 460 < t < 620\\ 0 & : Otherwise\end{cases} \tag{3.6}$$

$$\mathrm{F(t)} \;=\; 10000\mathrm{f(t)} \tag{3.7}$$

The function F(t), which is used as the reference signal can also be seen in the following plot (Figure 3.3).

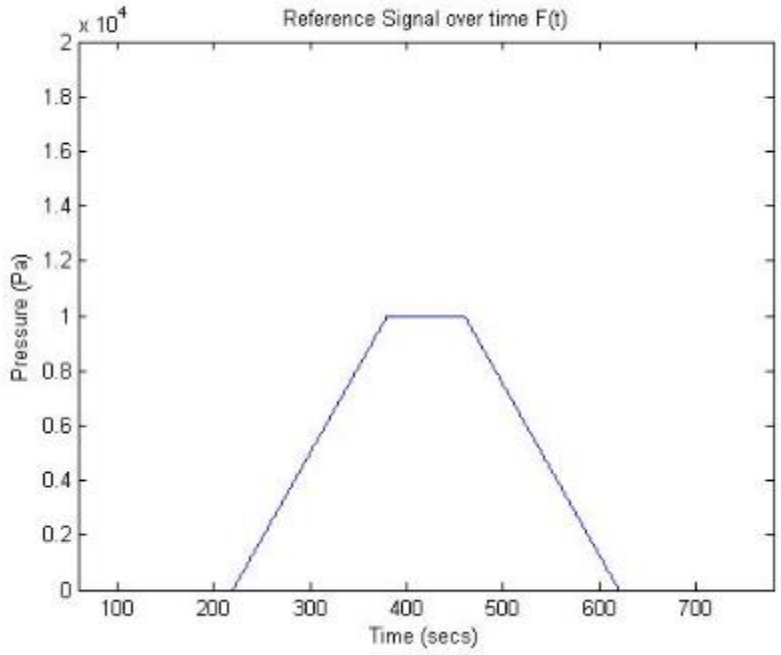


**Figure 3.3: Plot of Reference Signal**

Also, within the controller is an anomaly detection module (ADM) similar to the one used in the paper by Cardinas et al [2]. This part of the controller is implemented by using a simulated mathematical model of the plant. A comparison can be made between the ideal behavior of the plant as represented by the model and the behavior of the plant as indicated from information in the datagrams sent from the plant to the controller. It should be noted that the plant used in the simulations is itself a mathematical model as described previously, but it can be considered to be the "real plant." The mathematical model that is implemented in the controller is a model of this "real plant" and can be viewed as a simulation within a simulation. This model is the same as Eq. 3.4.

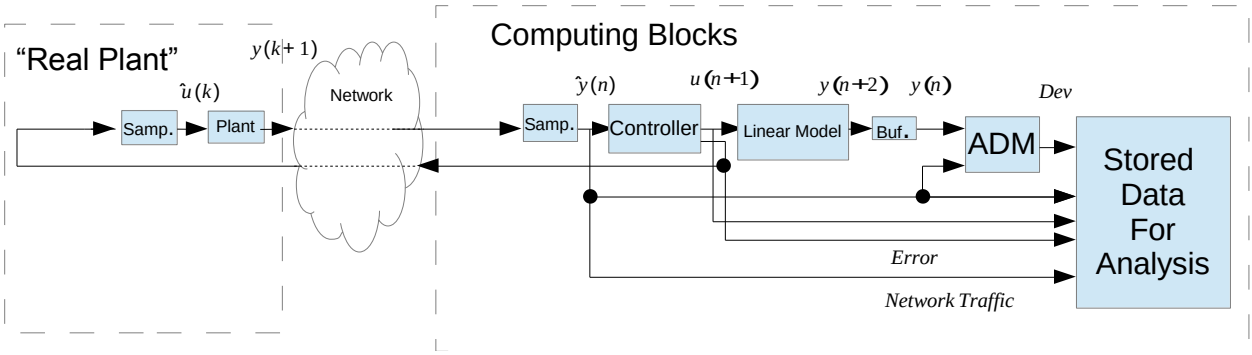


**Figure 3.4: Illustration of the NCS with Topology Used in This Work**

## Network

The following (Figure 3.5) is a diagram that shows the specific network topology that was chosen in this work. It was chosen so that there could be multiple routers. It was also chosen to be relatively simple.

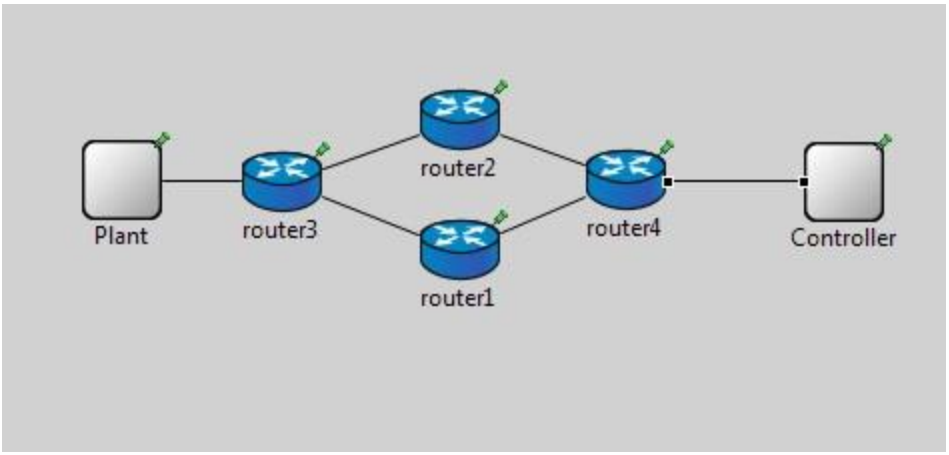


**Figure 3.5: Illustration of the NCS with Topology Used in This Work**

There was a certain bandwidth associated with the network and delay as well. Each channel in the network has the following characteristics: datarate = 1.0Mps, and delay = 10us. Also, it is important to understand the datagrams that are used in this work. The datagrams can carry certain information through the network. This information included a timestamp that indicates the time they were created by the sensors. Also, these datagrams contained the sensor readings

that are sampled every 10ms. for the plant. Therefore, the datagrams are also sent every 10ms once they are created. The byte length of the datagrams is 20 Bytes.

The controller receives this information from the plant over the network, which it then uses to update its variables that store the last values received, which is a similar setup to what was done in the paper by Huang et al [1].

**Possible situations**

For this work, certain situations that the networked control system can experience are studied. The situations can be categorized on three different levels as is illustrated in the following diagram in Figure 3.6. At the high level, they can be divided into two main categories: normal and abnormal. The abnormal scenarios can be further divided into faults and attacks, which are designated as the mid-level categories. Both faults and attacks are studied in this work since both of them can impact the physical system. These mid-level categories can be further divided into more categories on the low level: For the faults, the low-level categories consist of a broken pipe, the small sensor bias, the large sensor bias, and the unresponsive actuator. For the attacks, the low-level categories are DoS flooding attacks, injection attacks on the controller, and injection attacks on the plant. The next paragraphs describe all of these categories on all three levels.

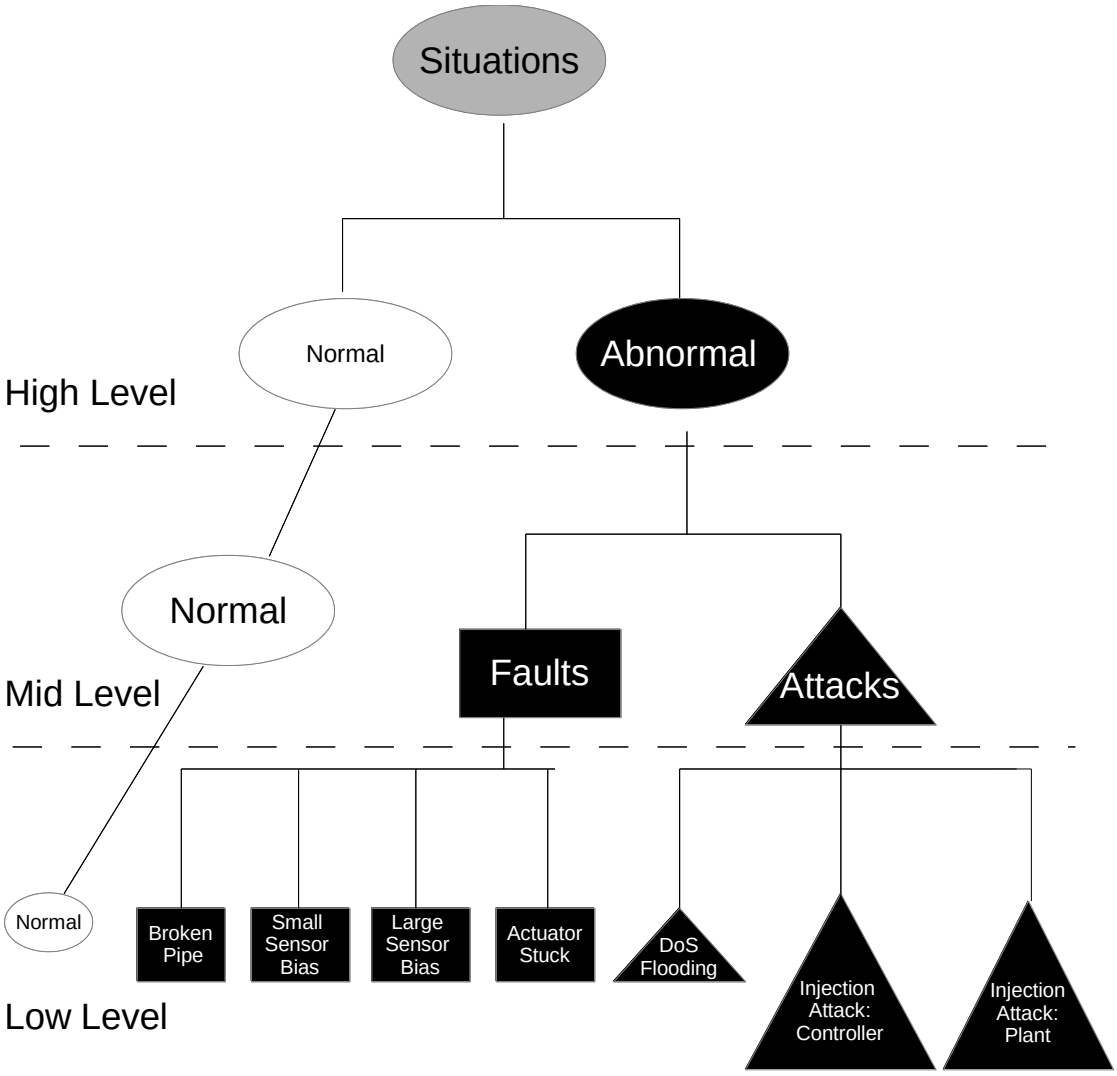


**Figure 3.6: Three Levels of Categories of Situations (White shapes represent normal. Black shapes represent abnormal. Of the black shapes, squares represent faults, and triangles represent attacks)**

### Normal

Normal behavior would simply be defined as behavior that the system is designed to have without any influence from attacks or faults. If normal behavior is occurring, then there would ideally be no deviation from the simulated plant in the controller that is large enough to be considered unusual or abnormal by the SCADA system. For this case, the controller would be able to meet the control objective.

**Abnormal**

Abnormal scenarios are defined as those in which the networked control system experiences situations that cause it to deviate from normal behavior. In some cases, this may cause chaotic and dangerous behavior for the physical system. These abnormal scenarios are divided into faults and attacks.

**Faults** Faults are degradations in the plant that are accidental and not caused by a malicious entity. These faults can be due to damage in the physical plant. In addition to the faults that are related to the plant itself, faults can be associated with the sensor and the actuator [9], [10]. The faults in this work are all persistent, meaning that once they occur they will continue to have their effect throughout the operation of the plant or until the operation of the plant is stopped. At which point, the problem can be addressed.

*Faults in the Plant: Damaged Pipe* One of the faults could be a damaged pipe between the two tanks. If this occurs in the plant, the model for the plant must change to account for the leak that is introduced, which will cause a change of behavior in the dynamics of the plant. Therefore, two mathematical models for this system must be used: One for a working plant and one for a damaged plant. This would mean that this plant is a hybrid system. A hybrid system incorporates both discrete states and continuous states. The plant would start in the working discrete state and therefore would use the mathematical model of the plant associated with that state. In the event that the pipe breaks, there would be a change of state to the other discrete state. However, there would be no change back to the original discrete state during continuous operation of the plant. If the pipe is to be repaired, continuous operation would be stopped before the repairs can take place. It should be noted that the instant that the model changes, the state variables associated with the continuous model would remain unchanged. In other words, the water levels in the tanks remain the same at the very instant that the pipe becomes broken. But as time progresses, the levels in the water would change according to the new model. For the case of this project, since the standard state space equations were used, the parameters in this new model would be different than those of the working model. The model would change such that the pressure in tank 1 would not affect the pressure in tank 2. Likewise, the pressure in tank 2 would no longer affect the pressure in tank 1. The reason for this is that tank 1 and tank 2 no longer have a pipe to link them together, which allows water to flow so that pressure is

transferred from one tank to another. Therefore, the plant can be visualized by the following figure.

