# Delay Performance and Cybersecurity of Smart Grid Infrastructure

# Delay Performance and Cybersecurity of Smart Grid Infrastructure

by

Huan Yang

Presented to the Graduate and Research Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy

in

Computer Engineering

Lehigh University

May 2019

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Date

Dissertation Advisor

Accepted Date

Committee Members:

Prof. Liang Cheng, Committee Chair

Prof. Mooi Choo Chuah

Prof. Michael Spear

Prof. Yinzhi Cao

Prof. Wenxin Liu

Prof. Xiaoguang Ma

# Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor, Professor Liang Cheng, in providing me guidance and support from the beginning of my Ph.D. program. Throughout my pursuit of the Ph.D. degree, Professor Cheng not only afforded me invaluable opportunities to explore a wide variety of research problems but also positively influenced various aspects of my life by sharing precious experiences and offering excellent advice. Without his patience, expertise, and persistent support, this dissertation would not have been possible.

I would also like to thank all members of my dissertation committee, including Profeesor Liang Cheng, Professor Mooi Choo Chuah, Professor Michael Spear, Professor Yinzhi Cao, Professor Wenxin Liu, and Professor Xiaoguang Ma, for taking the time to attend my depth study presentation, general exam, and disseration defense. Insightful comments and ideas shared by the committee members have not only helped me improve my dissertation draft but also inspired me to explore further in my fields of interest. I especially appreciate all the help, advice, and guidance provided by Professor Chuah and Professor Ma. My work on cybersecurity of the smart grid has greatly benefited from collaboration and frequent discussion with Professor Chuah. Professor Ma, on the other hand, has generously devoted time and resources to assiting me in both gaining a better understanding of smart grid network infrastructure and identifying research problems that are of immediate practical value.

I greatfully acknowledge the support from various research sponsors, including PPL Utilities Corp., ABB Inc., Pennsylvannia Infrastructure Technology Alliance (PITA), Ingersoll-Rand plc., U.S. Department of Energy (DoE) under Award No. DE-OE0000779, as well as the National Science Foundation (NSF) through CNS grant No. 1646458.

Though research is the theme of my life in the past few years, my dissertation would not

have come to a successful completion without the help from the the staff of the Department of Computer Science and Engineering and the P.C. Rossin College of Engineering and Applied Science. In particular, Jeanne Steinberg and Heidi Wegrzyn have been friendly, patient, and always willing to lend their services whenever I requested help. I would also like to thank Brianne Lisk for her help and advice.

I am also thankful to my colleagues and fellow students who have helped me in a variety of ways. Former members and visiting scholars of LONGLAB, including Zi Wang, Xu Li (Eric), Dr. Yanhong Yang, Dr. Kaifeng Zhang, and Dr. Wei Liang, have extended their support in a very special way, and I have gained a lot from personal and scholarly interactions with them. I also acknowledge fellow students working in the same office with me, including Xin Li, Xiaowen Ying, Xiao Zang, Wanzhao Yang, and Isaac Howenstine, for fostering a positive work and learning environment.

I owe a lot to my parents, who encouraged and helped me at every stage of my personal and academic life, and longed to see this achievement come true. I am very much indebted to my wife, Dr. Qi Liu, who supported me in every possible way. Without their constant support and patience, I would not have made it thus far.

# Contents

# List of Tables

# List of Figures

# Abstract

To address major challenges to conventional electric grids (e.g., generation diversification and optimal deployment of expensive assets), full visibility and pervasive control over utilities' assets and services are being realized through the integration of energy, information, and communication infrastructure that forms the smart grid. As an indispensable component of the smart grid infrastructure, Supervisory Control and Data Acquisition (SCADA) system interconnects fields devices and computer servers within a power substation and allows operators to remotely control these devices and equipment from the control center. In recent years, switched Ethernet gains growing popularity in power substations and an increasing number of control tasks are now carried out by SCADA hosts exchanging data and commands over Ethernet. This paradigm (i.e., network-based distributed control, interconnectivity between SCADA hosts and among different SCADA sites) has caused concerns over power system reliability and cyber-security: Network-induced delays experienced by SCADA network packets must be kept below task-specific upper bounds to ensure reliable system operation. In addition, SCADA hosts (e.g., protective relays, human-machine interface, and engineering workstations) must be protected from miscellaneous cyber attacks. To address these concerns, techniques for delay performance analysis as well as solutions against cyber attacks on SCADA systems are designed and developed in this dissertation.

Specifically, this dissertation investigates the following research problems: (i) Worst-case delay performance analysis for networked cyber-physical systems (CPSs): We first study the worst-case delay performance of Ethernet-based substation communication networks (SCNs) through the combination of measurements and network-calculus-based modeling, which enables SCN architects to estimate the delay performance of an SCN design under different

operational scenarios. The outcome of research task is a delay performance modeling and analysis framework helping SCN architects to verify whether stringent delay performance requirements of critical control operations (e.g., tripping a circuit breaker to isolate fault) are satisfied. Furthermore, we also analyze wireless Parallel Redundancy Protocol (PRP) infrastructure recently proposed for industrial control systems (ICSs) and obtain closed-form expressions for the network-induced worst-case delays under general, non-feedforward traffic patterns. (ii) Intrusion and botnet detection for SCADA networks: To protect SCADA systems from cyber attacks, we design network-based intrusion and botnet detection algorithms for SCADA systems. To detect SCADA hosts infected by peer-to-peer (P2P) botnets, we analyze traffic patterns and characteristics of different SCADA hosts and identify those performing command and control (C&C) communication with other bots or the bot master. Besides botnets, cyber attacks targeting SCADA network protocols (e.g., DNP3) are one of the primary ways for attackers to disrupt system operation and cause physical damages. A deep-learning-based algorithm is devised to detect these attacks by analyzing application-layer information of network packets. (iii): Payload attack detection for programmable logic controllers (PLCs): Widely used in SCADA systems, PLCs are susceptible to a special class of attacks (known as PLC payload attacks) where an attacker with PLC programming privilege injects malicious control logic into the PLC control program. To detect such attacks, we model the runtime behaviors of legitimate PLC control program and detect malicious control programs with abnormal runtime behaviors at PLC firmware. Evaluation results obtained in controlled lab environment and from hardware-in-the-loop cybersecurity test bed show that our approach effectively identifies modification attacks on PLC control logic with acceptably low runtime overheads, making it a viable firmware enhancement scheme for existing and future ICS field devices.

# Chapter 1

# Introduction

Supporting a broad spectrum of tasks (e.g., system control and protection, process visualization at human-machine interface), Supervisory Control and Data Acquisition (SCADA) system [6] is a critical component of networked cyber-physical systems (NCPSs) such as the smart grid. As conventional power grids evolve into smart grids, switched Ethernet becomes one of the most popular communication technologies for power substation automation [30,68]. To provide full visibility and pervasive control over utilities' assets and services, SCADA hosts, ranging from protective relays to engineering workstations, are now accessible from the control center. The increased interconnectivity between SCADA hosts and among SCADA sites, as well as the use of Ethernet for substation automation, have brought forth concerns over reliable control, operation, as well as cybersecurity. This chapter gives an overview of the research problems in delay performance analysis and cybersecurity of smart grid infrastructure that are addressed by this dissertation and provides necessary technical background to facilitate an in-depth understanding of the research challenges and contributions detailed in later chapters.

