Component Modeling for SCADA Network Mapping

Ayodeji J. Akande Colin Fidge Ernest Foo

School of Electrical Engineering and Computer Science
Queensland University of Technology Australia,
GPO Box 2434 Brisbane, QLD 4001,
Email: (ayodeji.james.akande, c.fidge, e.foo)@qut.edu.au

Abstract

Supervisory Control and Data Acquisition systems (SCADA) are widely used to control critical infrastructure automatically. Capturing and analyzing packet-level traffic flowing through such a network is an essential requirement for problems such as legacy network mapping and fault detection. Within the framework of captured network traffic, we present a simple modeling technique, which supports the mapping of the SCADA network topology via traffic monitoring. By characterizing atomic network components in terms of their input-output topology and the relationship between their data traffic logs, we show that these modeling primitives have good compositional behaviour, which allows complex networks to be modeled. Finally, the predictions generated by our model are found to be in good agreement with experimentally obtained traffic.

Keywords: Supervisory Control and Data Acquisition (SCADA) Topology, Network mapping, Modeling, Network traffic analysis

1 Introduction

Supervisory Control And Data Acquisition (SCADA) systems are used in industrial environments to manage and control critical infrastructure such as transport systems, telecommunications, power and energy services. These systems were invented to handle communication over a large geographical distance and multiple sites. Previously, they were designed to operate in an isolated environment using proprietary protocols which ensured some level of security by obscurity. However, in recent years, these systems have evolved from being stand-alone networks and are now interconnected with both enterprise networks and the Internet with the use of Transmission Control Protocols (TCP)/Internet Control Protocol (IP) leading to wider networks. Though this has led to easy manageability, and improvements in functionality and productivity, connecting SCADA networks to the Internet and corporate networks exposes the systems to attacks with possible ruinous effects.

SCADA networks have some distinct behaviours when compared to “traditional” Information Technology (IT) networks. SCADA networks are expected to be more stable over time with fewer or no devices added to or frequently leaving the network; and in contrast to a traditional network, SCADA networks support fewer services (Barbosa et al. 2012b). SCADA network has a well-defined structure with a predictable traffic behaviour as a result of its fixed number of network devices, regular network communication patterns and limited number of protocols (Barbosa & Pras 2010).

However, increased connectivity between SCADA and other networks can lead to instability in SCADA network communication behaviour. Nicholson et al. (2012) argued that the main cause of security vulnerabilities in SCADA systems seems to be the increased connectivity and the loss of separation between SCADA and other parts of organizations’ IT infrastructures. It is crucial to have up-to-date knowledge of the communications topology of SCADA networks to ensure stability of the network. This can be achieved via network mapping.

Network mapping is the study of network connectivity. This is very useful to industries with a large dispersed network. Several published works (Besaw et al. 1994, Breitbart et al. 2000, Govindan & Tangmunarunkit 2000) have shown the use of network mapping in the study of Internet structure, detecting shared bottlenecks and detecting network intrusions using various techniques including development of algorithms for discovering the network topology (Breitbart et al. 2000) and use of statistical analysis of network packet characteristics (Bykova et al. 2001). However, this previous research is limited to standard IT infrastructure.

The contribution of our work is a new modeling technique to detect whether or not a SCADA network’s traffic is consistent with its anticipated topology. This aids in checking network misconfiguration, legacy network mapping, maintenance of growing/evolving networks, monitoring for system failure and monitoring for intrusions. In this paper, we demonstrate a modeling process to assist SCADA network mapping via network traffic analysis. Our topological modeling technique defines network modeling primitives which can be combined to create complete network traffic flow models. This technique can be used to confirm whether or not a network topology has been changed. This work investigates top-down data flow in SCADA networks and we explore a set-theory approach to developing the logical relationships needed to describe the data flow in a network. We exploit the rigid hierarchical structure of SCADA systems to produce a simple modeling approach. This is as a result of fixed number of network devices, regular network communication patterns and a limited number of protocols (Barbosa & Pras 2010).

This paper is divided into four sections. Section 1 introduces the work and its motivation. Section 2 examines related work on network traffic flow, network discovery...
and SCADA traffic monitoring. In Section 3, we describe our network modeling primitives used for modeling and mapping network topologies and we give descriptions of the compositional behavior of the modeling primitives and how these primitives are connected to form topologies. Practical case studies are presented in Section 4 and Section 5 provides the discussion. Finally, the conclusion is discussed in Section 6.