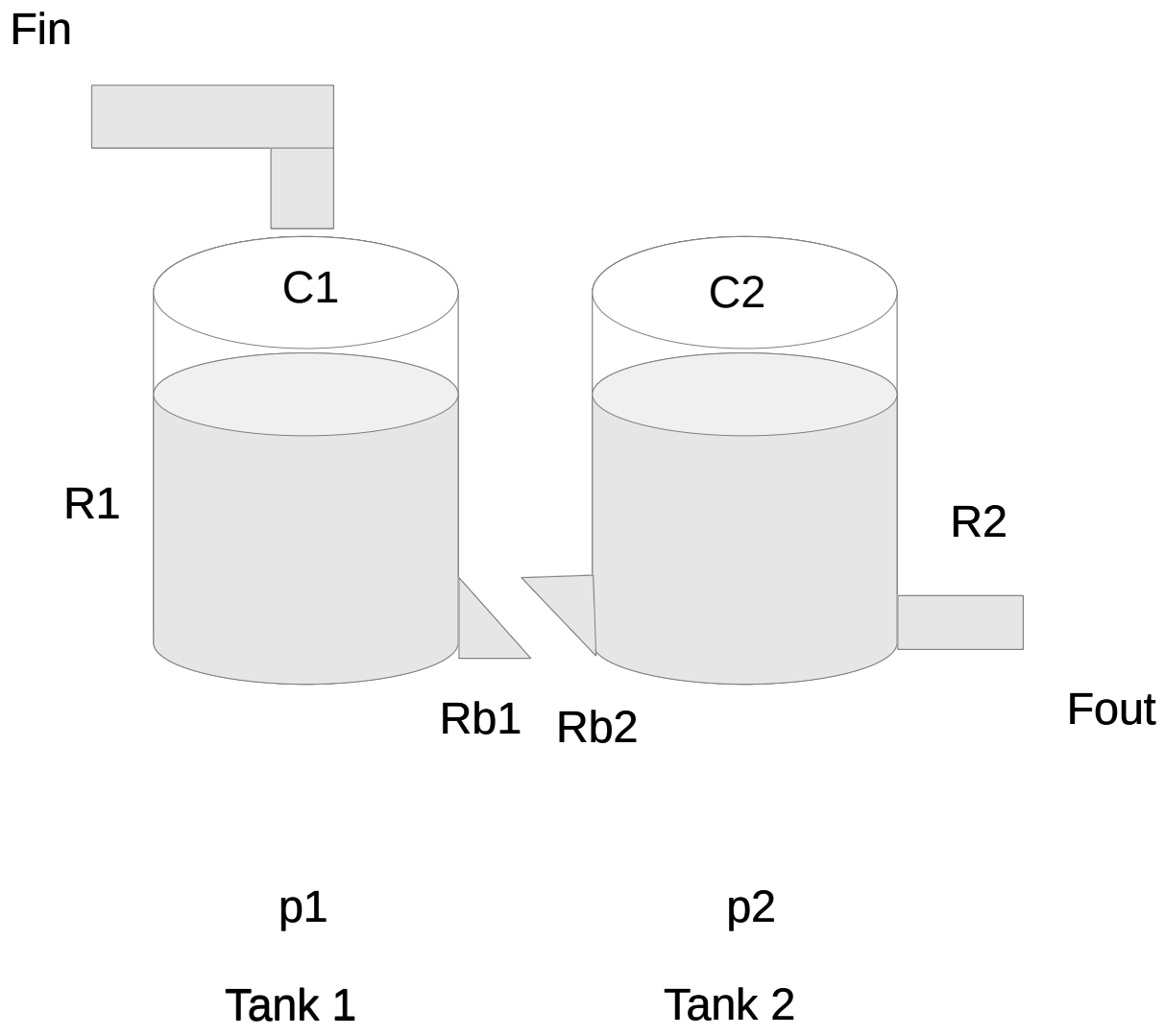


**Figure 3.7: Illustration of Broken Plant**

where Rb1 and Rb2 are the new resistances for the broken pipe.

The state space equations would then become the following:

$$\begin{bmatrix}\dot{P}_1\\ \dot{P}_2\end{bmatrix}=\begin{bmatrix}-\frac{1}{R_{b1}C} & 0\\ 0 & -\frac{1}{R_{b2}C}-\frac{1}{R_2C}\end{bmatrix}\begin{bmatrix}P_1\\ P_2\end{bmatrix}+\begin{bmatrix}\frac{1}{C}\\ 0\end{bmatrix}F_{in} \tag{3.8}$$

It should be noted that the A matrix (This matrix found in Equation 2.3 in Chapter 2 for the general form of state space equation) of the state space equation has zeros on the off-diagonal entries. This is because pressure 1 no longer affects pressure 2 and vice-versa as was mentioned since there is a leak. Equation (3.8) expressed with the numerical values for the parameters is as follows Equation 3.9:

$$\begin{bmatrix}\dot{P}_1\\ \dot{P}_2\end{bmatrix}=\begin{bmatrix}-\frac{1}{R_{b1}C} & 0\\ 0 & -\frac{1}{R_{b2}C}-\frac{1}{R_2C}\end{bmatrix}\begin{bmatrix}P_1\\ P_2\end{bmatrix}+\begin{bmatrix}\frac{1}{C}\\ 0\end{bmatrix} \tag{3.9}$$

*Faults in the Actuator* There are also faults that can occur with the actuator. One possible fault for the actuator is the actuator becoming stuck or a situation where it does not respond to commands. For this particular fault, the valve that is used as the actuator in this project remains at its current position.

*Faults in the Sensors* Other faults include faults with the sensors. For instance, it is possible for there to be a bias in the sensor readings. This could be due to mis-calibration. For this type of fault, constant biases were added to both signals that are generated by the sensors. These biases are introduced at some start time, and the bias persists throughout the rest of the simulation. Therefore, each of these biases can be represented as a step function with a certain delay once the simulation starts. There were two biases that were used in this work. The signal due to the "small" bias was 5,000Pa (50% of the highest value of Reference Signal) less than

what it should have read once the fault occurs. The signal with the "large" bias was 10000 Pa (100% of highest value of Ref. Signal) less than what it should have indicated.

**Attacks** Some of the possible attacks that can influence the behavior of the system include DoS attacks, and injection attacks.

*DoS Attacks* The DoS attacks can prevent the network from allowing communication between the plant and the controller. This can be achieved by flooding of a network with datagrams. Because of this flooding of datagrams, legitimate traffic would not be able to flow. This would mean that traffic from the real sensor to the controller would not be able to reach the controller. As a result, the NCS is prevented from functioning properly. This would also be true for traffic from the controller to the actuator.

*Injection Attacks* Injection attacks were also used in this work. For these injection attacks, malicious datagrams that appear to be legitimate would be sent to either the controller or the plant in order to manipulate the behavior of the NCS. These datagrams that are directed to the controller will have false information for the sensor readings. Likewise, these datagrams that are directed toward the actuator will have commands for malicious purposes. The plant or controller, of course, will respond according to the information sent to it. If the plant consists of tanks for water or tanks that contain some other liquid in it, it may make sense from the attacker's standpoint to influence the system such that the tanks overflow. The attacker could achieve this by sending datagrams with commands to keep the valve open in the plant. Therefore, the attacker may desire to inject datagrams with values to the actuator associated with the plant to keep the valve open. If the attacker is sending packets to the controller, the attacker would desire to send datagrams to cause the sensor readings to appear deceptively low. Therefore, the controller will respond by sending the datagrams to cause the valves to be opened more, which may eventually cause the tank to overflow.

**Variations in Scenarios**

It should also be noted that there were variations among the different scenarios, specifically in the low-level categories illustrated in Figure 3.6. These scenarios were those with faults and attacks. If they occurred in reality, these abnormal situations could start at any time and end at any time. Because of this, for a real NCS, there would be an infinite number of possible combinations of start times and end times even for a single occurrence of an attack or fault lasting for one period during the simulation. Despite this, it is still desired for the attacks to have different start times and end times associated with them for these different variations in the experiments during the operation of the NCS. Also, it is desired for the faults to have different start times associated with them and to persist throughout the simulation. Therefore, it is necessary to have a way of reducing the potentially infinite set of possibilities to a finite set for experimental purposes such that the whole space of possible simulations would still be covered. To achieve this, there were certain points designated for the start times and end times in the simulation period set at equal intervals from one another. It was therefore possible to create a finite set of all possible combinations of start times and end times restricted to these designated points. For the case of this work the set of designated points in terms of seconds into the simulation is the following: {60, 204, 348, 492, 636, 780}. Each set of start times and end times is such that the period of the attack or fault is at least the smallest period, which is defined as the period between two of these points which are adjacent. The following images illustrate these variations. Each of the low-level categories of the attacks had 15 variants of start times and end times. This is illustrated in the following table, which gives the start time, the end time, and period for each variant (Table 3.2).

| Variant for Attack Categories | | | |
|---|---|---|---|
| Variant | Start Time(sec) | End Time (sec) | Period (secs) |
| 1 | 60 | 204 | 144 |
| 2 | 204 | 348 | 144 |
| 3 | 348 | 492 | 144 |
| 4 | 492 | 636 | 144 |
| 5 | 636 | 780 | 144 |
| 6 | 60 | 348 | 288 |
| 7 | 204 | 492 | 288 |
| 8 | 348 | 636 | 288 |
| 9 | 492 | 780 | 288 |
| 10 | 60 | 492 | 432 |
| 11 | 204 | 636 | 432 |
| 12 | 348 | 780 | 432 |
| 13 | 60 | 636 | 576 |
| 14 | 204 | 780 | 576 |
| 15 | 60 | 780 | 720 |

**Table 3.2: Attack Variants based on Start Time and End Time for Each Low-Level Category.**

In addition to this, there are 5 variants for the faults since only the start times are varied (Table 3.2). The end times are not varied because they are assumed to be persistent, meaning they will last until the end of the simulation.

| Variants for Fault Categories | | |
|---|---|---|
| Varient | Start Time (sec) | End Time (sec) |
| 1 | 60 | 780 |
| 2 | 204 | 780 |
| 3 | 348 | 780 |
| 4 | 492 | 780 |
| 5 | 636 | 780 |

**Table 3.3: Fault Variants for Each Start Time and End Time for Each Low-Level Category**

In addition, the following table shows the number of variations for each of the low-level categories and their combined sum, which is referred to in the table in bold text as the "Total Unique Scenarios".

| Classification Scheme | | | |
|---|---|---|---|
| High Level | Mid Level | Low Level | Num of Varients per low -level class |
| Normal | Normal | Normal | 1 |
| Abnormal | | | |
| | Faults | | |
| | | Damaged Plant | 5 |
| | | Small Sensor Bias | 5 |
| | | Large Sensor Bias | 5 |
| | | Actuator Stuck | 5 |
| | Attacks | | |
| | | DoS Flooding | 15 |
| | | Injection to Plant | 15 |
| | | Injection to Controller | 15 |
| | | **Total Unique Scenarios:** | **66** |

**Table 3.4: All Unique Scenarios Used**

It is especially important to note that when using a time-varying reference signal, it is necessary to use attacks and faults that start at different times since they have different effects on the NCS. It is also reasonable to vary the end times for attacks as well so that the lengths of the periods of the attacks can be different.

**Symptoms**

The major goal of this work is to be able to diagnose what type of situation the SCADA system is experiencing if there is abnormal behavior. The recent work [2], [3], [4] that has been done involves observing the system in order to detect whether there is an anomaly in its behavior in the physical system. In this work, something similar to this was done, but more was done so that it can be determined whether the anomaly is due to an attack or fault. More information would have to be gathered from the NCS to make this determination. Such necessary information would include information from the network as well, such as the inter-arrival time of packets, the network traffic, whether the controller was updated with new information, etc. Therefore, the behavior of the system can be observed by collecting information from the physical system and the network. In the networked control system, it is possible that either the behavior in the

physical system, the network, or both may be affected. An abnormal situation, whatever it may be, would influence the system a certain way depending on the nature of the abnormal situation. By using the information that is collected, certain symptoms associated with the network control system may be seen. Different types of abnormalities would have different sets of symptoms associated with them. For the purposes of this work, these sets of symptoms can also be referred to as the signatures. If these signatures for the different abnormal situations are unique then it is possible to distinguish one abnormal situation from another. Therefore, if a specific abnormal situation occurs, it is possible to classify that situation into a certain category. There are several symptoms that are of interest. One would be a deviation of the behavior of the plant according to sensor data from a simulated model of the nominal plant. Other symptoms may be changes in network activity. In this work, there are three different levels of categories for scenarios. It would thus be desired ideally to categorize the status of the NCS correctly on each level.