## 1.1 Motivation

In the past decade, switched Ethernet gradually gathered momentum in substation automation due to its continuous improvements on bandwidth and noise immunity [118]. Based on Ethernet, the IEC 61850 standard promotes interoperability between field devices from different manufacturers, reduces cabling costs and complexity, and enables new services and capabi-

lities that are not possible with most legacy protocols [83]. While enabling an ever-growing number of new services and applications, the use of Ethernet and the increased interconnectivity between hosts in a substation arouse concerns over real-time control of critical system processes. In the IEC 61850 standard, worst-case delay performance requirements for different types of tasks and services are specified, which must be stringently enforced when designing Ethernet-based substation communication networks (SCNs). For safety-critical tasks, an SCN design must provision deterministic (i.e., worst-case) rather than stochastic delay performance guarantees to ensure reliable dissemination of crucial information under any circumstances. Therefore, it is of vital importance to devise an approach to analyzing and evaluating worst-case delay performance of smart grid network infrastructure.

In fact, many existing industrial control systems (ICSs) have started to embrace the networked cyber-physical system paradigm [138, 139]: Computer networks (e.g., switched Ethernet, Wi-Fi network) are deployed to enable data exchange among controllers, sensors, and actuators distributed across the system. Some of these ICSs require both ease/flexibility of field device deployment and timely transmission of critical process data and commands. To leverage the flexibility offered by wireless networking technologies while ensuring reliable transmission of data over communication channels that may occasionally become unstable, a redundancy mechanism based the Parallel Redundancy Protocol (PRP) has been proposed. Traffic patterns on such wireless networks are inevitably non-feedforward, rendering existing analytical approaches targeting tandem or feedforward networks inapplicable. To facilitate the design of wireless PRP infrastructure with deterministic delay performance guarantees, an analytical method that properly handles the intricacies introduced by non-feedforward traffic patterns is indispensable.

Another major concern aroused by the increasing interconnectivity among ICS field devices such as SCADA hosts in a power substation is cybersecurity. As SCADA network hosts exhibit traffic characteristics that are significantly different from those in IT networks (e.g., a corporate network or a campus network), existing network intrusion detection systems (NIDS) developed for IT networks may not work well in SCADA environments. Among recently emerged cyber threats, peer-to-peer (P2P) botnets can become a serious threat to SCADA

systems. Botnet malware can be slipped into SCADA systems using an approach that is similar to the Stuxnet [71] attack or by a malicious insider. The malware can replicate itself onto other hosts, and compromised SCADA hosts can launched coordinated attacks that can easily disrupt system operation and/or induce severe physical damages. Therefore, it is necessary to design a network intrusion detection solution against P2P botnets in SCADA systems. Once a SCADA device is compromised, an attacker may exploit vulnerabilities of SCADA network protocols and launch attacks purposely crafted for SCADA systems. Such specialized attacks must be detected by NIDS for SCADA systems in order to protect critical system processes from physical damages and operation disruptions. Therefore, a network intrusion detection system must be able to detect emerging cyber attacks such as P2P botnets as well as specialized attacks on SCADA network protocols.

In addition to network attacks on SCADA systems, modification attacks on control programs of field devices (also known as payload attacks) are also a major cybersecurity threat to industrial control systems such as the smart grid. Such attacks can easily be launched either by a malicious insider with device programming privilege or by an external attacker who successfully compromised an engineering workstation. Detecting payload attacks by manually reviewing control program source code can be a tedious task, and a malicious insider may still find ways to plant control logic bombs (e.g., right after a round of code review is completed). Existing methods introducing additional apparatus (e.g., a runtime behavior monitor) into an ICS indeed automate the attack detection process, but it has been shown that attacks on real-time performance requirements (e.g., a control action must be taken within 10 ms after the moment certain faults are observed) cannot be properly detected. To enhance the resilience of ICS field devices against payload attacks, it is important to design an automated detection method that does not require the installation of additional devices (which may become a new target for cyber attacks) and can properly detect attacks on real-time performance requirements of control programs.

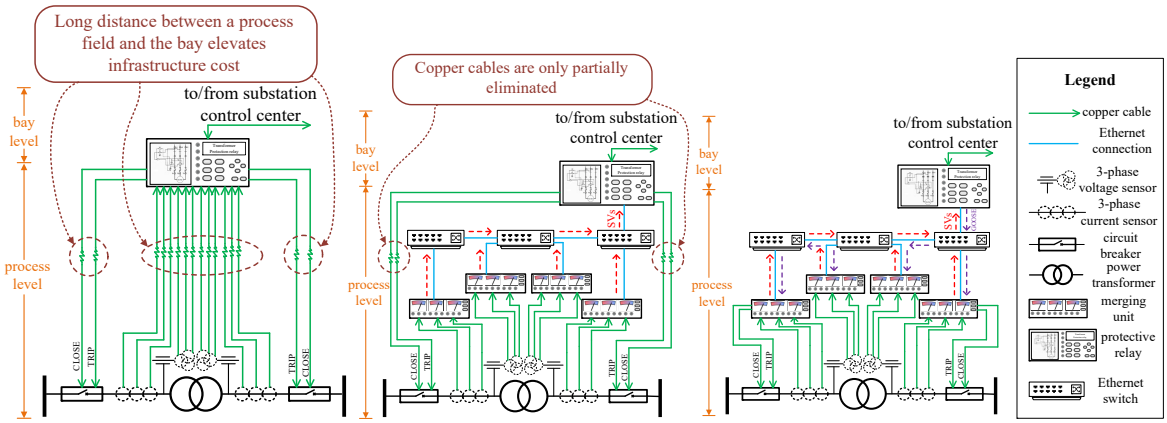
## 1.2 Research Problems and Challenges

### 1.2.1 Delay Performance Analysis of Substation Communication Networks

As an example instance of networked cyber-physical systems (NCPSs), substation automation system (SAS) leverages its network infrastructure (e.g., a switched Ethernet) to realize various tasks with different criticality levels. Figure 1.1 compares substation communication networks based on hardwired connections and switched Ethernet. In the substation communication network (SCN) shown by Figure 1.1a, voltage and current sensors as well as actuators are hardwired to a protective relay. If the physical process is located far away from the cabinet housing the protective relay, a large number of long metal wires must be employed. The SCN design in Figure 1.1b eliminates part of the hardwire connections. However, connections between actuators (i.e., the circuit breaker) and the protective relay are still hardwired. Figure 1.1c presents an SCN design where sensor readings and control commands are transmitted between process field and the control room using Ethernet. In addition to reducing cabling cost and complexity, the use of Ethernet also makes it easier to add/remove new devices [30, 68]. As new standards such as IEC 61850 recommend Ethernet for organizing SCADA hosts in a substation, Ethernet continues to gain momentum in substation automation, protection, and control. In fact, applying Ethernet in wide-area measurement systems (WAMS) and inter-substation communication networks has also obtained attention from different research groups [81, 119]. Therefore, switched Ethernet will become a major communication technology not only for SCADA systems within substations but also for systems consisting multiple SCADA sites (e.g., a control center and multiple substations it monitors).