2 Related Work

Studying the structure and characteristics of large networks, especially the Internet, has drawn the attention of many researchers. Though there are challenges in monitoring network traffic, Lakhina et al. (2004) explained that one way of addressing this problem is by recognizing that observed traffic on different links of a network is not independent. This paper stated that a direct and fundamental way of studying network traffic is by analyzing the network’s set of Origin-Destination flows which indicates all traffic entering at a specific point and exiting the network at some other points. This concept is a starting point for our work in network traffic analysis.

Bykova et al. (2001) described how statistical analysis of network packet characteristics can be used in detecting network intrusions. This paper tried to identify how much information can be deduced about a packet by checking the packet headers but not their contents. The paper did not address how to detect inconsistencies in a network topology as a result of configuration errors, routing error or improper documentation. By contrast, our approach can detect such inconsistencies in a network topology.

In a related work, Barbosa et al. (2012a) provided a view into the characteristics of SCADA network traffic. This paper showed the similarity between SCADA traffic and Simple Network Messaging Protocol (SNMP) traffic as a regular time series which “presents baseline changes at seemingly arbitrary time intervals” (Barbosa et al. 2012a) because the majority of the sources generate data in a periodical fashion.

In another work by Barbosa et al. (2012b), they verified whether or not the models used in describing traditional network traffic can be used in SCADA traffic. The research was based on a list of network traffic invariants such as diurnal patterns of patterns, self-similarity, long normal connection sizes and heavy-tail distributions. They compared characteristics of SCADA traffic traces analysis with traditional network traffic and they concluded that the existing traffic models can not be easily applied to SCADA traffic.

SCADA networks were designed to be reliable and fail-safe, but attacks or security breaches over the past decade indicate that risks of deliberate attacks on the systems have not been adequately considered (Nicholson et al. 2012). There are numerous sources of threats to SCADA such as human errors, equipment failures, natural disasters, terrorist groups, hostile governments and industrial spies. Most of the attacks on these systems require studying and identifying vulnerabilities in the network topology which can be detected by monitoring the network topology. Monitoring a network, especially a time critical network such as a SCADA network, is a difficult and a demanding task. It has been a difficult objective to perform whole network traffic analysis, that is, monitoring every point in the network and modeling it.

To detect any deviation from the intended or expected communications topology of a SCADA system, which may indicate that the system has been compromised, network traffic analysis is required to study the “normal” data flow and generate a model topology, thus necessitating a modeling process. In our research, we show how to model SCADA network components and compose these primitives to predict expected traffic patterns.

3 SCADA Network Modeling and Mapping

SCADA networks consist of components such as Human Machine Interfaces (HMI), Remote Telemetry Units (RTU), Programmable Logic Controllers (PLC) and other devices (see Figure 1), connected hierarchically (in a tree-like structure) to achieve scalability. Data communications in SCADA systems are based on a master/slave (server-client) architecture (Igure et al. 2006, Daneels & Salter 1999). A SCADA control unit/master station communicates with field devices via communication channels such as fiber networks, public switch telephone networks (PSTNs), satellite links and even Ethernet. The control unit transmits SCADA messages to the field sites using SCADA protocols, but the field sites do not interact directly with each other. These features distinguish SCADA systems from general-purpose IT systems.

Observing network traffic assists in learning about a network topology and its behaviour. Our modeling and mapping approach for SCADA network topology discovery begins by modeling primitive components and combining them to predict what traffic should pass each point in the network. The run-time analysis process then entails matching captured packets at both ends of a network connection to compare with the model to detect inconsistencies.

In this paper, we study the network traffic behaviour assuming perfect communications in the network, however, future work will be the application of the traffic modeling process described in this paper to lossy communications as well as considering SCADA network analysis in real time. Our models focus on the communication path from the HMI to the PLC/RTU.

3.1 Modeling technique

We developed our modeling technique using set-theoretical relations to express the expected network behaviour as deduced from the physical topology. This section presents the various modeling primitives and shows their compositional behaviour.

3.1.1 Modeling Primitives

We model network components in terms of their input-output traffic flow by expressing the expected relationship between the set of packets entering the device and those
Figure 2: Atomic components for SCADA network modeling

excluding it. Figure 2 describes the various atomic components. For this purpose, we do not need to consider the full behaviour of networking devices (such as switches, hubs and routers). We can also ignore details such as the various protocols used in a SCADA network.