## Implementation of the experiments

### Implementing the Models in Software

The equations referred to in the previous section cannot be directly used in C++ very easily since they are in continuous form and are therefore not suitable for simulations of a digital system. Therefore, these equations must be transformed into discrete equations. There are mathematical formulas to allow for this transformation. The author of this work did not use these methods directly, but instead relied on MATLAB scripts with certain commands to generate a new set of equations in discrete form from the continuous equations. The resulting equations are as follows along with the appropriate matrices: The discrete state space equations for the two-tank system with the pipe undamaged at 100HZ:

$$\begin{bmatrix} \dot{P}_1 \\ \dot{P}_2 \end{bmatrix} = \begin{bmatrix} -.9999 & 0.0001 \\ 0.0001 & -.9999 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} + \begin{bmatrix} 6365.9 \\ 0 \end{bmatrix} \tag{3.9}$$

It should also be noted that equation 3.9 is also used to implement that model in the ADM of the controller to serve as a basis of comparison. The discrete state space equations for the two-tank system with the pipe damaged, also at 100HZ:

$$\begin{bmatrix} \dot{P}_1 \\ \dot{P}_2 \end{bmatrix} = \begin{bmatrix} -0.9999 & 0.0 \\ 0.0 & -.9997 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} + \begin{bmatrix} 6365.9 \\ 0 \end{bmatrix} \tag{3.10}$$

Similarly, it was necessary to create a discrete control algorithm for the controller with the correct parameters for the control law. This was also done using MATLAB. Basically, a model of the discrete plant was made in Simulink. In addition, a discrete controller was used for it. MATLAB has a feature in Simulink that allows for the tuning of the parameters of the controller. Using this feature, the author tuned this controller as it was connected to the plant such that it had a certain response time. This response was chosen to be 50 seconds. As a result of this tuning, parameters were obtained from the MATLAB Simulink model that could be used for the controller in the actual implementation for the simulations. The simulations were done in Omnet++ and INET [17], [18]. Here is the model created in MATLAB:

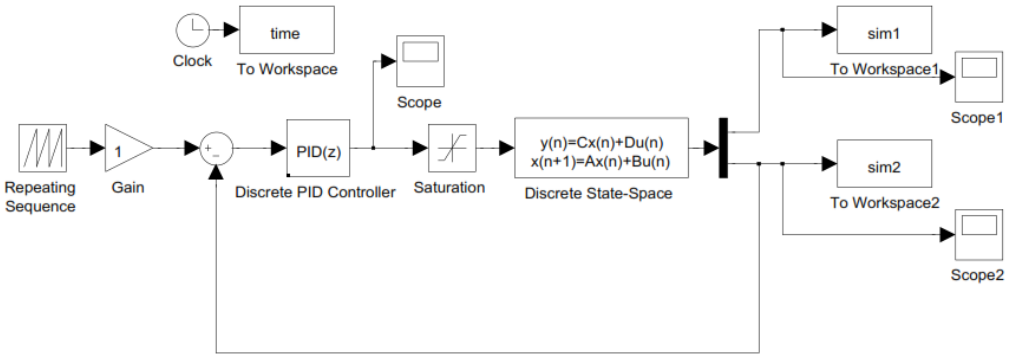


**Figure 3.8: MATLAB Implementation of the Experiments**

**Simulation Software**

Special software was used to run simulations with the various scenarios. Omnet++ and INET were chosen as libraries and frameworks that are based in C++ [6]. Omnet++ is a helpful tool for simulating networks. It is a general purpose simulator. INET is a library for that simulator that can be used for creating networks with sophisticated routers and protocols, such as UDP, TCP, etc. Omnet++ was specifically used for this work because it allows for the creation of simulated networks with nodes whose applications can serve as the plant or the controller. Although it is possible to simulate the plant and the controller in MATLAB, MATLAB does not have the tools that Omnet++ provides for networks. It should be noted, however, that there are some tools that allow for something resembling network behavior for MATLAB. TrueTime is one of them [39]. TrueTime is a simulator made for Matlab/Simulink to allow for simulations of networks with delays. Also, there has been work done with SimEvents to simulate packets being forwarded through a network link, experiencing DDoS Attacks [13]. But these tools were not quite suitable for the purposes of this work since it was desired to use a more sophisticated network simulator that allowed for more elaborate simulation, one that would involve routing packets to their destinations. Omnet++ is capable of doing such simulations. If such a simulator is to be used, the question of how to implement the plant and controller then arises since Omnet++ does not have tools that are specifically tailored for simulations of control systems like MATLAB. It should be noted that work has been done in the literature to integrate MATLAB and Omnet++ with certain simulation environments, such as C2WindTunnel that enables them to interface with one another [15]. However to use such an environment in this work may cause the simulations to be unnecessarily complex. Therefore, it was decided to add C++ code to the applications that would implement the behavior of the plants and the controller. The plant application implements the discrete equations for the plant. The state of the plant is updated at regular intervals of simulated time. The discrete equations are chosen using the MATLAB tools discussed for this specific interval of simulation time. In the case of this work, this interval of time that was chosen was 10ms as mentioned previously. This is referred to as the step time. When the controller was tuned in MATLAB, it was specified in the settings that the states of the plant would be updated every 10ms.

Omnet++, in conjunction with INET, allows for the creation of models of networks that can be simulated. It allows for simulations with the standard protocols found in networks, which include UDP and TCP, etc. The UDP protocol was used in this work. Within Omnet++ and INET, the user can create the topology of the network for the simulations and store them as NED files, which are also known as network description files. Both the controller and the plant are implemented in C++. This code is used as applications for the nodes that are connected to the network within the Omnet++ simulation environment.

**The Simulations and How They are Used**

First, the large set of the unique 66 simulations were performed using Omnet++/INET to generate the data associated with the NCS. This data was created in the form of CSV files. Then, two main things were done with the data. The first part deals with studying the data generated by the scenarios to gain an intuitive sense of their behavior. The reason for doing this first is to observe the potential harm to the system that these attacks and faults can cause. Another reason is to be able to understand intuitively how certain circumstances affect certain variables associated with the system. Then it can be understood what are the symptoms associated with a particular circumstance. The second part of the experiments deals with using a machine learning algorithm, which is included in MATLAB, so that experiments with classification can be done to determine how well a distinction can be made between different classes of scenarios. The following is a diagram that illustrates this setup:

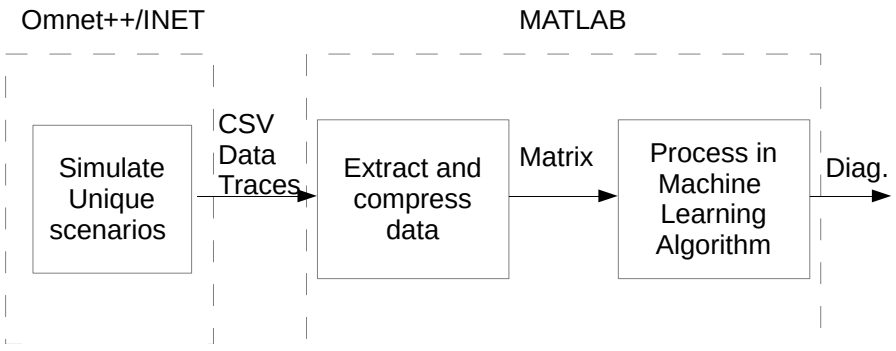


**Figure 3.9: Diagram illustrating the Overall Experiments, where Omnet++/INET generates the simulation CSV Data; MATLAB extracts/compresses the data in the form of a matrix with features as columns and rows as scenarios to be used by the Machine Learning Algorithm for diagnosis.**

Before using the machine learning algorithm, the data must be processed so that a certain very small subset can be extracted from the large set of data. This very small subset would then be used to train a Naïve Bayes Classifier. The data that are extracted would be fed as features to the classifier. The purpose of conducting experiments with a Naïve Bayes Classifier would be to test the ability for a well-known machine learning algorithm to classify these circumstances given their data. There is something that must be mentioned that is specific to the Naïve Bayes classifier: For this classifier, it must be trained using a set of features. These features come from the data. For instance, the data that is extracted from the CSV files that are created by the simulations. Data are written to CSV files as the simulations run. These CSV files contain columns of data for pressures in the tanks, the deviation of the tank 1 from the ideal tank, etc. The features, which are derived from this data, can then be used to train the classifier.

It is necessary to label the simulations from which the data is produced. This is done for the training phase according to the categories that the simulations fall under, since this is a supervised learning algorithm. There are three ways that the data is labeled in this work. This is according to the three levels of categories discussed previously. The first way is simple in that it only uses "1" for normal and "2" for abnormal. It is important to be able to know whether the system is experiencing abnormal behavior since the possibility that such behavior is destructive exists, regardless of how it occurred. The second way of labeling the data is to use "1" for normal, "2" for fault, and"3" for attack to have three classes. For this second way to classify the situations experienced by the system, attacks and faults are two separate classes in addition to normal behavior. It is desired that a distinction can be made between faults and attacks based on the features fed into the classifier. It should be noted that the signature of the attacks and faults as represented by the features given to the classifier must be statistically different from one another to allow for a diagnosis. It is also possible to break down each of the categories on the second level (normal, fault, attack) into subcategories. These subcategories would be on the third level of the tree. Once the classifier is trained, the test set of the data can be used. It is important to understand how the results will be generated, in the case of the experiments. It is also important to understand these results for the case that there is an actual implementation. The results of the testing class may be considered as a form of status update in which the user of the system is told the most likely situation that the system appears to be experiencing. The most important piece of information that the user is told is whether there is abnormal behavior occurring in the system.

The next important thing that the user must know is whether the behavior is due to a fault or an attack. What is important for the features is that the most important characteristics of the system are captured such that different scenarios would be seen as statistically different from the others. In order to determine what features should be included; it is useful to gain an intuitive sense of the behavior of the system. This can be achieved through observing the results of the simulations. As was mentioned, because there is an enormous amount of information produced by the simulations, a subset must be extracted from these simulations, or a small set of values must be calculated from the data. For the features, it would make sense to include data that would be most useful in determining that there is an abnormality in the system. One useful piece of information from the system would be the maximum deviation of the sensor values. The deviation is determined by the anomaly detection module in the controller. It would make sense to divide the simulation into "windows." In the case of this project, the simulation is divided into 10 windows. The maximum can be taken as a feature from each window. The following is a figure to illustrate this:

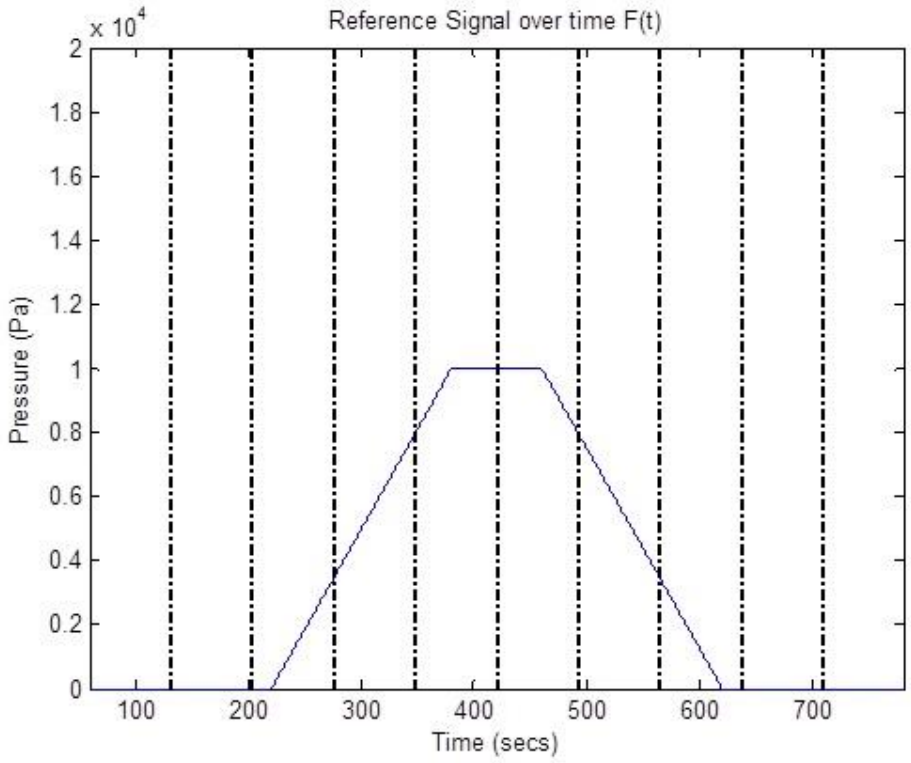


**Figure 3.10: Plot of the Reference Signal with the Simulations Divided into Windows**

Suppose that m features are extracted from each "window" in a given simulation. Windows in this work will be defined as equal time intervals of the simulation. If there are n windows then m*n features can be fed into the classifier. This set of futures that are derived from the simulation data can be considered a compressed version of that data, where the features represent the most important or useful characteristics of that data. It may be useful to see one simulation as one cycle of some pattern of behavior for this system, especially if there is a time-varying reference signal. This signal may be periodic. The information generated by the simulations is from the perspective of the controller. The reason that this setup was chosen is that a similar setup was chosen in the paper [2]. In this paper, there was an anomaly detector, which is similar to what is done in this work. The anomaly detector in that work incorporated a mathematical model of the plant in the controller which was simulated. This was done so that a comparison could be made between the simulated model's sensor readings and the sensor readings from the datagrams that were sent over the network. Of course, the controller would also have to feed the actuator commands into its simulation model that it is sending the actual plant. This difference between the ideal behavior and the behavior as indicated by the sensors could be recorded by the controller as a function of time, not a continuous function of time but a function of discrete time. This alone may not be perfect for analyses. It may be necessary to filter out noise from this. But it is important to realize that this difference gives a symptom for abnormality, which alone would not always allow for the distinction to be made between faults and attacks since both faults and attacks can cause similar deviation. A larger difference is more likely to indicate an abnormal situation than a smaller difference which may be seen as being within some noise margin. As was mentioned, other pieces of data are also needed. Such data would have to allow for a distinction to be made between a fault and an attack. This type of data that would allow for this distinction would most likely be information associated with the network. It may not be possible in all cases to use information from the network, but in some cases it certainly can. This information includes the inter-arrival time of packets received by the controller, and also the traffic on the network.

# CHAPTER IV

# RESULTS

In this chapter, the results of the experiments are presented and described so that the effects of various situations on the NCS can be understood and conclusions can be drawn as to how these different situations can be distinguished. There are two main parts to this chapter:

In the first part, a subset of the scenarios, which are described in the previous chapter, is studied. The subset contains scenarios that fall under all of the low-level categories of the scenarios. These scenarios are studied by observing plotted data from the simulations to gain insight into the various behaviors of the NCS due to the effects of various operating conditions that the NCS is subject to. Some of this insight is also gained through knowledge of control systems, networks, etc. It is desired to understand the various signatures or symptoms that might be associated with these conditions, or specifically how the conditions manifest themselves to a supervisory system. This can be seen in the data or lack of data that this system receives. In the case of this project, this supervisory system would simply be the device which has the control algorithm implemented in it, along with the ADM, which was discussed in the previous chapter.