As more and more new services and applications are realized on SCNs based on IEC 61850, the communication patterns on an SCN continues to evolve. In [72], message sizes and requirements on network-induced delays of assorted power system applications are surveyed. It is shown in [57] that a maximum of 22 merging units can be supported by Ethernet switches with 100 Mbps bandwidth. However, it is hard to answer the following questions merely through measurements:

1. Given the expected traffic patterns for an SCN design, how can we verify whether the




(a) Hardwired (b) Ethernet and hardwired (c) Switched Ethernet

Figure 1.1: Substation communication networks using different communication media.

delay performance of an SCN design will satisfy the real-time requirements of the control, protection, and automation tasks?

2. As new services continue to be integrated into the SCADA system, is there a tool that will allow us to verify the delay performance of the SCADA network under new traffic patterns without interrupting system operation?

3. The network traffic patterns imposed by SCN applications and tasks are generally non-feedforward, whereas existing analytical methods are designed for tandem or feedforward networks. Can we leveraging existing methods to model and analyze non-feedforward networks?

To answer these questions merely through measurements, new measurements must be taken whenever the SCN design or the expected traffic patterns change. After a substation is put into service, it is sometimes hard to take measurements without affecting the operation of the substation (e.g., we may need to install network taps into the SCADA network for measurement purpose, which will inevitably disconnect certain devices from the network temporarily).

To address these challenges, it is of vital necessity to devise an analytical framework that can derive worst-case delay performance bounds for Ethernet-based substation communication networks. Characteristics of traffic sources and networking devices of SCNs that can be observed from measurement should be incorporated into the analytical framework to obtain application-specific, tight delay bounds. Designing such a framework and incorporating it into

the different phases of an SCN's life cycle (e.g., initial design, deployment, maintenance, and renovation) will not only facilitate the rapid verification of deterministic delay performance of various SCN designs but also provide guidance on how existing SCNs can be renovated to achieve better delay performance.

### 1.2.2 Delay Performance Analysis of Wireless Network Infrastructure for Cyber-Physical Systems

As many industrial control systems (ICSs) start to embrace the paradigm of networked cyber-physical systems (NCPSs), various wireless networking technologies have become increasingly popular primarily because of their flexibility (e.g., deploying new devices no longer requires reorganizing existing cabling). However, performance of many wireless communication technologies deteriorates under noisy environments. To take advantage of existing wireless communication technologies while realizing reliable and timely information dissemination, various wireless networking schemes leveraging the redundancy principle introduced by the Parallel Redundancy Protocol (PRP) have been proposed.

To verify whether hard-real-time performance requirements are satisfied by a certain wireless PRP network, application-specific evaluations (e.g., discrete-event simulation or field experiments) have been conducted. However, new experiments have to be carried out when the same networking scheme is applied to a new site (or the same site with a different set of services and tasks). Furthermore, the traffic patterns on a wireless PRP infrastructure are non-feedforward by nature, and a proper analytical method analyzing the delay performance of such networks is yet to be established. To facilitate the adoption of wireless PRP network infrastructure by industrial applications with hard-real-time performance requirements, the following research challenges must be properly addressed:

1. Given an NCPS design based on wireless PRP, can we devise a method to analyze the worst-case delay performance over its non-feedforward traffic patterns so that whether the design satisfies all the hard-real-time delay performance requirements can be verified?

2. In addition to simply accepting or rejecting an NCPS design based on our analytical results, can we further provide guidance on how an existing design can be enhanced (e.g., to

accommodate new services or improve the delay performance for existing applications)?

A possible response to these identified research challenges is to derive the closed-form expressions of the worst-case delay bounds for time-critical network traffic flows. So far, little has been done to find such expressions and discuss possible performance enhancements for non-feedforward networks.

### 1.2.3 Intrusion and Botnet Detection for SCADA Networks

As the interconnectivity between SCADA hosts and among SCADA sites continues to improve, an increasing number of cyber attacks targeting SCADA systems are launched through SCADA networks. Figure 1.2 shows a SCADA system for a substation consisting of four protective relays, a remote terminal unit (RTU), an engineering workstation, and a human-machine interface (HMI). An Ethernet switch interconnects the field devices and computer servers and carries SCADA network packets exchanges between them. In addition, operator in the control center can access the SCADA hosts within the substation via the router. Over the past few decades, instances of different cyber attacks have been reported [90, 141]. Some of these attacks are launched by external attackers. For instance, the 2016 attacks on the Ukrainian power grid [75, 76] are launched by external attackers who first gain access to SCADA hosts of a transmission-level substation and then executed breaker "trip" commands to cause power outage. Although such attacks employ multiple components, the last step of these attacks is to issue SCADA commands and cause the system to malfunction/fail. To prevent SCADA systems from physical damages caused by cyber attacks, it is important to design intrusion detection algorithm that identifies abnormal network activities (e.g., operators should be alerted if control commands that are not supposed to be executed by protective relays are observed on the SCADA network).

Among recently emerged cyber attacks, peer-to-peer P2P botnet is a potential threat to SCADA systems. When SCADA hosts are infected by peer-to-peer botnets, they first enter command and control phase to collect commands issued by the bot master. The command and control (C&C) traffic between P2P bots bears resemblance to the decentralized communication patterns of regular SCADA networks. After the C&C phase, each bot-infected SCADA host

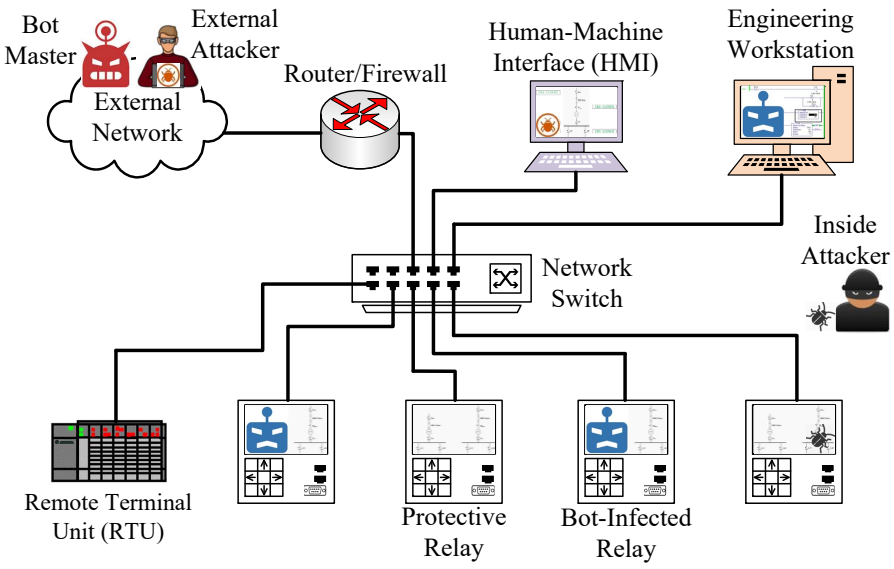



Figure 1.2: A small SCADA system with field devices and computer servers.

obtains an up-to-date copy of commands that can cause severe damages to SCADA devices and the physical process. To prevent P2P bots from causing any damages, it is important to identify bot-infected SCADA hosts during their C&C phase.