3.1.2 Atomic Components Behavior

The behavior of atomic components expresses the expected data flow through them and can be illustrated using set-theoretical relationships between various interfaces where network traffic could potentially be monitored. In Figure 2, letters A, B, C, etc represent the set of all packets passing a particular point in the network. The behaviours for various components are defined as follows:

Definition 1 (Buffer/Forward) This is a component with input equal to its output as shown in the pattern labeled B in Figure 2. Messages are transferred from the input interface to the output interface without any alteration or changes, so the set of entries at A is equal to the set of entries at interface B. This expresses the behavior of networking devices such as simple repeaters and it can be defined by:

\[ A = B \] (1)

Definition 2 (Broadcaster) This is a component with one input and multiple identical output interfaces as shown by the pattern labeled R in Figure 2. A broadcaster receives messages and the messages are repeated at all the output interfaces. This expresses the behavior of networking devices such as simple repeaters and it can be defined by:

\[ A = B = C \] (2)

Definition 3 (Switch) This is a component having one input and multiple output interfaces where a choice is being made between the outputs as shown by the pattern labeled S in Figure 2. When a host sends packets through the device, the packets received at B and C are not the same. This is as a result of switching according to the addressed destination (but the switching rules are not modeled here). This is defined by:

\[ A \subseteq B \cup C \] (3)

\[ B \cap C = \emptyset \] (4)

Definition 4 (Filter) This is a component in which messages from the input interface are filtered depending on configuration-specific rules as shown by the pattern labeled F in Figure 2. The filtering is defined based on user-defined policies (although these do not form part of our model). Security networking devices such as firewalls and intrusion prevention systems demonstrate this kind of behaviour. This is defined by:

\[ A \supseteq B \] (5)

Definition 5 (Merger) This is a component with multiple inputs and one output as shown by the pattern labeled M in Figure 2. Messages from multiple inputs are combined and sent to another interface as an output. This is defined by:

\[ A \cup B = C \] (6)

\[ A \cap B = \emptyset \] (7)

The log of packets captured at any interface is regarded as the elements of the interface. A set is defined as a collection of unique objects, hence, every packet observed at each interface must be uniquely identifiable.

3.1.3 Primitives Interconnectivity

The atomic components above are used to express the expected behavior of a network. This requires that some of these atomic components can be linked together to form a network topology. For instance, a filter component can be combined with a buffer and their respective input-output relationships can be merged to create a new compound network component. To achieve this, the steps are:
Step 1: Connection of required components
This first step entails linking required components together by joining their “connectors”. Connectors are represented with a dot (●) in Figure 2 and they are the observation points where packets are assumed to be captured. Figure 3 shows how two components, a switch and a buffer, are connected by joining their connectors to form a simple network topology.

Figure 3: Interconnectivity of atomic components

Step 2: Unification of the set variables
This step requires unifying the set variables of the two connected components. It is important to rename the set variables in the set equations to eliminate ambiguities. The diagram in Figure 4 shows how the variables are renamed and unified in this case.

Figure 4: Unification of set variables and renaming

Step 3: Simplification of resulting set equations
This last step involves simplifying the resulting set of equations to identify the relationships between the intended “observation points” in the network. Often this will show only end-to-end input and output points, but sometimes we will also allow for the possibility of adding a “tap” in the middle. An example of inferring the combined behaviour from the two component patterns is shown below in relation to Figure 4.

Pattern 1:
\[ A = B \cup C \]  (8)
\[ B \cap C = \emptyset \]  (9)

Pattern 2:
\[ B = D \]  (10)

Summary:
\[ A = D \cup C \]  (11)
\[ D \cap C = \emptyset \]  (12)

The set-theoretical relationships can be derived from the physical topology of the network and possibly other network documentation that reflect all necessary components in the network. The result describes the expected communications pattern of the network which will assist in network mapping and to verify if the actual network traffic behaviour is consistent with the assumed topology.

3.2 Network Analysis Process
Our SCADA Network Analysis process can be divided into four main procedures. The first procedure starts with developing a set of component models as described in Section 3.1.3 from physical topology diagrams and/or network documentation of the network to create the design model which expresses the expected communication topology.