In the second part, the data generated from all the scenarios were processed, and features were extracted from the data to train a naïve Bayes classifier. The features were chosen heuristically such that they capture the important aspects of the data. It is important to capture certain aspects of the data such that a distinction can be made between scenarios of different categories. The term "categories" refers to the way the scenarios are divided according to Chapter 3. In order to train the supervised learning algorithm, each scenario is labeled based on the categories that the scenario falls under. There are three main ways that the scenarios are labeled, and this is done according to the hierarchy of categories. The goal for this part of the experiments is to assess how well a distinction can be made between the various types of circumstances that the networked control system may experience using a well known machine learning algorithm.

The results of these two parts of the experiments are shown in this chapter. Also, a discussion and an analysis are included.

## Part I: Behavior of the Networked Control System Subject to Various Circumstances

As was mentioned, it is desired to study some of the simulated scenarios to gain an understanding of their various behaviors. Plots were created in order to observe this behavior graphically. What follows are the plots to illustrate all 8 of the low-level categories (The normal case and two selected scenarios from each of the seven abnormal low-level categories). They were chosen such that all the abnormal cases have the same start time and end time for when the abnormal situation occurs. This was done so that a comparison could be made between attacks and faults as far as their behavior in the physical system is concerned. The plots of the scenarios are broken down as follows: First, the normal scenario's plots are shown and described. Then two sets of scenarios of the abnormal low-level categories are discussed with a set of plots. The first set of these abnormal scenarios have the start time for the onset of the abnormal circumstances at 348 sec. The second set of these abnormal scenarios has the start time at 492 sec. These times were chosen because they occur when the reference signal is sloping upward and downward respectively. The pressure in Tank 2 must track the reference signal according to the control objective. For this part of the results, it should be noted that there are certain variables that are studied which are associated with the NCS. These variables are as follows and are indicative of the behavior of the network: (1) The Actual Behavior is the true behavior of the pressures of the tanks over time. (2) The perceived behavior is the behavior that the supervisory system observes from the sensor data that this supervisory system receives. (3) Deviation is defined as the difference between the perceived behavior and the nominal behavior. (4) Nominal behavior is defined as the simulated behavior for the ideal plant. This simulation is performed in the ADM of the supervisory system and serves as a basis for comparison.


**Figure 4.1: Legend for the Following Plots of the Data Associated with Tanks 1 and 2 [(a) and (b) respectively.]**

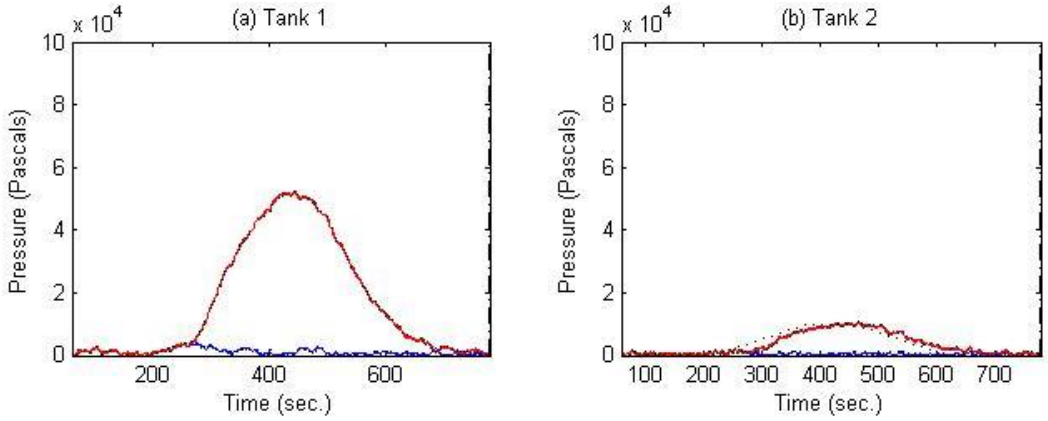


**Figure 4.2: Simulation 1 - Normal Operating Conditions**

Above is the normal scenario (Figure 4.2). On the left (a) and on the right (b) are plots of the actual pressures of Tanks 1 and 2 over time respectively (black line in plots (a) and (b)). It should be noted that these pressures can be considered the "real" pressures of the system. In other words, the data used is taken directly from the plant so that the true behavior of the system can be understood. It should be noted that the controller does not see this data directly, but only sees the data that it receives, which is "the perceived behavior" (red line in plots (a) and (b)). Also, the plots show the deviations for Tank 1 and 2 (blue line in plots (a) and (b)). The data for these two previously mentioned plots are data that the controller actually has or calculates based on the data that in receives. The deviation is defined as the difference between the perceived behavior of Tank 1(2) and the nominal behavior, which is simulated in the ADM. This simulation within the controller is implemented in the ADM, as described in the experimental setup in Chapter 3. Because the deviations are relatively small (Deviation < 3% of maximum value used for pressure in the simulations) for both tanks, they appear as noise in the plots. Also, the fact that the deviations are insignificant in magnitude is a sign that there are no abnormal circumstances present. If, however, the data points of the plots for the deviations take positive values that are significantly greater than zero beyond any reasonable noise margin, then that would be an indication of abnormal behavior. Also the network behavior for these simulations remains constant throughout, meaning that the network has no abnormalities. This is an

indication of no network faults and no cyber-attacks. The network behavior can be characterized by its network utilization over time, which is steady for the normal case as would be expected for a networked control system as mentioned previously. Also, whether the controller is updated or not is also important information for characterizing the network behavior.

As was mentioned, a set of abnormal scenarios were selected from the 66 unique scenarios such that the onset of the abnormal situations begins at 348 sec. What follows are the plots that have been generated from the simulations of these scenarios. Also, these scenarios are discussed. After this set, an additional set is plotted and discuss where the onset of the abnormal scenarios begins at 492 sec.

## First Set of Selected Scenarios (Onset = 348 sec): Faults – Simulations 4, 9, 14, 19


**Figure 4.3: Legend for the Following Plots of the Data Associated with Tanks 1 and 2 [(a) and (b) respectively.]**

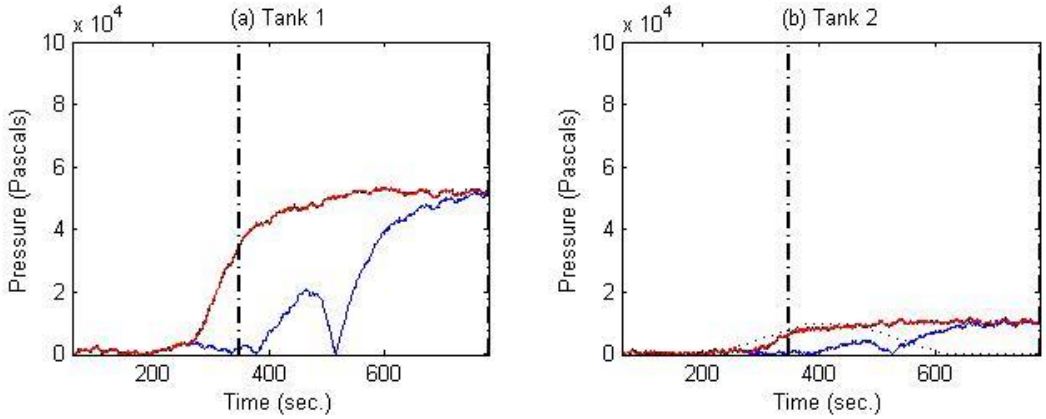


**Figure 4.4: Simulations 4 - Actuator Stuck (Onset of Fault = 348 sec.)**

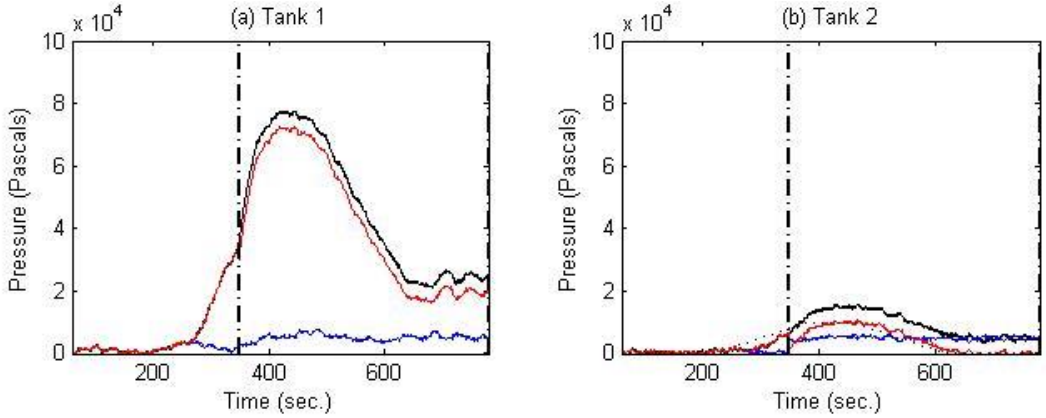


**Figure 4.5: Simulation 9 - Small Sensor Bias (Onset of Fault = 348 sec.)**

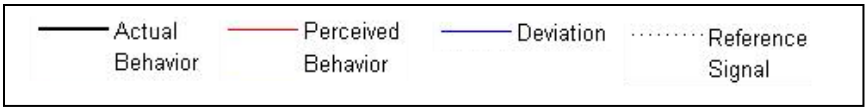


**Figure 4.6: Legend for the Following Plots of the Data Associated with Tanks 1 and 2 [(a) and (b) respectively].**

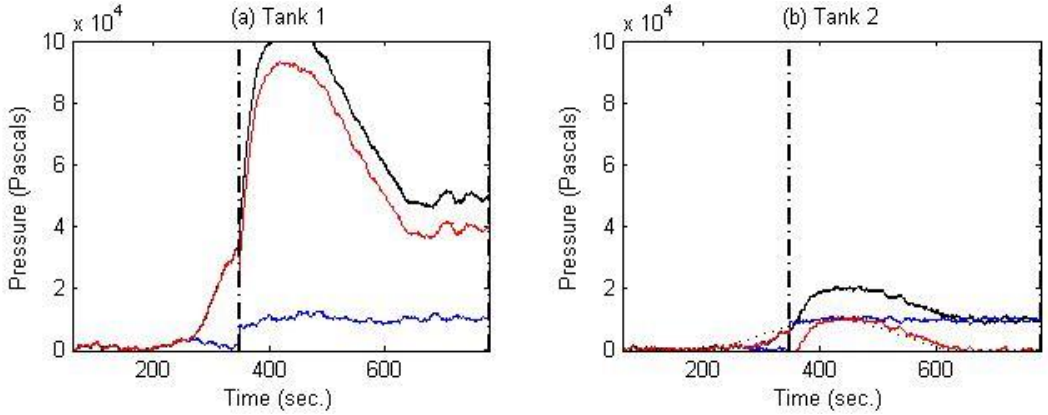


**Figure 4.7: Simulation 14 - Big Sensor Bias (Onset of Fault = 348 sec.)**

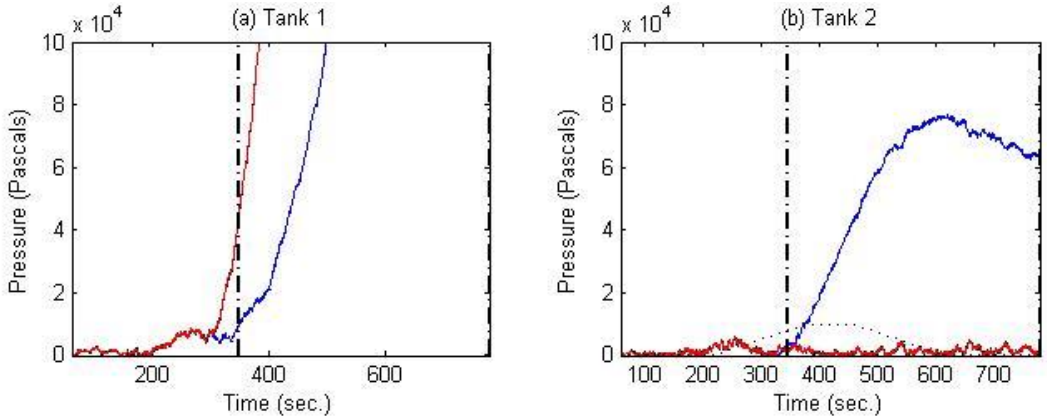


**Figure 4.8: Simulation 19 – Damaged Plant (Onset of Fault = 348 sec.)**

**Faults for the First Set:**

These faults have a few things in common as far as their symptoms are concerned: (1) The network behavior is essentially the same as the normal case discussed previously. This is to be expected since the network is not manipulated or interfered with in any way. (2) There is abnormal behavior in the physical system as seen in the deviation, which is a measure of how the perceived behavior differs from what is expected according to the simulated behavior in the ADM. What should be noted is that the behavior of the deviation or the shape of its graph may vary for different faults.