In addition to newly emerged threats such as P2P botnets, specialized attacks on SCADA network protocols also need to be taken into account by intrusion detection systems (IDS) for SCADA networks. In contrast to network attacks observed in IT networks, these specialized attacks typically aim to interrupt field device operation, hoax actuators into taking incorrect control actions, masking malicious activities or faults with falsified process data, and ultimately induce severe physical damages. Existing intrusion detection solutions for IT networks cannot be directly adopted to detect such attacks because (i) the vulnerabilities exploited by attackers are specific to SCADA network protocols or well-known control mechanisms and (ii) some of the message types are leveraged both by normal system operations and specialized attacks. To detect such attacks, application-layer information of SCADA network packets needs to be properly extracted and analyzed by intrusion detection systems.

### 1.2.4 Detecting Payload Attacks on Programmable Logic Controllers (PLCs)

Programmable logic controllers (PLCs) are special-purpose computers that runs control program written by control system engineers on PLC firmware. The control program is also

known as the payload to the PLC's firmware. An insider (e.g., a disgruntled employee) can easily modify the control program source code and download a malicious copy to the PLC. Since the attackers usually have knowledge of the inner-workings of the SCADA system (e.g., which set of control signals can cause severe damage, how long an HMI will poll for updated process data), modification attacks on PLC control program (also known as PLC payload attacks) are easy to implement but hard to detect. In addition, such attacks may not generate any network packets, so a network-based intrusion detection system (IDS) cannot capture such attacks effectively.

Take the SCADA system depicted in Figure 1.2 as an example. An insider with PLC programming privilege can easily modify the control program and inject malicious control logic from the engineering workstation. To cause severe damage and escape the notice of other control system engineers, the attacker can set the malicious actions to be triggered by certain system events (e.g., blocking circuit breaker trip signals when short-circuit faults are observed). To protect SCADA systems from PLC payload attacks, it is of critical importance to devise a mechanism that detect payload attacks and prevents malicious control actions from being executed.

## 1.3 Dissertation Statement and Contributions

As discussed in Sections 1.1 and 1.2, reliable control and operation of modern networked cyber-physical systems demand that control applications, especially those with real-time performance requirements, should be able to carry out all their tasks correctly and timely under any circumstances. To help realize this goal, this dissertation examines the research problems of the worst-case delay performance analysis and evaluation and cybersecurity in the context of the smart grid, which has been regarded as a representative example of networked cyber-physical systems. Contributions made in this dissertations to each of the research problems and challenges identified in Section 1.2 can be summarized as follows:

- In order to help architects of substation communication networks (SCNs) verify worst-case delay performance of their network designs at different stages of an SCN's life cycle, we devise an analytical framework combining delay measurements with deterministic

network calculus to derive worst-case delay bounds for time-critical traffic flows. Our analytical framework is well-suited for the delay performance analysis of Ethernet-based SCNs for the following reasons:

  - ▷ By properly modeling Ethernet switches and taking end-to-end delay measurements, our framework transforms non-feedforward traffic patterns imposed by SCN applications into feedforward ones, making it possible to apply state-of-the-art network-calculus-based analytical methods and obtain accurate bounds on network-induced delays.
  - ▷ Through the combination of measurements and network-calculus-based modeling, we proposed approach refine analytical models utilized by state-of-the-art theoretical analyses with measurement-based observations, further tightening the delay bounds without increasing the time complexity of existing techniques.
  - ▷ We compare measurement results obtained from an emulated SCN test with analytical bounds and show that delay bounds given by our framework are sufficiently tight for SCNs with representative traffic patterns and workloads.
  - ▷ We also extend the application of our framework to other networked cyber-physical systems (NCPSs) and demonstrate that our approach is also able to give tight bounds for these Ethernet-based NCPS applications.

- To derive worst-case bounds on delays induced by wireless PRP network infrastructure of NCPSs, we apply network calculus with stopped sequences to model and analyze non-feedforward traffic patterns imposed by industrial control applications. Our approach finds the closed-form expressions of the delay bounds for time-critical flows, which can be applied to NCPSs with wireless PRP infrastructure in two different ways:
  - ▷ Given an existing NCPS network design and the expected traffic pattern, our approach can be applied to derive analytical delay bounds to help NCPS architect quickly verify whether the current design satisfies the hard-real-time performance requirements of industrial applications.

- ▷ During different phases of an NCPS's life cycle, closed-formed expressions derived by our framework (which can also be further refined with measurement data) can help NCPS architects identify potential delay performance bottlenecks, providing guidance on the design and renovation of wireless PRP infrastructure of NCPSs.

  Our approach facilitates worst-case delay performance analysis using back-of-the-envelope calculations, making it a valuable tool that can be repeatedly applied during the NCPS design process.

- To address concerns on cybersecurity of SCADA systems with ever-increasing interconnectivity, we design machine-learning-based intrusion detection algorithms for SCADA networks. Our proposed solutions are well-suited for SCADA systems for the following reasons:

  - ▷ By modeling flow-based and connectivity-based characteristics of SCADA netowrk hosts, we devise an unsupervised learning algorithm to detect peer-to-peer (P2P) botnets during their peer discovery phase in SCADA environments. Our approach is capable of identifying previously unseen bots and achieve high detection accuracy with few false positives, which meets the requirements for intrusion detection systems for SCADA networks.
  - ▷ In addition to detecting P2P botnets, we also design deep-learning-based algorithm that inspects application-layer information of SCADA network packets and identifies both conventional and specialized attacks on SCADA networks. Our approach can identify attacks that leverage the same packet formats as normal control applications and generate only a few false positives.
  - ▷ Our proposed algorithms leverage traffic monitoring capabilities of existing SCADA networking devices and do not interrupt SCADA system operation. Our experiences with different SCADA cybersecurity test beds show that the overheads introduced by traffic monitoring are sufficiently low, keeping the intrusiveness of our methods to the minimum.

- To enhance the resilience of ICS field devices under payload attacks, we design a firmware-

level detection mechanism against malicious modifications to control logic of programmable logic controllers (PLCs). In addition to complementing existing approach that introduces at least one additional code analyzer, our approach can further enhance PLC resilience for the following reasons:

- ▷ Our approach models the runtime behaviors of legitimate control programs and compares the behaviors of the currently active control program against the model. In contrast to existing methods based on linear temporal logic, our approach is able to detect attacks on real-time performance requirements of control programs. This capability is extremely important in industrial control systems because failures to execute real-time tasks typically cause severe physical damages.
- ▷ Our approach introduces firmware enhancements with acceptably low runtime and memory overheads. The proposed enhancements can be implemented in many existing PLCs on the market, making it possible to enhance the resilience of existing ICSs through firmware upgrades.

Our detection mechanism servers as the last line of defense against PLC payload attacks, both complementing existing methods based on automated code review and formal logic without introducing extra apparatus that may become the target of new attacks.

By developing analytical methods for deterministic delay performance analysis and enhancing the cybersecurity of cyber-physical systems, this dissertation aims to help address concerns over the emerging networked cyber-physical system paradigm in various industrial application domain such as the smart grid.

## 1.4 Dissertation Organization

Focusing on delay performance analysis and cybersecurity of smart grid infrastructure, this dissertation is structured as follows.