The second procedure is capturing network packets from various points in the network. Network traffic is monitored at several possible points. The third procedure is the analysis to see if the observed traffic matches that predicted by the design model. This process includes comparing the captured packets at the input interface with the captured packets at the output interfaces. However, a unique packet identifier is required to uniquely match captured packets between the sender and various destinations. This could be one or a combination of fields in the packet. An Internet Protocol (IP) datagram consists of a packet header and data. The packet header contains fields such as source IP addresses, destination IP addresses, protocol, identification, Type of Service, header length, header checksum, time to live, among others used in routing, fragmentation and defragmentation of packets. In a network, each IP datagram is assigned a unique value in the identification field of every packet and all fragments of the datagram have the same identification number. The identification field has 16 bits which get reused. Thus, to uniquely identify a packet in a large capture, more fields of the packet header will be required. For this paper, the combination of source IP address, destination IP address and packet IP identification numbers is used as the unique packet identifier.

Having identified the packets, the next procedure is to compare the result against the packet set relationships in the derived model to verify whether or not the network traffic behaviour is consistent with the anticipated traffic behaviour. The derived logic expresses the expected communications behaviour of the network and any deviation from it proves that the network has been compromised.

4 Experimental Set-up
As a proof of concept, two experiments were conducted, a simple network using a SCADA testbed (simulating a water treatment and distribution system) and a more complex network in a simulated environment using a graphical network simulator (GNS3). Under each experiment, two tests were conducted and the following are the assumptions made:
• no loss of packets;
• data flow from top-down, that is from the HMI to the physical devices via PLCs;
• all packets are uniquely identifiable; and
• network traffic can potentially be monitored at every possible point in the network.

The first test for each experiment is to show how the model can be used to confirm that the network is consistent with the anticipated topology as stated in the network documentation while the other test shows how the model can be used to detect a change in the network.

This section describes the experimental set-up and the application of the network modeling and mapping techniques as stated in this paper.

4.1 Experiment 1

The first experimental set-up consists of two machines (a Controller and an attacker) and two PLCs. The controller provides both the HMI and Master Telemetry Units (MTU) functionality implemented using a LabView application while the attacker runs a Transmission Control Protocol (TCP) Modicon Communication Bus (Modbus) hacker program. The network includes a conventional Ethernet network as the data communications network and Modbus/TCP was utilized as the SCADA protocol. The target PLCs using a National Instruments Compact RIO provide the remote terminal unit (RTU). The controller communicates with the PLCs using a master/slave technique which is the architecture Modbus/TCP is based on. Each PLC controls a process setup shown in Figure 5 which comprises a water tank controller as may be found in a chemical plant. For this experiment, two tests were conducted; the first under a perfect environment while the other was in a compromised condition, to generate normal and attack traffic.

4.1.1 Methodology

Each test was conducted over a period of 180 seconds. The controller and attacker sent messages to the PLCs controlling the water pump and packets were generated.

For the first test, the controller attempted to turn the pump on and off at 30 second intervals while the attacker remained dormant. Figure 6 depicts the physical topology of the network and also shown in Figure 7 is a network pattern showing the expected communications topology constructed using our primitives.

Adopting the steps described in Section 3.1.3, a model for the network was derived using the physical topology to confirm if the predicted packet relationships conform to the traffic pattern generated from the captured packets. Below is the formulated relationships derived from this model:

\[
A = B \cup C \\
B \cap C = \emptyset \\
B = D \\
C = E
\]

The above equations can further be simplified as follows:

\[
A = D \cup E \\
D \cap E = \emptyset
\]

After developing the relationships representing the expected communication topology, we then tapped the network to observe traffic and captured network packets at points A, D and E using the Wireshark application. Shown in Figure 8 is a sample of captured packets at input interface A between the HMI.

The next step taken was the analysis of network packets. The captured packets were analyzed using the following steps:

• Extraction of required fields from the captured network packets such as source IP address, destination...
IP address and identification number. This forms the unique packet identifier to be used in matching packets between the sender and receivers;

- Packets captured at the sender interface were matched with the captured packets at the receivers’ (PLC 1 and PLC 2) interfaces. The result represented the number of matched and unmatched packets.

A sample of a packet analyzed showing the source IP address, destination IP address and the identification number is shown in Figure 9.