**Simulation 4: Actuator Stuck:** In this scenario, (figure 4.4) where the actuator becomes stuck, the valve is no longer responsive and remains in the open position for the rest of the simulation. Therefore, water will flow into tank 1 and continue to do so during the operation of the NCS as seen from the actual behavior (black line (a) and (b)) and the perceived behavior (red line (a) and (b)). Water will also flow from Tank 1 and Tank 2. It should be noted that once the inflow from tank 1 to tank 2 equals the outflow, then the pressure in tank 2 will approach a specific constant. Because of this, the behavior diverges from the nominal behavior simulated in the ADM. This is evidenced by the fact that the deviation takes on positive values. At some points, the deviation is equal to zero. This is because the simulated behavior happens to be equal to the perceived behavior at some points in time. This is certainly not an indication that the behaviors are the same at those points since these behaviors are changing over time. In other words, they are dynamic in nature and not static. Therefore, it is important to understand this behavior over the course of time rather than at a single point only. The change of deviation with respect to time may also be determined so that the dynamic nature of the behavior can be captured. This is important for a machine learning algorithm since this information concerning the change of behavior can be used as a feature for this algorithm. This will be discussed in Part II of the results. It should also be noted that the network behavior is essentially no different than the normal case, meaning that the controller received updated packets and the network traffic is fairly constant.

**Simulations 9, 14: Sensor Bias (Small and Large):** These two scenarios are essentially the same in nature, but one has the effect of the fault greater than the other so that there can be variation in the experiments. For simulation 9 (the small bias), there is a bias that causes the sensor values in the packets to be 5000 Pa (Pascal) lower than they actually are. The plots of the deviation show these biases on the sensors. The deviation increased from being merely noise to having a higher value once the fault occurred. It also remained at the higher level. This abnormality can be detected by the ADM in the controller, which calculates the deviation. For simulation 14 (the large bias seen in figure 4.7), the bias causes pressure of the water to appear 10000 PA lower than they actually are, and is similar to the previous scenario. The plots of the deviations show this. Like the last scenario, the values are larger than they ought to be.

For both of these scenarios, the controller responds to the values that it receives from the sensors by taking the control action that would seem to be appropriate based on those values. In reality, the pressure is higher than what the sensors indicate. Therefore, when the controller receives the incorrect values for the pressure from the network, it will respond such that those lower values will track the reference signal. As a result, the actual pressure will not track the reference signal but instead will be higher. The plots of tank 1 and 2 illustrate this behavior.

**Simulation 19: Damaged Plant: For this simulation:** The pipe is broken (figure 4.8). Therefore, there is a leak in the plant, meaning water cannot move from tank 1 to tank 2. As a result, the second tank does not have pressure that can track the reference signal. Tank 1 is constantly being filled, but it does not affect tank 2. If tank 2 has any water in it, then the water in tank 2 will simply be emptied out once the pipe has become broken. The controller is designed such that it will continue to send commands via the network to the actuator so that the valve is open because this action in normal circumstances would cause the pressure in tank 2 to track the reference signal. But as the water goes into tank 1, it can never reach tank 2. Therefore tank 2 is not able to track with the reference signal of the controller. Tank 1 will continue to be filled with water as a result.

## First Set of Selected Scenarios (Onset = 348 sec): Attacks – Simulations 33, 48, 63

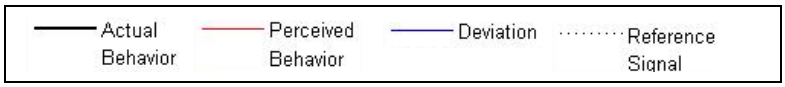


**Figure 4.9: Legend for the Following Plots of the Data Associated with Tanks 1 [(a) and (b) respectively].**

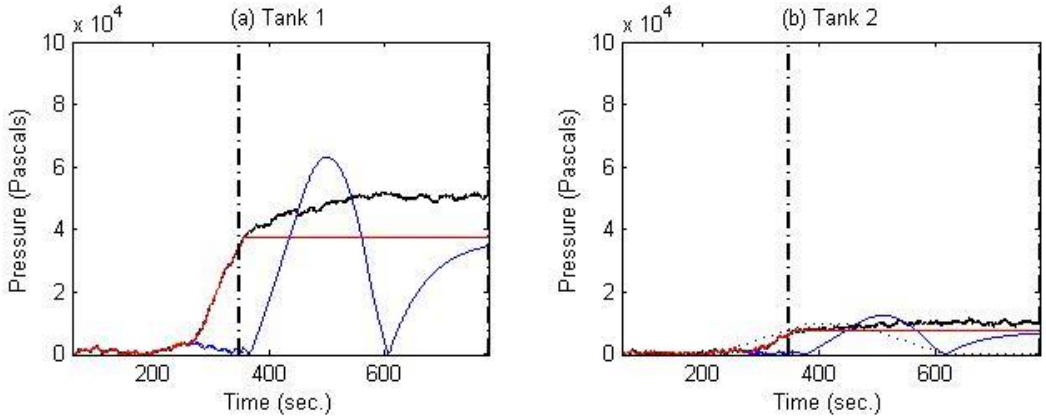


**Figure 4.10: Simulation 33 – DoS Flooding Attack (Onset of Fault = 348 sec)**

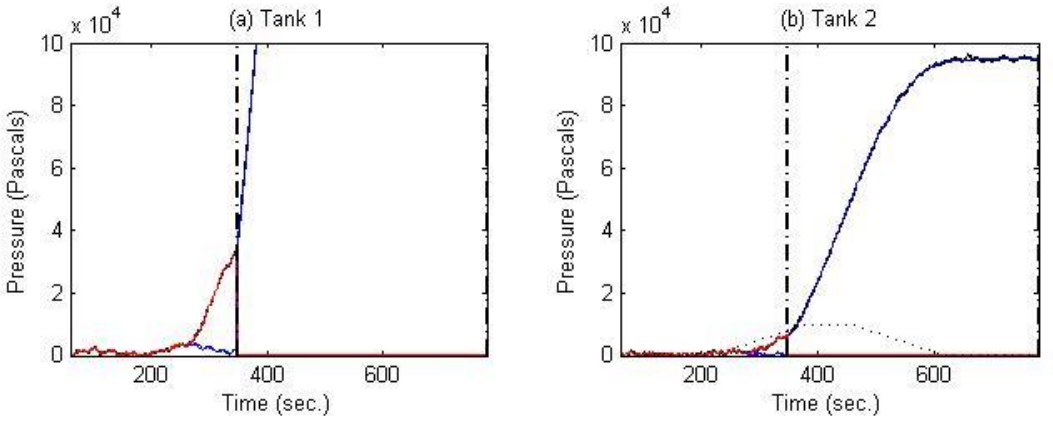


**Figure 4.11: Simulation 48 – Injection Attack on the Controller (Onset of Fault = 348 sec.)**

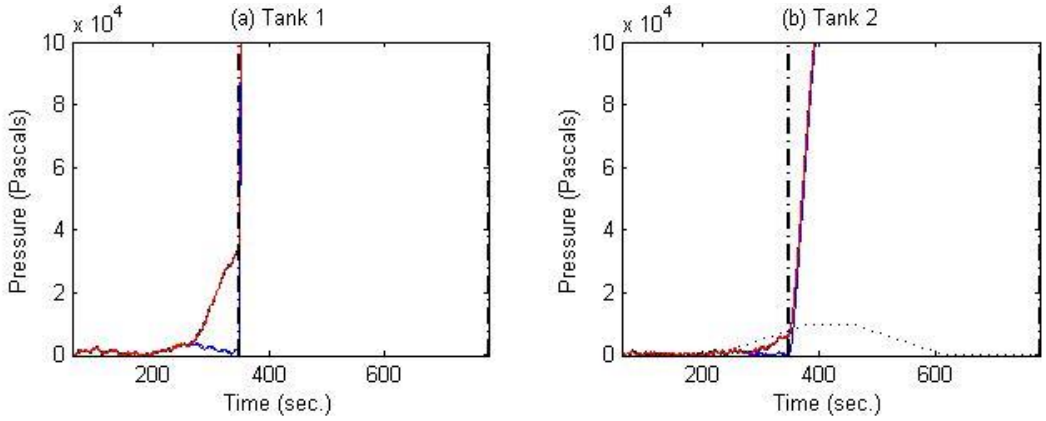


**Figure 4.12: Simulation 63 – Injection Attack on the Plant (Onset of Fault = 346 sec.)**

**Attacks in the First Set**

Cyber-attacks may influence the network in order to carry out their objective. Therefore, it is important to not only observe the deviation over time, but also the network behavior in terms of its network utilization and whether the controller is updated with a new datagram that is correctly formatted. For each of the discussions of the simulations, the network behavior will be described.

**Simulation 33: DoS Flooding Attack:** For this attack (Figure 4.10), after the valve is open, the attack is started. The set-point is higher than the pressure at that point as seen by the controller. The controller will continue to hold that sensor reading until it is updated. Therefore, the controller responds by sending commands to open the valves of the controller and keep them open. The valve will remain open until it receives another command to change its state. But during the DoS attack, communication stops for the duration of the attack. It should also be noted that the behavior behaves abnormally according to the deviations for the two tanks. What is also notable is that the controller is not updated with a new datagram from sensors of the plant during the duration of the attack. This can be seen by the observed network behavior in that the controller receives no updated packets with new sensor readings during the period of the DoS attack, which in the figure will be between the vertical dash-dot lines in figure 4.10. During the attack the network utilization goes to 0%

**Simulation 48: Injection attack on the Controller:** For this simulation, datagrams with false values for the sensor readings are sent to the controller (Figure 4.11). These values are deceptively low, so that the controller would respond by sending a command to open the valves and add more water to the tanks. The network activity as seen from the controller becomes higher than usual because of the increased datagrams in the network. In order for the attacker to be successful, there would have to be a large number of datagrams in the network in order to overwhelm the network preventing the legitimate data from being sent to the controller. Observing the network activity would aid in distinguishing this type of attack from certain types

of faults, such as the sensor bias, which may have a signature similar to this attack as far as the physical system is concerned.

With regard to the network behavior, a few things should be noted: (1) The controller receives a datagram which will update the controller via the network. It is possible that many of these datagrams could be from a malicious entity. (2) The network behavior has a much greater network utilization during the time of the attack. When the attack does occur the utilization is 58% as opposed to 1.6% when the attacks do not occur.

**Simulation 63: Injection attack on the Plant:** For this set of experiments, commands from a malicious entity are sent to the plant (4.12). The packets from the malicious entity contain commands that cause the valve to be open, which may be contrary to the commands that the controller would send. The packets coming from the malicious entity are sent at such a high frequency that they overwhelm any legitimate packets from the controller. Therefore, tank 1 is constantly filled with water.

As far as the network behavior is concerned, the controller is constantly updated with datagrams. The network utilization also is different for when the attack takes place (33.5% as opposed to 1.6% in the normal case) (This occurs between the dash-dot lines).