In Chapter 2, we study the delay performance of Ethernet-based substation communication networks (SCNs). Observing the fact that traffic patterns imposed by practical automation and protection applications are generally non-feedforward, we propose an approach combining

network delay measurements and deterministic network calculus to the analysis and evaluation of worst-case delay performance of SCNs. We introduce a measurement-based method of extracting service characteristics of Ethernet switches and empirically show that the switch models constructed from measurements can be leveraged to transform non-feedforward traffic patterns into feedforward ones. Moreover, we also leverage measurement data to refine intermediate arrival curves (IACs) and show that such refinements can significantly tighten the delay bounds. In addition to studying Ethernet-based SCNs, we also discuss the application of our approach to other cyber-physical system applications.

Chapter 3 examines wireless PRP network infrastructure that are well-suited for industrial control systems requiring both deterministic delay performance guarantees and link redundancy. Instead of applying the approach developed in Chapter 2, we propose the introduction of stopped sequences, which enables us to directly derive closed-form expressions of worst-case bounds on network-induced delays. Our results show that the proposed method can be leveraged by NCPS architects in both network design and performance evaluation, making it a valuable tool for delay-sensitive and/or time-critical NCPS design.

In Chapter 4, we design machine learning algorithms to protect SCADA networks from various cyber threats such as P2P botnets and specialized attacks on SCADA network protocols. Our proposed methods leverage the traffic monitoring capability of existing networking devices in SCADA systems and will not interrupt normal system operation. By analyzing connectivity-based and flow-based traffic characteristics, we are able to detect various P2P botnets with high accuracy. In addition, a deep-learning-based algorithm is devised to detect conventional and specialized network attacks with high accuracy and very few false positives. Features of our solution makes it well-suited for intrusion detection in SCADA environments.

Chapter 5 proposes a firmware-level detection mechanism against modification attacks on PLC control programs. By modeling the runtime behaviors of legitimate control programs and collecting runtime data in the PLC firmware, our approach can identify modified control programs with runtime behaviors deviating from the established models. Our experiments show that the execution time and memory overheads of our proposed approach are sufficiently low for mainstream PLCs on the market, making it possible to enhance the resilience of existing

industrial control systems via firmware upgrades. Furthermore, our approach is capable of detecting attacks on real-time performance requirements for control programs, which cannot be detected by existing approaches based on linear temporal logic.

Finally, Chapter 6 concludes this dissertation and discusses possible future directions.

# Chapter 2

# Delay Performance Analysis of Substation Communication Networks (SCNs) Based on IEC 61850

## 2.1 Introduction

To take advantage of modern technologies, such as switched Ethernet, power substation automation systems (SASs) across the globe are being modernized by adopting international standard IEC 61850 [31]. To ensure system reliability and responsiveness, critical information carried by a substation communication network (SCN) has to be delivered within hard time constraints. In IEC 61850-5 [62], maximum transmission times for different class of messages are explicitly specified. For instance, in a substation requiring performance classes P2 and P3, the maximum transmission time for Type 4 sampled values must not exceed 3 milliseconds [58]. The worst-case delay performance of a particular SCN is quantified by the set of worst-case network-induced delays experienced by all its traffic flows. Evidently, for networked systems controlling critical infrastructure such as the power grid, it is the worst-case rather than the average delay performance that is of practical interest to SAS designers and architects. Be-

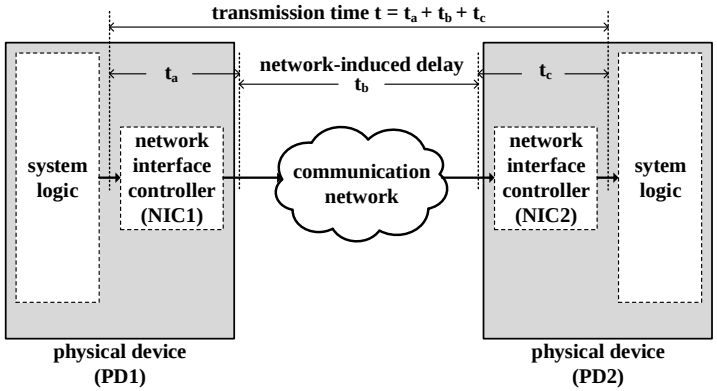



Figure 2.1: Transmission time and its components defined in IEC 61850-5.

fore an SAS project is commissioned, it is therefore of vital importance to ensure that all the worst-case delay performance requirements are satisfied.

Two major interfaces defined by IEC 61850 are the IEC 61850-9-2 [61] interface (i.e., process bus) and IEC 61850-8-1 [60] interface (i.e., station bus). Network architecture of SASs based on IEC 61850 has been presented in several research projects, such as [31], [78] and [137]: An IEC 61850-9-2 process bus carries sampled values (SVs) from synchronized merging units (MUs), which monitor a section of a power system through multiple current and voltage sensors, to various protective relays. In addition, an IEC 61850-8-1 station bus transmits generic object-oriented substation event (GOOSE) messages, greatly reducing the need for inter-equipment hardwired signals. As IEC 61850 gains momentum in the energy sector, its application has been expanded to inter-substation communication [59], as well as information exchange between substations and control centers [65]. Network engineering guidelines as well as time constraints for these emerging application scenarios are also defined [63, 64]. As illustrated in Fig. 2.1, network transmission time (i.e., network-induced delay) is one of the major components of the transmission time defined in IEC 61850-5 [62]. In essence, protective relays (also known as intelligent electronic devices) rely on sampled values delivered by the process bus to infer system status and collaboratively perform assorted control/protection tasks. Hence, it is necessary to analyze the worst-case (i.e., maximum) network-induced delays experienced by sampled value messages (SVMs), which is dependent on various factors such as network topology and traffic characteristics.

At the bare minimum, an IEC 61850-9-2 process bus network organizes multiple merging

units and transmits their sampled value messages to one or multiple protective relays. Synchronization among merging units can be achieved via hardwired signals or via precision time protocol (PTP) messages [56]. Similarly, inter-equipment signals among protective relays may be transmitted via hardwired signals, a separate station bus network, or over the same network implementing the process bus [51]. In this work, we focus on finding the worst-case network-induced delays for sampled value messages on a process bus network where synchronization and inter-equipment GOOSE signals are transmitted out-of-band [58]. Even under such settings, the traffic pattern of a process bus network is generally non-feedforward because SVMs from synchronized merging units are typically broadcast or multicast to their subscribers (e.g., protective relays) via a process bus consisting multiple Ethernet switches. As state-of-the-art worst-case delay analyses target feedforward networks (e.g., [19,114]), it is infeasible to directly adopt these techniques to analyze IEC 61850-9-2 process bus networks.

In this chapter, we propose an approach to deriving the accurate upper bounds on the worst-case delays for IEC 61850-9-2 process bus networks with arbitrary (i.e., either feedforward or non-feedforward) traffic patterns. In essence, our proposed approach is a combination of measurements and network calculus: Non-queuing delays introduced by Ethernet switches are modeled through measurements, whereas network calculus is applied to find worst-case delay bounds on network-induced delays consisting of both queuing and non-queuing delay components. The contributions of our work are as follows:

1. To obtain realistic delay bounds that are sufficiently tight using network calculus, we establish service models of Ethernet switches by taking proper measurements and exploiting the fact that Ethernet switches have sufficient capacity to support multiple simultaneous interconnections. In addition, we construct accurate traffic models for merging units, leveraging both their synchronized operations and the serialization effect of Ethernet switches.