The second test was conducted with the presence of an attacker. The attacker attempted to disrupt the control of the pump by using the TCP Modbus Hacker program. For the second test, the Attacker lays dormant in the time period up to 60 seconds and in the time period of 61-90 seconds, the Attacker floods PLC 1 by performing the opposite actions of the Controller in the same period. This test was performed to show how our model can be used to detect deviations from the intended communications topology in the SCADA network. All generated traffic for the two environments were captured and then analyzed. The following section presents the obtained results and analysis.

### 4.1.2 Experimental Results and Analysis

In the two tests conducted, the captured packets at the controller interface were uniquely matched with the ones at each of the PLCs’ interfaces using the chosen unique packet identifier, and two tables were generated from the analysis. Table 1 shows the analysis of network packets captured under perfect conditions while Table 2 records the analysis of the network with the presence of an attacker.

**Table 1: Results under normal conditions for Experiment 1**

<table>
<thead>
<tr>
<th>Interface</th>
<th>No of matched packets</th>
<th>No of unmatched packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>40620</td>
<td>nil</td>
</tr>
<tr>
<td>A and D</td>
<td>20311</td>
<td>nil</td>
</tr>
<tr>
<td>A and E</td>
<td>20309</td>
<td>nil</td>
</tr>
</tbody>
</table>

The result of the first test as shown in Table 1 indicates the number of packets sent as messages from the controller via interface A to each of the PLCs, observed at each of the interfaces D and E. The result was compared to the packet relationships formulated from the physical topology of the network using our modeling primitives. According to the model, the log of entries at point A must be the union of entries observed at points B and C. In Table 1, all packets matched, so the topology is consistent with the expected behavior of the network.

On the other hand, the result of the second test as shown in Table 2 indicates that not all packets sent from the controller and observed at point A matched the captured packets observed at point E of PLC1. This suggests that the network traffic is not consistent with the anticipated network communications.

For further analysis, we generated a traffic model from the network for each test. This is done by matching the log of packets from the input interface with its output interface(s) using the unique packet identifier, as shown Figure 7, packets at point A are matched against output packets at points B and C, while D and E are the output of B and C respectively.

The created traffic models are compared to the design model generated from the physical topology and Table 3 presents the comparison.

**Table 2: Results under an unsecured environment for Experiment 2**

<table>
<thead>
<tr>
<th>Interface</th>
<th>No of matched packets</th>
<th>No of unmatched packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>40620</td>
<td>nil</td>
</tr>
<tr>
<td>A and D</td>
<td>16934</td>
<td>10054</td>
</tr>
<tr>
<td>A and E</td>
<td>20308</td>
<td>nil</td>
</tr>
</tbody>
</table>

**Table 3: Comparison between original model and traffic models for Experiment 1**

<table>
<thead>
<tr>
<th>Design Model</th>
<th>Traffic Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>Test 2</td>
</tr>
<tr>
<td>$A = B \cup C$</td>
<td>$A = B \cup C$</td>
</tr>
<tr>
<td>$B = D$</td>
<td>$B = D$</td>
</tr>
<tr>
<td>$C = E$</td>
<td>$C = E$</td>
</tr>
<tr>
<td>$D \cap E = \emptyset$</td>
<td>$D \cap E = \emptyset$</td>
</tr>
</tbody>
</table>

These results indicate that the state of the network does not match the expected behavior.
not conform to the anticipated behaviour and the implications are as follows:

- This suggests that there are deviations from the intended communications topology, which means that the original network topology has been compromised.
- It also indicates that there might be an intrusion, error in routing or configuration error.

Section 5 discusses further the analysis of this experiment.

4.2 Experiment 2

In this second experiment, a more complex topology was designed in a simulated environment using a graphical network simulator (GNS3). The experimental set-up consisted of a router with firewall, switches, hubs and six workstations, and two tests were conducted. The network was divided into two network domains with two workstations (I and II) controlling each domain. Assigned to domain 1 were workstations I, III and IV while workstations II, V and VI formed the domain 2. The physical topology of the network is shown in Figure 10.

4.2.1 Methodology

Two tests were conducted and for the first test, the network was set up as shown in Figure 10 while for the second test, the network set-up was altered. The aim was to detect whether or not the alteration could be detected using our network modeling and mapping approach.

For clarity, the entire network is divided into four domains as indicated in Figure 10 and shown in Figure 11 is the model developed using our primitive components.

Traffic relationships were derived from the model of the network as shown below.