## Second Set of Selected Scenarios (Onset = 492 sec): Faults – Simulations 5, 10, 15, 20


**Figure 4.13: Legend for the Following Plots of the Data Associated with Tanks 1 and 2 [(a) and (b) respectively].**

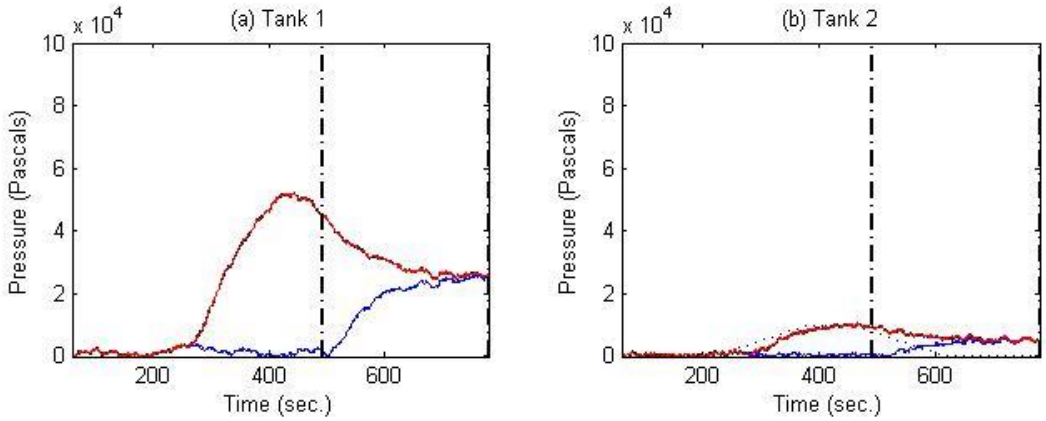


**Figure 4.14: Simulations 5 - Actuator Stuck (Onset of Fault = 492 sec.)**

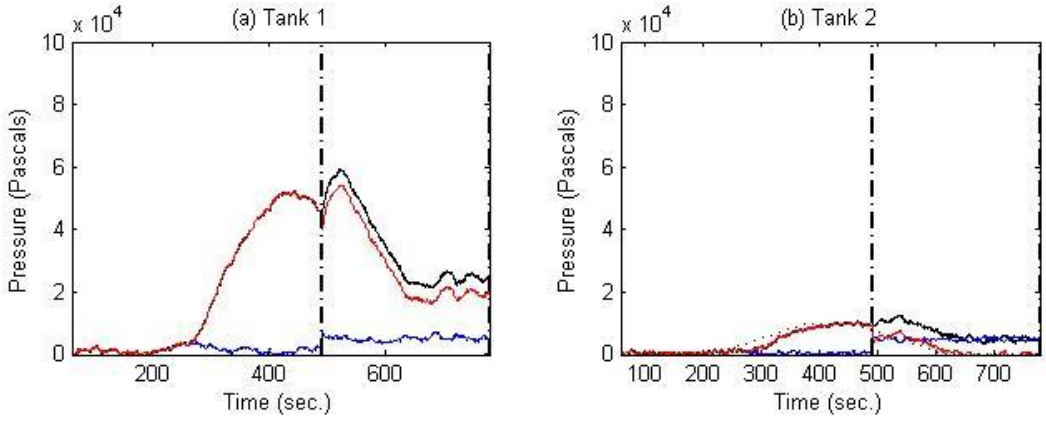


**Figure 4.15: Simulation 10 - Small Sensor Bias (Onset of Fault = 492 sec.)**


**Figure 4.16: Legend for the Following Plots of the Data Associated with Tanks 1 and 2 [(a) and (b) respectively.]**

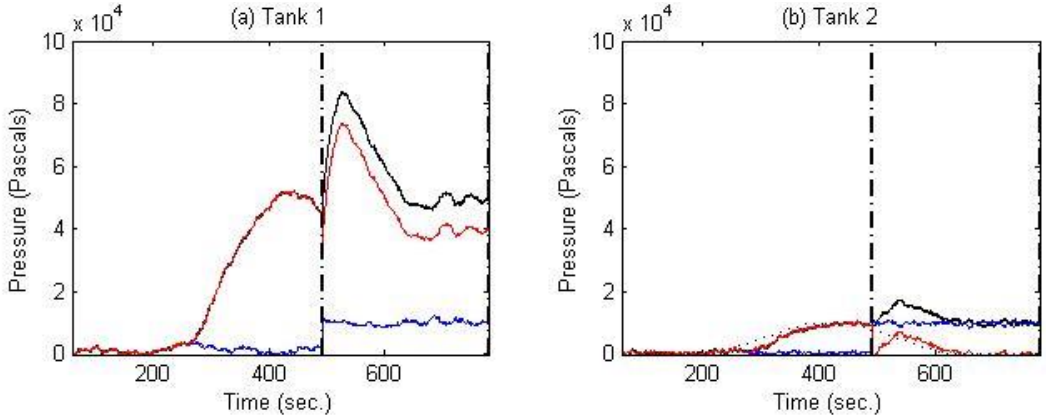


**Figure 4.17: Simulation 15 - Big Sensor Bias (Onset of Fault = 492 sec.)**

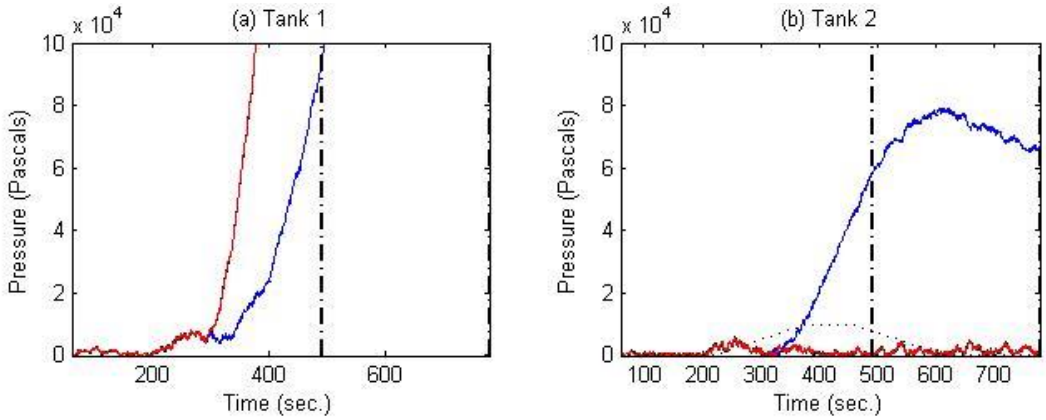


**Figure 4.18: Simulation 20 – Damaged Plant (Onset of Fault = 492 sec.)**

**Faults in the Second Set**

This is similar to the first set in that the network activity has no abnormalities, but the physical system does have anomalous behavior according to the plot of deviation, which takes on positive values that are higher than a mere noise level. The shape of the deviation may be unique for different faults.

**Simulation 5: Actuator Stuck:** The plots for this simulation (Figure 4.14) show the actuator becoming stuck. Therefore, the flow of water going into the tank remains a constant because it does respond to new commands. There will be a certain point in which the pressure in the tanks converges such that the inflow equals the outflow for the water.

**Simulations 10, 15: Sensor bias (small and large):** For simulation of the small sensor bias (Figure 4.15), the effects of a sensor bias fault are observed. It can be seen that the pressure becomes higher than it is supposed to. This simulation of the large sensor bias (Figure 4.17) is similar to the previous scenarios but the effect is greater. As far as the physical system is concerned when compared with that of the previous set, the difference is that the deviation is simply delayed more for second for when it abruptly changes to a higher value.

**Simulation 20: Damaged Plant:** The simulation (Figure 4.18) as seen by the graphs is similar to the damaged plant when there was an upward slope in the reference signal. Between this set and the first set, the behavior for the damaged plant is similar for the physical system.

## Second Set: The Attacks (Onset of attacks = 492 sec.)–Simulations 30, 45, 60

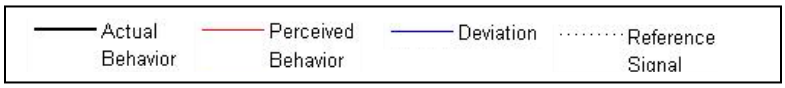


**Figure 4.19: Legend for the Following Plots of the Data Associated with Tanks 1 [(a) and (b) respectively.]**

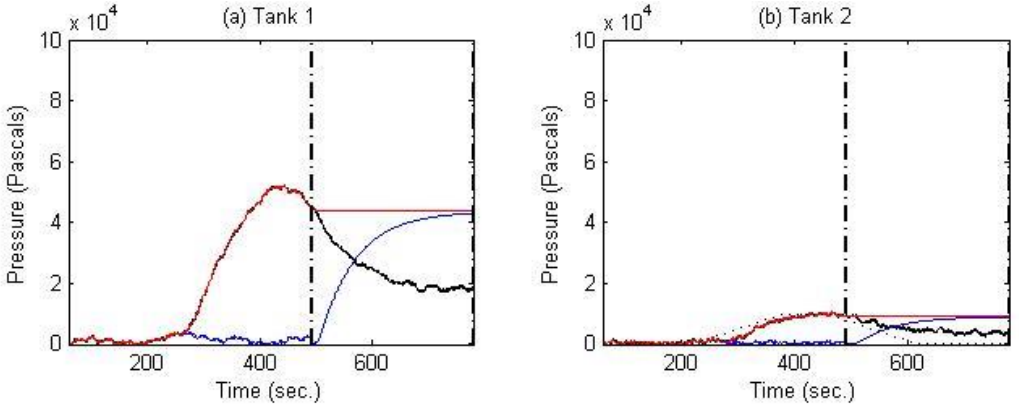


**Figure 4.20: Simulation 30 – DoS Flooding Attack (Onset of Fault = 492 sec.)**

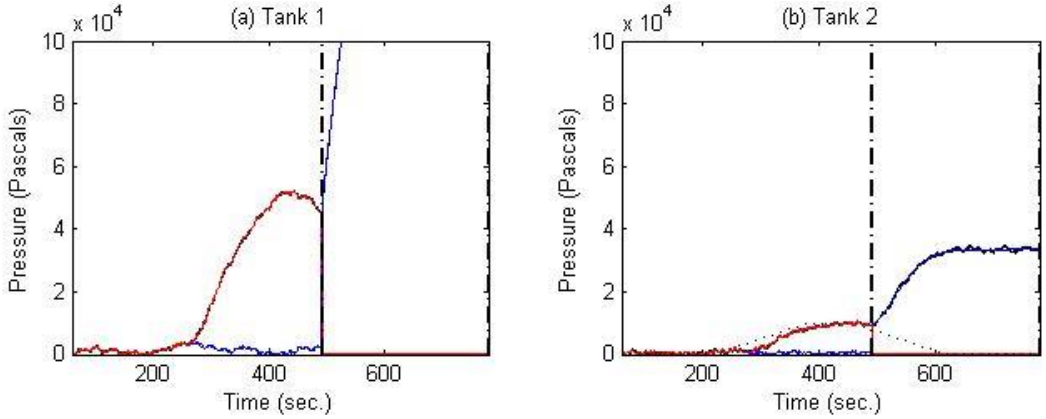


**Figure 4.21: Simulation 45 – Injection Attack on the Controller (Onset of Fault = 492 sec.)**

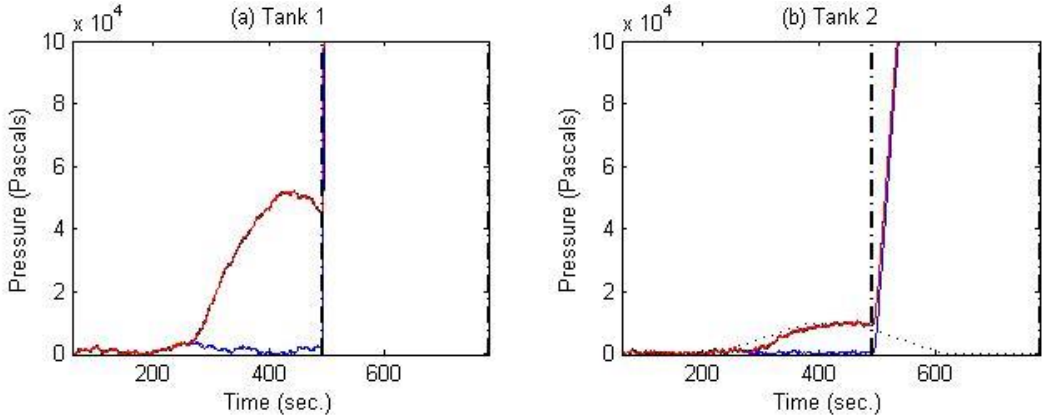


**Figure 4.22: Simulation 60 – Injection Attack on the Plant (Onset of Fault = 492 sec.)**

**Attacks in the Second Set**

For the above attacks, like the previous set, the scenarios have various abnormal behavior as seen in the deviation plot. Also, because they use the network to achieve their goal, the attacks have behavior that can be seen on the network.

**Simulation 30: DoS Flooding Attack**: The DoS flooding attack (Figure 4.20) is practically the same as the actuator becoming stuck in the way that it manifests itself in the physical system. When the DoS attack occurs, no new information is sent to the actuator. This means that the actuator remains in the same position according to the last command it receives from the controller. Also, no new information is sent to the controller, which is different than the actuator becoming unresponsive.

As far as the network behavior is concerned, the controller is not updated during the time of the DoS attack, which is similar to the previous set. Therefore because the controller is not updated it will use the last known sensor value [1]. The network utilization goes to 0%.

**Simulation 45: Injection Attack on the Controller**: The scenario (Figure 4.21) shows the effects of an injection attack on the controller that involves sending deceptive data that indicates to the controller that there is no pressure in tank 2. As the attack occurs, the controller responds to the false data by opening the valves. Once the attack occurs, the sensor values of the datagrams sent by the malicious entity to the controller are at 0.

As for the network behavior, it should be noted that the controller is also updated and the network utilization is at a higher level during the period of the attack (network utilization = 59%). The physical system behavior is similar the previous set.

**Simulation 60: Injection Attack on the Plant**: The injection attack on the plant causes the valves to be open (Figure 4.22). Therefore, the pressures in the tanks will continue to increase. The controller will still receive datagrams from the plant. Therefore, the perceived behavior will essentially be the same as the actual behavior.

Similar to the attack on the controller, the attack on the plant has certain network behavior such as the following: (1) updated datagrams throughout the simulation. (2) Increased

network utilization during the time of the attack. The network utilization becomes 24% during the attack.

**Discussion of the Abnormal Sets**

After observing these plots, the similarities and differences of these scenarios should be noted. For the abnormal cases, there are similarities in the physical system for many of these cases. For instance, these scenarios do have deviations associated with them. It is true that the way these deviations may appear are different as functions of time, but it may also be possible that these abnormal scenarios can be constructed so that they are closer in appearance. For some abnormal scenarios, it is more difficult to distinguish the different abnormal situations from one another only using the information gained from the physical system. These abnormal situations include attacks and faults. For such cases more information must be included. The best example of this is seen in the DoS attacks (figures 4.10 and 4.20) and the unresponsive actuator (figures 4.4 and 4.14), which are essentially identical if information from the physical system is all that is observed. It should be noted that these plots have deviations that are similar in shape, although one is larger than the other. There are some differences between these two scenarios as well. One difference is that the sensor value that the controller uses from the onset of the abnormality to the end of the simulation remains constant. Using information from the network, the controller sees that it is not updated because of the lack of legitimate packets that it receives. On the other hand, for the unresponsive actuator, it is the state of the valve that remains constant, and the network behavior will appear to be normal to the controller. Therefore, the different scenarios do not have behavior that is totally unique from the information of the physical system alone. However, they have different behavior of the network that causes them to be unique. Uniqueness is required in order to make a distinction. It should be understood that in fault diagnosis, there is a concept known as insolubility. This refers to the ability to distinguish one fault from another. This is useful in order to be able to locate the fault in the system [20]. This work does not focus on distinguishing one fault from another. Instead, it focuses on distinguishing faults from attacks, based on certain characteristics that they have. The main characteristic that an attack has which differentiates it from a fault is that the network behavior is correlated with the physical behavior that the attack causes. When the plots are compared, it can be seen that there are similarities in the plots for the physical system, but there are some differences in the network behavior. This

network behavior that is observed by the controller includes whether it is updated, the amount of network activity, and the inter-arrival time between packets that are received by the controller.