2. Instead of leveraging conventional techniques in network calculus to deal with non-feedforward networks, we show that a non-feedforward traffic pattern on switched Ethernet can be converted into feedforward ones, facilitating the applications of existing analytical techniques targeting feedforward networks.

3. To make it less laborious to evaluate SCN design alternatives, we propose a hybrid approach that takes measurements only during switch modeling and relies on the theory of network calculus to analytically evaluate different SCN designs.

4. Our case studies of both feedforward and non-feedforward process bus networks show that the proposed combination of measurement and network calculus produces accurate delay bounds that can be validated against measurements.

We envision that our proposed approach can be utilized by SAS architects as a tool for evaluating worst-case delay performance at various stages of system design.

## 2.2 Related Work and Background

### 2.2.1 Evaluations and Analyses of Delay Performance of SCNs Based on IEC 61850

The importance of ensuring delay performance conformance of SCNs based on IEC 61850 has motivated investigations exploiting a variety of methods. In [116], discrete-event simulation is conducted to verify the delay performance of SCNs. To facilitate the study of SCNs with different traffic patterns and/or network topology, simulation models of merging units and protective relays are proposed and constructed. In [58], measurements are taken from merging units in a real transmission substation. Delay performance of a process bus network with a feedforward traffic pattern is then investigated in controlled lab environment. Both discrete-event simulation and measurement-based experiment allow us to study the delay performance of a particular SCN. However, it is hard to generalize the knowledge gained from such case-specific evaluations: When changes need to be made to a particular SCN, new experiment has

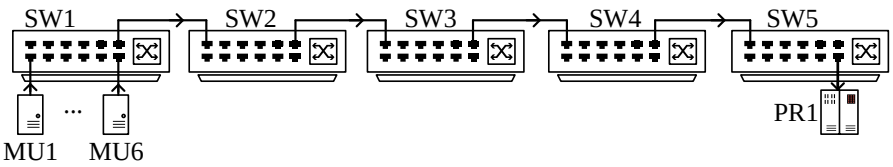



Figure 2.2: Feedforward process bus network with six merging units (MUs), one protective relay (PR), and five Ethernet switches (SW).

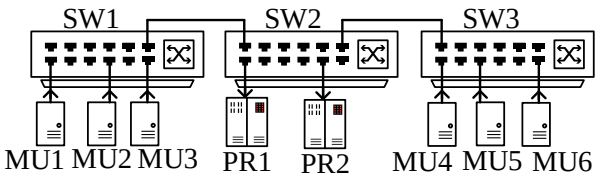



Figure 2.3: A three-switch non-feedforward process bus network.

to be set up to re-evaluate the worst-case delay performance because network-induced delays is usually dependent on network topology and device configurations (e.g., data rates). Furthermore, results obtained from these methods are sometimes not conclusive since boundary or extreme scenarios may still be overlooked even after extensive experiments/simulation.

To find out the upper bounds on worst-case delays, network calculus is applied in [42]. The derived delay bounds are then validated against simulation results, showing the feasibility of employing network-calculus-based analysis in SAS design. However, the SCNs analyzed in [42] consist of a single Ethernet switch, whereas the actual design of an SCN (e.g., the number of switches to use and the topology of the network) depends on various physical constraints (e.g., the locations and number of current and voltage sensors [137]). The traffic patterns on SCNs can be complex and non-feedforward, whereas the toolbox [15] used in [42] is designed for feedforward networks. An approach to analyzing the worst-case delay performance of non-feedforward traffic patterns on SCNs is yet to be devised.

### 2.2.2 Feedforward vs. Non-Feedforward Networks

Given a network with multiple traffic processing nodes, we can assign unique integers as node identifiers. We say that a network has a feedforward traffic pattern if the paths traveled by all its traffic flows can be represented by a set of monotonically increasing sequences of node identifiers [19]. Take the multiple-switch process bus network depicted in Fig. 2.2 as an example. If we model each switch as a network node and assign integer identifiers as shown in Fig. 2.2, the traffic pattern is feedforward because traffic flows from all merging units traveled through the same path, which can be represented by the sequence (SW1, SW2, SW3, SW4, SW5).

On the other hand, we note that SCNs designed in existing SAS projects (e.g., [137]) ge-

nerally carry non-feedforward traffic patterns. For instance, the process bus network depicted in Fig. 2.3 collects sampled values from two groups of merging units separate from each other and far away from the control room, where protective relays are deployed. Suppose that both protective relays subscribe to the sampled values published by all the merging units and that SVMs are broadcast over the process bus, the traffic pattern is non-feedforward because there is always at least a path that has to be represented by a non-increasing sequence. It should be noted that, for the network in Fig. 2.3, we can interchange the identifiers for SW2 and SW3 so that the flows from MUs $4\sim6$ to PR1 can be represented by an increasing sequence. However, the sequence representing the path for the flows from MUs $4\sim6$ to MUs $1\sim3$ is still not monotonically increasing. Such a non-feedforward traffic pattern is tricky to analyze in that a proper starting point for deduction cannot be found: If we want to find the delay induced by SW1 for SVMs sent by MUs $1\sim3$, we need to assess the "interference" of the SVMs arriving at SW1 from MUs $4\sim6$. This in turn requires us to analyze the SVM flows from MUs $4\sim6$ starting from SW3, which leads us back to MUs $1\sim3$ because their passage through SW3 introduces similar interference.

We note that non-feedforward traffic patterns are the most general ones that can be found on switched Ethernet: The path of any traffic flow should not form a cycle since it makes no sense for a device to transmit its own messages back to itself. Once a message is received by a certain sink node, it may be replayed back to the original source, but this scenario can be modeled by two flows with the same (or similar) traffic profile passing through the same set of nodes in forward and inverse orders. Although non-feedforward scenarios can be studied via simulation or measurements, an analytical approach suitable for IEC 61850-9-2 process bus network can significantly reduce the effort on measurements/simulation and is yet to be developed.

### 2.2.3 Worst-Case Delay Analysis Using Network Calculus

Based on min-plus algebra, network calculus [16, 26] provides a set of models and theorems on the worst-case delay and backlog performance of communication networks. For feedforward networks with multiple nodes and flows, the problem of tightening the delay bounds derived

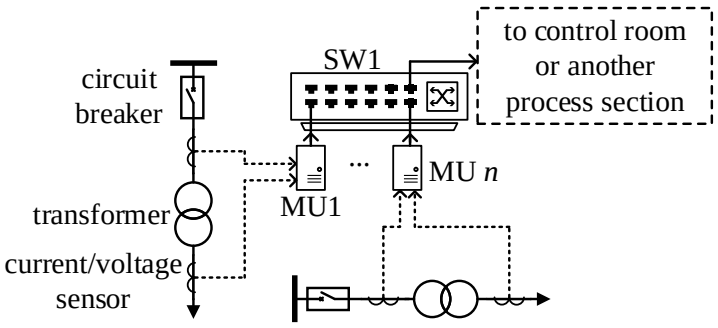



Figure 2.4: A group of $n$ merging units monitoring a section of a power system process.

using network calculus is of practical interest for researchers of real-time networked systems. Two categories of analytical methods, namely compositional methods and optimization-based methods, have been proposed [12]. A compositional method is essentially a set of rules for applying various network calculus theorems along the path traveled by a particular traffic flow (also termed as the flow of interest). Several well-known compositional methods are proposed and discussed in [112, 114]. An optimization-based method finds the worst-case delay by solving multiple optimization problems with constraints derived from network-calculus theorems and definitions. Several optimization-based methods targeting network nodes with different service disciplines, such as arbitrary multiplexing [18, 112], first-come first-served [19], and fixed priority [21], have been proposed. Optimization-based approach achieves tighter bounds by exhaustively searching for the maximum possible delays within constrained trajectory spaces. Our analysis in this work falls into the category of compositional methods, which typically have low computational complexity. We show that delay bounds can be tightened through taking proper measurements for process bus networks and exploiting the traffic characteristics of merging units.