Group 1:

\[ A \cup B = C \]
\[ A \cap B = \emptyset \]

Group 2:

\[ C \supseteq D \]

Group 3:

\[ D = E \cup F \]
\[ E \cap F = \emptyset \]

For this experiment, a firewall was configured on the router to filter outgoing packets from the sender and this implies that packets are checked to confirm if they conform to the set policy which states that any packets outside the specified network addresses should be dropped. Therefore, in a lossless communication environment, no packet is expected to be dropped, i.e.

\[ C = D \]

To simplify the resulting set equations:

\[ A \cup B = G \cup H \cup I \cup J \]
\[ A \cap B = \emptyset \]

In addition, based on our knowledge of the way the generated packets are addressed and the switches are set in this network configuration, we further introduce more specific constraints for this particular set up:

\[ A = G \cup H \]
\[ B = I \cup J \]

Following the derivation of the network logic is network capturing. Network packets were captured at various input and output points; A, B, G, H, I and J, using the
Wireshark application. In GNS3, network interfaces can be tapped to monitor network traffic. 200 packets were sent simultaneously from the workstations (I and II) in group 1 as shown in Figure 10 to other workstations within their assigned domain.

For the second test, the switch connecting workstations V and VI was replaced with a hub. This was to investigate the impact of a change in a device on network communications.

The captured packets were analyzed by matching the entries at both inputs with the entries at the outputs. Presented in the following section is the experimental result and analysis.

### 4.2.2 Experimental Result and Analysis

For the first test, network traffic was observed at interfaces A, B, G, I, H and J, and a unique packet identifier which is the combination of the source IP address, destination IP address and identification number, was used to match packets between the senders and the receivers. Packets sent from workstation I as observed at point A were matched against captured packets at points G and H, and likewise entries at point B sent from workstation II were matched against entries at points I and J. Tables 4 and 5 reveal the results of the matched packets for both domains.

Table 4: Experiment 2, the analysis of domain1 for test 1

<table>
<thead>
<tr>
<th>Interface</th>
<th>No of matched packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>200</td>
</tr>
<tr>
<td>A and G</td>
<td>100</td>
</tr>
<tr>
<td>A and H</td>
<td>100</td>
</tr>
</tbody>
</table>

In the second test conducted, an increased number of packets were observed at points A, I and J when compared with the observed packets at points B, G and H. Table 6 shows the comparison between the observed packets at points A and B. Also observed at point B, I and J were the same number of packets as shown in Table 7.

Table 5: Experiment 2, the analysis of domain 2 for test 1

<table>
<thead>
<tr>
<th>Interface</th>
<th>No of matched packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>200</td>
</tr>
<tr>
<td>B and I</td>
<td>100</td>
</tr>
<tr>
<td>B and J</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 6: Experiment 2, the analysis of domain 1 for test 2

<table>
<thead>
<tr>
<th>Interface</th>
<th>No of packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>200</td>
</tr>
<tr>
<td>B</td>
<td>205</td>
</tr>
</tbody>
</table>

Table 7: Experiment 2, the analysis of domain 2 for test 2

<table>
<thead>
<tr>
<th>Interface</th>
<th>No of packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>205</td>
</tr>
<tr>
<td>I</td>
<td>205</td>
</tr>
<tr>
<td>J</td>
<td>205</td>
</tr>
</tbody>
</table>

Tables 4 and 5 reveal the result of the first test showing that the number of packets detected at the senders’ interfaces are equal to the sum of the number of packets detected at the receivers’ end of the connection and accords with the network configuration which was derived, $A = G \cup H$ and $B = I \cup J$. The network traffic analysis thus shows that the network behaviour is consistent with the anticipated topology as stated in the physical topology of the network.

For each test conducted in this experiment, a traffic model was also generated by matching each packet cap-
tured at the input point(s) against its output point(s). As shown in Figure 6, packets at points A and B are matched against packets at point C, and likewise other points. The created traffic models are compared to the design model generated from the physical topology and depicted in Table 8 is the comparison of the models.