## Part II: Experiments with Diagnosis

For this work, a set of 66 unique simulations were performed in Omnet++ through the use of a batch script in the command line interface of Omnet++. The script allowed for the automation of multiple simulations. The break-down of these simulations according to the mid-level categories (normal, fault, and attacks) were as follows: Simulation 1 was a simulation with normal operation. Simulations 2-21 were simulations with faults. Simulations 22-66 were simulations with attacks. Features were extracted from the simulations according to what was discussed in the previous chapter. Therefore, there were 66 sets of features. There are a certain number of features per "window". Basically, in this work, a "window is defined as an equal section of the simulation time. In the case of this project there are 10 windows per simulation. Ten features are extracted from each window. Therefore, 100 features are used per simulation. The following is a table of the features of a window extracted for each simulation followed by a description of each (Table 4.1).

| Feature # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Abbreviated name | T1PMax | T2PMax | T1DevM | T1DevM | Dev1Ch | Dev2Ch | ErrorMax | InterTime | Update | NetTraff |

**Table 4.1: Table of the Features for a Given Window**

The descriptions of each of these features are as follows according their feature number:

1. T1PMax: This is the maximum pressure in tank 1 according to the packets received by the controller. Specifically, it is the maximum of a set of values for pressure that the controller receives for a given window.
2. T2PMax: This is the same as above except it is for tank 2 instead
3. T1DevM: This is the deviation of tank 1.
4. T2DevM: This is the deviation of tank 2.
5. Dev1Ch: This feature is used to indicate whether there is a change in the deviation for a window compared with the previous window in tank 1. The possible values for this feature are -1, 0, 1. The value of -1 indicates that there is a significant decrease. The

value 0 indicates that there is no change or hardly any change. Basically there is a certain tolerance or range of values. Within this range, the deviation is considered to have no significant change. The value 1 indicates that there is a significant increase.

6. Dev2Ch: This is the same as the one above except it is for tank 2.
7. ErrorMax. This is the maximum error for a given window that is associated with the control system. The error is defined as the difference between the set-point and the measured value. This variable is useful because DoS attacks are particularly destructive when there is a significant error as opposed to when the system is in steady state.
8. InterTime: This is the inter-arrival time between packets. It is specifically defined as the time between two consecutive incoming packets. This time is determined according to the clock of the controller.
9. Update: This is a Boolean variable that indicates that the controller is updated by a legitimate packet from the sensor via the network. It is also possible that the controller is updated by a packet that appears legitimate but is really from a malicious entity. This is a useful feature because it allows for a way to detect whether a DoS attack is occurring. If this Boolean variable has a value of 0, then based on intuition it can be hypothesized that a DoS attack is occurring. Also, although this is not included in this work, if it is a 0, then it may be that there is a network failure. For this case, there would not be network activity in the NCS. Therefore, it can be hypothesized that a distinction can be made between a DoS attack and a network failure.
10. NetTraff: This is a measure of the Network Traffic Activity that the Controller receives from the channel of the network that it is directly connected to. It indicates the amount of traffic on the network from the perspective of the controller. This is measured in bps (Bits per Second).

The 66 scenarios were expanded to 125, by making copies of some of the features with noise added (Faults and Attacks) for several scenarios or using noise variation within experiments (Normal). This was done so that there was a roughly equal number of normal cases, cases with faults, and cases with attacks. This could be done in this work because it was not known exactly how often one type of situation occurs as compared to others since there is no known statistical

data that can be used to make this determination. The focus of this work is to instead determine how well different abnormal scenarios can be distinguished and classified based on a unique signature. The table below shows how these simulations were expanded (Table 4.2):

| Classification Scheme | | | | |
|---|---|---|---|---|
| High Level | Mid Level | Low Level | Scenario Numbers | Number of Scenarios per low-level category with noise |
| Normal | Normal | Normal | 1 | 40 |
| Abnormal | | | | |
| | Faults | | | |
| | | Damaged Plant | sims 2-6 | 15 |
| | | Small Sensor Bias | sims 7-11 | 15 |
| | | Large Sensor Bias | sims 12-16 | 15 |
| | | Actuator Stuck | sims 17-21 | 15 |
| | Attacks | | | |
| | | DoS Flooding | sims 22-36 | 15 |
| | | Injection to Plant | sims 37-51 | 15 |
| | | Injection to Controller | sims 52-66 | 15 |
| | | **Total Unique Scenarios:** | **66** | 145 (125 used) |

**Table 4.2: Scenarios by Classification Scheme**

One hundred sets of features were given as input for the classifier in order to train it. Each set of features consisted of 100 features. The remaining 25 sets were used for the testing phase.

It is important to note that the labeling of scenarios for the naïve Bayes classifier was done according to all of the three classification schemes, which are basically the three levels of categories described in the previous chapter. MATLAB provides this Naïve Bayes classifier, which can be used by means of MATLAB scripts. The following tables show how the categories for all three levels are labeled for the classifier:

| Label | 1 | 2 |
|---|---|---|
| Class Name | Normal | Abnormal |

**Table 4.3: Labels Used for the First Scheme (High Level)**

| Label | 1 | 2 | 3 |
|---|---|---|---|
| Class Name | Normal | Fault | Attack |

**Table 4.4: Labels used for the Second Scheme (Mid Level):**

| Label | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Class Name | Normal | Actuator Stuck | Sensor Bias(small) | Sensor Bias(large) | Damaged Plant | DoS Attack | Injection Con. | Injection Plant |

**Table 4.5: Labels Used for the Third Scheme (Low Level):**

**Results from using the naïve Bayes classifier:**

After all the necessary CSV files were generated with the data that the features are based on, the machine learning algorithm was executed using a MATLAB script, which generated the results according to the three main schemes of classifying the scenarios. For this set of experiments with the classifier, all 10 features were used to train the classifier:

The following are sets of confusion matrices. Confusion matrices are used to display the data and to summarize the results in concise format. Confusion matrices are useful for representing results from the machine learning algorithm for cases of classifying. The rows are actual classes for the data. The columns indicated what is predicted as a result of the classification of the machine learning algorithm. The numbers in the diagonal indicate correctly classified experiments. Off of the diagonal are the incorrectly classified experiments [22]. The accuracy can be calculated by dividing the correctly classified scenarios by the total numbers of scenarios that are used in the testing phase. It can be defined as the result of dividing the trace of the matrix or the sum of the diagonal elements by the total number of test cases. The result of this is expressed as a percentage. These matrices show the results of using the machine algorithm to classify 25 sets of test data. Each of these sets has 100 features as described previously. The results are according to the three classifications schemes, which are based on the levels in figure 3.6 found in Chapter 3.

## Confusion Matrices for the Results where the Features from Both Physical System and Network are used to train the Classifier

| | | Predicted | |
|---|---|---|---|
| | | Normal | Abnormal |
| Actual | Normal | 5 | 0 |
| | Abnormal | 0 | 20 |

**Table 4.6: Confusion Matrix for the First Scheme of Classifying (Accuracy: 100%)**

| | | Predicted | | |
|---|---|---|---|---|
| | | Normal | Faults | Attacks |
| Actual | Normal | 5 | 0 | 0 |
| | Faults | 0 | 9 | 0 |
| | Attacks | 0 | 1 | 10 |

**Table 4.7: Second Scheme of classifying**

**(Accuracy: 96%; Accuracy for distinguishing faults and attacks (As seen from grayed area in the above table): 95%)**

| | | Normal | Faults | | | | Attacks | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | **5** | 6 | 7 | 8 |
| Normal | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Faults | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 |
| | 5 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| Attacks | 6 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 |
| | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |

**Table 4.8: Third Scheme of classifying**

**(Overall Accuracy: 100%, Accuracy between faults and attacks: 100%)**

In order to calculate the accuracy for the mid-level categories when using the Third Classification scheme, the sub-matrices in the grey area that are separated from each other by the dotted lines are used. All the entries of a given sub-matrix were added together. This was done for each sub-matrix so that the following table could be produced.

| | Faults | Attacks |
|---|---|---|
| Faults | 9 | 0 |
| Attacks | 1 | 10 |

**Table 4.9: Submatrix-Attacks/Faults (Total abnormal scenarios: 19; Total correctly classified abnormal scenarios: 17; Accuracy indistinguishing faults and attacks: 95%)**

The accuracy for each of these matrices was fairly high. Of course, it is useful to have some sort of a comparison. Since there is no other work to compare with for this particular data, it is useful to instead compare these results with the case where the network data that is sent to the controller is excluded from the features fed into the machine learning algorithm. The results for doing this are expressed in the next set of tables. The following three confusion matrices show the results for the Naïve Bayes Classifier when only the first seven features are used:

## Confusion Matrices for the Results where the Features from Both Physical System and Network are used to train the Classifier

| | | Predicted | |
|---|---|---|---|
| | | Normal | Abnormal |
| Actual | Normal | 5 | 0 |
| | Abnnormal | 0 | 20 |

**Table 4.10: First Scheme (Accuracy: 100%)**

| | | Predicted | | |
|---|---|---|---|---|
| | | Normal | Faults | Attacks |
| Actual | Normal | 5 | 0 | 0 |
| | Faults | 0 | 9 | 0 |
| | Attacks | 0 | 3 | 8 |

**Table 4.11: Second Scheme (Overall accuracy: 88% Accuracy in distinguishing attacks from faults: 85%)**

| | | Normal | Faults | | | | Attacks | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | **5** | 6 | 7 | 8 |
| Normal | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Faults | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 |
| | 5 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| Attacks | 6 | 0 | 1 | 0 | 0 | 0 | 2 | 2 | 0 |
| | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |

**Table 4.12: Third Scheme (Overall accuracy: 88% Accuracy for classifying the mid-level categories - attacks and faults: 95%)**

In order to calculate the accuracy for the mid-level categories when using the Third Classification scheme, the sub-matrices in the grey area that are separated from each other by the dotted lines are used. All the entries of a given sub-matrix were added together. This was done for each sub-matrix so that the following table could be produced.

| | Faults | Attacks |
|---|---|---|
| Faults | 9 | 0 |
| Attacks | 1 | 10 |

**Table 4.13: Submatrix-attacks/faults (Total abnormal scenarios: 19; Total correctly classified abnormal scenarios: 17; Accuracy indistinguishing faults and attacks: 95%)**

There appears to be increased accuracy when the faults are broken down into their many types. The reason that there is increased accuracy is that each of the attacks and faults that were simulated had signatures that were somewhat distinct from the others. But when all the attacks were put together as one class, these signatures were obscured to the point that faults and attacks were not as distinguishable.

**Discussion**

From the results, certain experiments were done to calculate the accuracy for distinguishing faults from attacks using the Naïve Bayes Classifier. The results of using all of the features for this classifier are compared with the results of using the features restricted only to the physical system. The accuracy improved when features from the network were included. These results are useful for showing accuracy, but there needs to be a more practical use for this diagnosis. It must be understood that a plant operator or the SCADA system itself would be alerted of an attack or fault. Periodically, there will be a status message of the system sent to the plant operator that would use a machine learning algorithm to classify the behavior of the system. This status message may tell the user whether the system is experiencing normal conditions or abnormal conditions. If the user is told that abnormal conditions are occurring, the problem must be addressed. But it must be addressed appropriately, which means that once the nature of the problem (i.e. whether it is a fault, attack, etc.) is diagnosed; the problem can be

dealt with. For instance, if it is discovered that the problem is an attack, the situation may be dealt with by increasing the security of the facility and use whatever means necessary to eliminate the threat. On the other hand if the problem is a fault, it is important to investigate the plant to determine if must be shutdown. It is also necessary to identify the problem and repair it. Therefore, because these two types of situations are dealt with differently, it is important to be able to distinguish between faults and attacks. In the example of Stuxnet, it would have been helpful for the plant operators to know not only that an abnormal situation is occurring but that a cyber-attack is occurring as opposed to faults or degradations of equipment. In the case of Stuxnet, plant operators kept replacing the centrifuges, but this did not address the true problem. If it can be determined that a cyber-attack is occurring, then it may be more properly addressed. In general, the process of dealing with the various situations based on the information given is illustrated in Figure 4.23. In the figure, a supervisory system periodically analyzes data received from the networked control system. It then diagnoses the situation experienced by the NCS. The box labeled “update” in the figure is the updated diagnosis, which is done at certain intervals that are set according to design specifications.