The fact that it is hard to find a starting point for deduction on a non-feedforward network has been observed in classic network-calculus work, such as [26], which also demonstrates that simple non-feedforward networks can be analyzed by introducing stopped sequences. This technique is applied in [8] to analyze software-defined networks where the traffic patterns are also non-feedforward. However, applying this approach to non-feedforward networks with broadcast/multicast sources can be intricate because the number of stopped sequences to be introduced grows rapidly with the number of nodes and sources. To address this issue,

this work takes a measurement-based approach that properly models Ethernet switches and converts a non-feedforward network into feedforward ones.

## 2.3 Modeling Traffic Flows from Merging Units

In an IEC 61850-9-2 process bus network, merging units are synchronized to ensure that they are able to carry out sampling operations simultaneously and generate a "snapshot" of the power system. For a power system operating at 60 Hz, the length of its system cycle is simply $\frac{1}{60}$ seconds. During each system cycle, a merging unit perform sampling operations at its configured sampling rate (e.g., 80 samples per cycle). The flow of sampled value messages generated by a particular merging unit $i$ can thus be represented by a real-valued, non-negative, non-decreasing function $F_i(t)$, which is the cumulative traffic volume observed from the Ethernet output interface of $i$ up to time instant $t$. $F_i(t)$ is called the arrival process of the flow generated by $i$. Without loss of generality, we define any arrival process $F(t)$ in this work for $t \geq 0$ and assume that $F(0) = 0$. In network calculus [16, 26], a bounding traffic model known as arrival curve, is defined to characterize arrival process:

**Definition 2.3.1.** *Given an arrival process $F(t)$, a real-valued, non-negative, non-decreasing function $\alpha(t)$ defined for $t \geq 0$ is an arrival curve of $F(t)$ if and only if*

$$\forall t \geq s \geq 0 \colon F(t) - F(s) \leq \alpha(t - s).$$

If $\alpha(t)$ is an arrival curve of $F(t)$, we write $F(t) \sim \alpha(t)$. A commonly used form of arrival curve is the leaky-bucket arrival curve [26, 112] $\alpha(t) = \sigma + \rho \cdot t$, where $\rho$ is the average rate component and $\sigma$ the burstiness component. If an arrival process $F(t)$ has an arrival curve $\alpha(t) = \sigma + \rho \cdot t$, we write $\alpha(t) \sim (\sigma, \rho)$.

### 2.3.1 Simultaneous Arrival and Traffic Aggregation

Although arrival curves of traffic flows generated by individual sources are typically constructed separately, we propose to construct arrival curves for different groups of merging units in IEC 61850-9-2 process bus network to exploit its following characteristics:

1. Merging units are synchronized and generate sampled values simultaneously.

2. A group of merging units are typically deployed to collectively monitor a section of a power system process.

3. At each section of the system process, data from the merging units can be collected by an Ethernet switch provided that it has an adequate number of network interfaces.

4. Merging units of the same group are configured to work at the same sampling rate and generate sampled value messages of the same size.

Let us consider the process section depicted in Fig. 2.4. Since the $n$ merging units are synchronized, they generate sampled value messages nearly simultaneously at the beginning of each sampling cycle. By connecting them to an Ethernet switch, their sampled value messages are put onto the process bus network for transmission. Suppose that the size of sampled value messages is $L$ and that the length of the sampling cycles is $T$. For any individual sampled value message flow $F_i(t)$ generated by merging unit $i$ $(1 \leq i \leq n)$, we have $F_i(t) \sim (L, \frac{L}{T})$, which is depicted in Fig. 2.5a. To find the arrival curve of the aggregate output from the $n$ merging units in Fig. 2.4, the multiplexing/aggregation theorem [26] can be leveraged:

**Theorem 2.3.2.** *Given a set of $n$ arrival processes $F_1(t), F_2(t), \ldots, F_n(t)$ and their respective arrival curves $\alpha_1(t), \alpha_2(t), \ldots, \alpha_n(t)$, we always have*

$$\Sigma_{i=1}^{n} F_i(t) \sim \Sigma_{i=1}^{n} \alpha_i(t).$$

Thus, the arrival curve for the aggregate output of $n$ merging units is simply $\Sigma_{i=1}^{n} F_i(t) \sim (n \cdot L, \frac{n \cdot L}{T})$. We note that for synchronized merging units generating sampled values simultaneously, the arrival curve of their aggregate output given by Theorem 2.3.2 is tight. We term this phenomenon as simultaneous arrival, which is illustrated in Fig. 2.5b. Since transmission time for all sampled values must not exceed 3 ms [58], it is unnecessary to distinguish individual traffic flows generated by MUs of the same group: The worst-case delay experienced by the aggregate flow upper bounds that of any individual flows. In other words, simultaneous arrival facilitates worst-case delay analysis with aggregate flows.

### 2.3.2 Arrival Curve of Serialized Switch Output

After $n$ simultaneously generated sampled value messages pass through switch SW1 in Fig. 2.4, they will be serialized and put onto the output interface connecting to another switch or a locally deployed protective relay. The inter-frame gap specified by the Ethernet specification is 12 bytes, which is much smaller than the size of a sampled value message (e.g., 126 bytes according to [58]). For 100 Mbps (Fast) Ethernet, this inter-frame gap translates into a $\frac{12\times 8 \text{ bits}}{100 \text{ Mbps}} = 0.96$ $\mu$s time interval. Hence, the arrival curve $\alpha(t) = n \cdot L + \frac{n \cdot L}{T} \cdot t$ is not strictly tight (in other words, slightly loose) for the output of SW1, but it is still a good approximation to its tight arrival curve.

In our proposed approach, we always group together merging units connected to the same switch to exploit simultaneous arrival. This is because synchronized merging units monitoring the same power system section typically use the same sampling rate. For the corresponding switch output, we still use the arrival curve of the aggregate input as an approximation. As we will see in Sec. 2.5.1, accurate arrival curves can help us tighten the delay bounds.