Table 8: Comparison between original model and traffic models for Experiment 2

<table>
<thead>
<tr>
<th>Design Model</th>
<th>Traffic Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ∪ B = C</td>
<td>A ∪ B = C</td>
</tr>
<tr>
<td>A ∩ B = ∅</td>
<td>A ∩ B = ∅</td>
</tr>
<tr>
<td>C = D</td>
<td>C = D</td>
</tr>
<tr>
<td>D = E ∪ F</td>
<td>D = E ∪ F</td>
</tr>
<tr>
<td>E ∩ F = ∅</td>
<td>E ∩ F = ∅</td>
</tr>
<tr>
<td>E ∪ H = E</td>
<td>E ∪ H = E</td>
</tr>
<tr>
<td>F ∪ I = I</td>
<td>F ∪ I = I</td>
</tr>
</tbody>
</table>

As shown by the final row, the change of device is revealed by the change in the observed traffic.

5 Discussion

The two experiments shown in this paper have demonstrated how network traffic analysis can be used to identify whether network flow is consistent with the expected topology. In each experiment, two tests were performed to check how set theoretical component models can be used to depict data flow in a network and also used in developing a model topology.

In Experiment 1, the result of the first test confirmed the consistency of in the network’s behaviour when compared with the expected communications topology expressed using our model derived from the network physical topology, while on the other hand, the second test revealed the inconsistency.

The result of the second test of Experiment 1 indicated that the set of the received packets at both output points D and E was not equal to the observed packets at the input point, that is, A ≠ D ∪ E which is contrary to the relationship formally derived from the physical topology, A = D ∪ E. This was because not all packets sent from the HMI were observed at PLC 2 while all the packets sent from the same HMI to PLC 1 got to their destination. In order to detect where the changes occurred, network traffic was observed at every point on the network. After monitoring point C, analysis detected the presence of an unknown source (an attacker) connected to the network and overriding the operations of the HMI (as shown in the model in Figure 12 where F represents the attacker). While matching the packets from the input (A) to the output (E) using the unique packet identifier, there were some unmatched packets which came from an unidentified source IP address.

In Experiment 2, the result of the first test confirmed that the network behaviour was consistent with the anticipated topology. According to the derived relationships for the network, the captured packets at the input points (A and B) must be the union of the packets captured at the output points (G, H, I and J) and the result showed that this was true.

However, the result obtained from the second test was not consistent with the anticipated behaviour. The test revealed a change in the expected communications topology, 200 packets were sent from workstations I and II each but the observed traffic at point B indicated extra packets. Further analysis showed that the captured packets at point I and J were the same, that is, I = J, which is contrary to the derived logic, J ∩ J = ∅. Further investigation showed that the changes observed in the network were as a result of the change in a network device. Every point in the network was observed and analyzed to identify the cause of the change in the network’s traffic behaviour. The analysis discovered a behavioural change as a result of the removal of the switch connecting workstation V and VI which was replaced with a hub, this led to an increase in the number of packets observed at the interfaces. A hub broadcasts messages and this explains why common traffic was observed at points B, I and J.

We noted that our approach serves as an essential basis for SCADA network topology analysis. Even though using our modeling and mapping process as described in this paper assisted in identifying the consistency of an anticipated network communications topology with the network traffic, there is more that could be achieved using this approach. The following work is to be investigated in future using our modeling and mapping process:

- SCADA network topology discovery. Listening at every point on a network could assist in generating a complete topology of the network but this is impracticable in a real network. Therefore, to monitor a SCADA system and develop a complete topology of the system creates a problem: What point(s) on the network need to be observed to generate a topology of the network without monitoring every point?
- Lossy communication analysis. In SCADA network monitoring, lossy network communication cannot be disregarded. In this paper, we considered perfect communication with some assumptions but for lossy communications, the assumptions will be weakened to allow for packet loss.
- Automated Analysis. In this paper, we performed our analysis manually, however, this might not scale properly in a network with thousands of components. In our future work, we will consider using an automated process in solving scalability issue while still using the same methodology.
- Real-time network analysis. Our future work will investigate how our modeling techniques could be applied in analyzing a SCADA network in real time where complete logs of network traffic are not available. This requires the development of an automated modeling approach for analyzing “recent” network traffic.
6 Conclusion

In this paper, we have presented simple models to describe data flow through a SCADA network topology based on the captured network traffic. This approach can be used in SCADA network mapping due to its simple and predictable traffic behaviour. We considered various network primitives as well as the underlying input-output relationships needed for modeling a SCADA network. This work has shown how such relationships can be developed by studying the physical topology of a network and can then be used in detecting deviations in a network’s communications from the intended communications topology, indicative of intrusions or system failures.

References