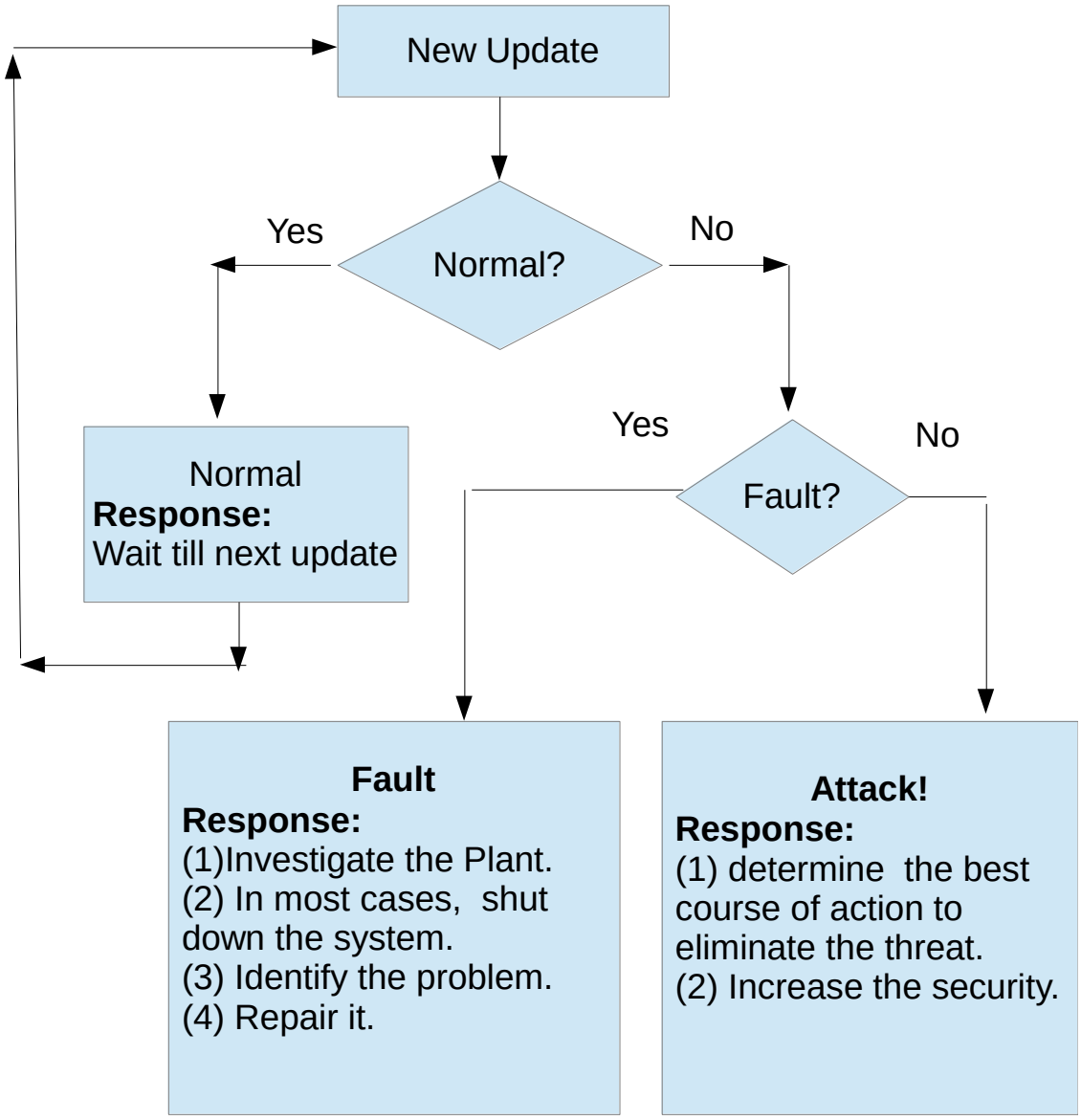


**Figure 4.23: Flow Chart showing responses to be taken based on the updated status of the NCS**

As far as generalizing the problem is considered, it may be useful to understand the systems from a qualitative sense. For instance, a DoS attack may cause a change in the behavior that is different from the nominal behavior, where this change can be detected. Also, the controller detects that it is not updated with new packets. This description for these symptoms may be applicable to other NCS. In addition, the actuator becoming stuck may be detected based on the fact that there is a change in behavior that deviates from the nominal, yet the controller continues to be updated. This qualitative description may be true for other NCS's as well.

# CHAPTER V

# CONCLUSIONS

Detection and response are two important aspects of ensuring security for SCADA systems. One of the aspects of this work that distinguishes this work from others is the fact that it takes into account the possibility of faults in more depth than other work in addition to cyber-attacks. It also explores different scenarios with the major types of abnormal circumstances that a cyber-physical system may experience. It should be noted that there is some work that addressed the issue of faults along with cyber-attacks. This has been done to a limited degree, but in other cases it is done to a greater degree. For instance, the work by Litrico et al. discusses faults [3],[4], but it is not concerned with isolating them from attacks but instead is mainly concerned with finding an anomaly in the system if such an anomaly occurs. The work by Bruno Sinopoli [25], which is focused on integrity attacks also addresses faults and does look at faults and attacks based on their signature at least in the discussion of the simulation results.

For this work, a supervisory system can analyze the data it received from the system. This is done to determine the most likely scenario that the system is experiencing so that the system or the plant operator can make a good decision on what to do with the SCADA system. Being able to respond appropriately to the system would ensure that safety and efficiency can be achieved, or to at least ensure that safety and efficiency can be achieved to some reasonable level with safety being the priority. It is important to note that the different plants for SCADA systems may be quite different in their behavior. Therefore, for every unique model, it may be necessary to simulate that unique model to observe the effects of certain attacks and other types of abnormal circumstances. However, it is possible that the problem can be generalized. In order for this to be done, the symptoms associated with the various anomalies can be understood in a qualitative sense. For instance, it may be possible that other networked control systems will have a similar set of symptoms present for specific types of faults or attacks.

This work explores the basic notion that, given information from the network in addition to information from the plant, which can be seen through the sensor data, it may be possible to better understand the situation that the networked control system is experiencing. With this additional information, it is possible that a better distinction between faults and attacks can be

made. The reason that the information from the network is important is that it is possible that with certain common attacks, the network behavior is somehow correlated with the behavior as seen from the physical system.

The results of the experiments show that the accuracy of the diagnosis can be improved if information can be included from the network in addition to information in the physical system to allow for distinguishing between faults and attacks. Previous work dealing with the security in SCADA systems focused on computer systems and networks alone, or it focusing on the sensor readings from the physical system to detect for an anomaly. One of the goals of this work was to collect information from both the network and the physical system in a NCS. Another goal was to understand symptoms in the NCS due to various circumstances that the NCS has a possibility of experiencing, i.e faults and attacks. It was also the goal of this work to be able to diagnose the given situation of the NCS based on the symptoms as detected by the supervisory system also to know how well various scenarios can be distinguished from the each other.

Future work that could be done would be to investigate other types of attacks that the networked control system can experience. Simulations of these other attacks on the networked control system can be performed to observe their behavior and better understand various symptoms. Therefore, it should be noted that the scenarios used in this work may not have been exhaustive, meaning that all the possible attacks that the system could experience were not included. The scenarios only included attacks where the nodes of the control network were not compromised or reprogrammed. Future work could address the types of attacks where these nodes are compromised to understand their behavior. The focus was on attacks that send packets that influence the network maliciously. These attacks include injection attacks and DoS attacks. It may also be necessary to investigate other pattern recognition and machine learning techniques to experiment with other methods of diagnosis or classification. The main goal of this work was to investigate some of the well-known attacks, and assess how well they can be distinguished from common faults that could occur. One goal for future work could be to devise a new type of intrusion detection system that is more specific to SCADA systems. It was of the aims of this work to move in the direction of doing so by understanding the symptoms of various circumstances and determining how well they can be distinguished with the goal of being able to diagnose the system to some reasonable degree.

## REFERENCES


[1] Yu-Hu Huang, Alvaro A. Cardenas, et al, Understanding the Physical and Economic Consequences of Attacks on Control Systems, Elsevier, International Journal of Critical Infrastructure Protection 2009.

[2] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang and S. Sastry "Attacks against process control systems:Risk assessment, detection, response" *Proc.6th ACM Symp. Inf., Comput. Commun. Security, pp. 355-366, 2011*

[3] Amin, S.; Litrico, X.; Sastry, S.; Bayen, A. M., Cyber Security of Water SCADA Systems-Part I: Analysis and Experimentation of Stealthy Deception Attacks, IEEE Transactions on Control Systems Technology, 2012.

[4] S. Amin, X. Litrico, S. Sastry, and A. Bayen, "Cyber security of water SCADA systems: II attack detection using an enhanced hydrodynamic model," IEEE Trans. Control Syst. Technol., 2012 doi:10.1109/TCST.2012.2211874.

[5] R. Chabukswar, B. Sinopoli, G. Karsai, A. Giani, H. Neema, and A. Davis. Simulation of network attacks on scada systems. In Proc. First Workshop on Secure Control Systems, Cyber Physical Systems Week 2010, 2010.

[6] "Chapter 1: An Overview of Computer Secuirty." William Stallings, *Cryptography and Network Security: Principles and Practice*, Prentice Hall, 2011

[7] Knapp, Eric, and Joel Langill. *Industrial network security: securing critical infrastructure networks for smart grid, scada, and other industrial control systems*. Access Online via Elsevier, 2011.

[8] Security Focus. Slammer Worm Crashed Ohio nuke network. http://www.securityfocus.com/news/6767

[9] A. Daneels and W. Salter, "What is SCADA?," Proc. of Int. Conf. on Accelerator and Large Experimental Physics Control Systems, pp. 339--343, 1999.

[10] Gordon R. Clarke, Deon Reynders, Edwin Wright, *Practical modern SCADA protocols: DNP3, 60870.5 and related systems* Newnes, 2004 ISBN 0-7506-5799-5 pages 19-21[Check]

[11] Bequette BW: Process Control: Modeling, Design, and Simulation. Upper Saddle River, NJ: Prentice-Hall, 2003

[12] Zhou, Jie. Fault Detection and Isolation of a Three-tank System. Diss. Vanderbilt University, 2002.

[13] Janos J. Gertler, Fault Detection and Diagnosis in Engineering Systems, Marcel Dekker, New York, 1998

[14] Seborg, Dale, Thomas F. Edgar, and Duncan Mellichamp. *Process dynamics & control*. Wiley. com, 2006.

[15] Larry L. Peterson , Bruce S. Davie, Computer Networks: A Systems Approach, 3rd Edition, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2003

[16] Nakhaeizadeh, Gholamreza, and C. C. Taylor. *Machine Learning and Statistics: The Interface*. New York: Wiley, 1997.

[17] Omnet. www.omnetpp.org

[18] INET. Inet.omnetpp.org

[19] Davis, Andrew. *Developing SCADA simulations with c2windtunnel*. Diss. Vanderbilt University, 2011.

[20] Khazan, Golriz, and Mohammad Abdollahi Azgomi. "A distributed attack simulation for quantitative security evaluation using SimEvents." *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*. IEEE, 2009.

[21] Alvero A. Cardinas. Research challenge for the security of control systems

[22] James P. Farwell and Rafal Rohozinski. Stuxnet and the future of cyber war. In Survival, volume 53 of 1, page 23 – 40, January 2011

[23] J. Verba and M. Milvich "Idaho National Laboratory Supervisory Control and Data Acquisition Intrusion Detection System (SCADA IDS)", *Proc. IEEE Conf. on Technol. Homeland Security*, pp.469 -473 2008

[24] "Chapter 20: Intrusion" William Stallings, *Cryptography and Network Security: Principles and Practice*, Prentice Hall, 2011

[25] Bruno Sinopoli. "Detecting Integrity Attacks on SCADA Systems". Talk or presentation, 10, October, 2013.

[26] D. Yang, A. Usynin, J. W. Hines, "Anomaly-Based Intrusion Detection. Instrumentation, Control and Human Machine Interface Technologies (NPIC&HMIT 05) , Albuquerque, NM, Nov 12-16, 2006. for SCADA Systems", 5th Intl. Topical Meeting on Nuclear Plant

[27] T. Fleury, H. Khurana, and V. Welch. Towards A Taxonomy Of Attacks Against Energy Control Systems. Critical Infrastructure Protection II, The International Federation for Information Processing, 290, 2009.

[28] B. Zhu and S. Sastry. SCADA-specific intrusion detection/prevention systems: a survey and taxonomy. In Proceedings of the 1st Workshop on Secure Control Systems (SCS), 2010.

[29] Ravi Akella, Han Tang, Bruce M. McMillin, Analysis of information flow security in cyber–physical systems, International Journal of Critical Infrastructure Protection, Volume 3, Issues 3–4, December 2010, Pages 157-173, ISSN 1874-5482

[30] A Lee, Edward A. Cyber Physical Systems: Design Challenges. EECS Department, University of California, Berkeley 2008, 8 January 23

[31] N. Falliere, L. O Murchu, and E. Chien. W32.Stuxnet dossier version 1.3, Nov. 2010. Online: http://www.symantec.com/content/en/ us/enterprise/media/security_response/ whitepapers/w32_stuxnet_dossier.pdf.

[32] "Chapter 1: An overview of Computer Secuirty" William Stallings, *Cryptography and Network Security: Principles and Practice*, Prentice Hall, 2011

[33] Dorf, Richard C., and Robert H. Bishop. *Modern control systems*. Pearson, 2011.

[34] B. Singer, Correlating Risk Events and Process Trends Proceedings of the SCADA Secuirty Scientific Symposium (S4) Kenexis

[35] Zhu, B., & Sastry, S. (2010). SCADA-specific intrusion detection/prevention systems: a survey and taxonomy. In Proceedings of the 1st Workshop on Secure Control Systems (SCS). Stockholm: Team for Research in Ubiquitous System Technology.

[36] Domingos, Pedro & Michael Pazzani (1997) "On the optimality of the simple Bayesian classifier under zero-one loss". *Machine Learning*, 29:103–137.

[37] Cyberattack. Technopedia. http://www.techopedia.com/definition/24748/cyberattack

[38] MATLAB. http://www.mathworks.com/products/matlab/

[39] TrueTime. http://www.control.lth.se/truetime/

[40] McDonald, M., and G. Conrad. *T. Service and R. Cassidy, Cyber Effects Analysis Using VCSE: Promoting Control System Reliability*. Technical Report SAND2008-5954, Sandia National Laboratories, Albuquerque, New Mexico and Livermore, California, 2008.

[41] Falliere, Nicolas, Liam O. Murchu, and Eric Chien. "W32. stuxnet dossier."*White paper, Symantec Corp., Security Response* (2011).

[42] Distributed Monitoring and Control and Physical System Modeling for a Laboratory Three-Tank System; Jeff Lyons; MS Thesis, Dept. of EECS, Vanderbilt University, April 2004

[43] Matlab Simulink. http://www.mathworks.com/products/simulink/