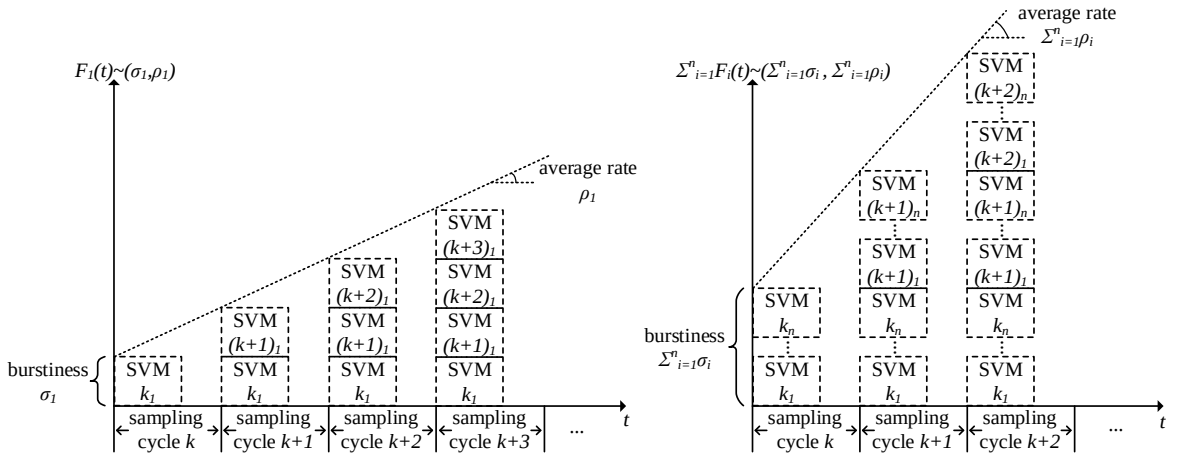



Figure 2.5: Arrival processes and arrival curves of an individual merging unit (MU1) as well as a group of $n$ merging units.

## 2.4 Modeling Ethernet Switches

In network calculus, traffic processing capability of a networking device is modeled by another bounding model, which is known as service curve [16, 26]:

**Definition 2.4.1.** *Given a traffic processing network node with input arrival process $F^{in}(t)$ and output arrival process $F^{out}(t)$, a real-valued, non-negative, non-decreasing function $\beta(t)$ is the service curve of the node if and only if*

$$\forall t \geq 0 : F^{out}(t) \geq \inf_{0 \leq s \leq t} \{F^{in}(t) + \beta(t-s)\} \equiv (F^{in} \star \beta)(t),$$

*where $[\cdot]^+$ denotes the operation $\max\{\cdot, 0\}$.*

The $\star$ operator is used to denote the min-plus convolution operation. A commonly-used type of service curve is the rate-latency service curve $\beta(t) = R \cdot [t-T]^+$, where $R$ is the processing capacity and $T$ models constant non-queuing delay. For a rate-latency service curve $\beta(t) = R \cdot [t-T]^+$, we introduce the shorthand $\beta(t) \leftrightarrow (R, T)$.

Sampled values may need to pass through a series of Ethernet switches to reach a subscriber. Take the process bus network depicted in Fig. 2.2 as an example. The sampled values travel through five switches in tandem in order to reach the protective relay. The concatenation theorem [26] is well-suited to model such a tandem of switches:

**Theorem 2.4.2.** *Suppose that a flow $F(t)$ passes through $m$ network nodes in tandem and that the service curves offered by these nodes are $\beta_1(t), \beta_2(t), \ldots, \beta_m(t)$, respectively. The service curve $\beta(t)$ offered by the tandem of these $m$ nodes to $F(t)$ is*

$$\beta(t) = (\beta_1 \star \beta_2 \star \ldots \star \beta_m)(t) \equiv \Pi_{j=1}^{m} \beta_j(t).$$

To apply this theorem to a process bus network, we need to first find out the rate-latency service curves offered by individual Ethernet switches.

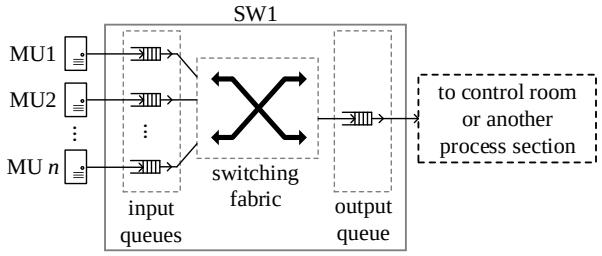



Figure 2.6: A group of $n$ merging units connecting to Ethernet switch SW1.

### 2.4.1 Identifying Rate Component for the Service Curve of Process Bus Ethernet Switch

For an Ethernet switch, we use the rate component of its service curve to capture its traffic processing capacity. Let us consider the single-switch scenario in Fig. 2.4, which is re-drawn in Fig. 2.6. For sampled value messages from merging unit $i$, the entry point into switch SW1 is the input interface connecting $i$. We assume that the data rate of the output interface of $i$ matches that of the input interface of SW1, which is typically the case. The switching fabric provides dedicated interconnections between input and output interfaces of SW1, allowing different pairs of input/output interfaces to communicate simultaneously. However, as shown in Fig. 2.6, the traffic pattern of a process bus network inevitably creates a bottleneck at the output interface: Suppose that the switching fabric offers sufficient capacity. As the number of merging unit increases, the output interface connecting to another switch or a protective relay will eventually overflow.

Suppose that the input queue, switching fabric, and output queue, are modeled by three work-conserving links [26] with rate-latency service curves $\beta_{\text{in}}(t) \leftrightarrow (R_{\text{in}}, T_{\text{in}})$, $\beta_{\text{sw}}(t) \leftrightarrow (R_{\text{sw}}, T_{\text{sw}})$, and $\beta_{\text{out}}(t) \leftrightarrow (R_{\text{out}}, T_{\text{out}})$, respectively. Theorem 2.4.2 suggests that the service curve $\beta(t)$ offered by the tandem of these three work-conserving links is

$$\begin{aligned}\beta(t) &= (\beta_{\text{in}} \star \beta_{\text{sw}} \star \beta_{\text{out}})(t) \\ &= \min\{R_{\text{in}}, R_{\text{sw}}, R_{\text{out}}\} \cdot [t - (T_{\text{in}} + T_{\text{sw}} + T_{\text{out}})]^{+}. \end{aligned} \tag{2.1}$$

According to the specifications of Ethernet switches on the market, their switching fabrics are capable of supporting all interfaces to communicate at wire rate in full-duplex mode (in

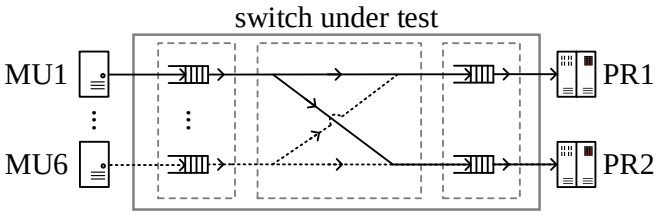



Figure 2.7: Experiment settings to extract latency component of the service curve of the switch under test.

other words, perfect parallelism is offered by the switching fabric and it will not become a bottleneck). Therefore, we have $\min\{R_{\text{in}}, R_{\text{sw}}, R_{\text{out}}\} = R_{\text{out}}$. In process bus networks, the rate component of the service curve offered by an Ethernet switch to a merging unit can thus be the wire rate of the output interface traveled by its sampled values. For a multicast or broadcast flow, an Ethernet switch replicates the flow in its switching fabric, so multiple service curves are offered by the switch, corresponding to all the directions it travels (i.e., for all the output interfaces that process the broadcast/multicast flow).

### 2.4.2 Extracting Latency Component for Service Curve from Measurements

From Eq. 2.1, we also find that the latency component of the service curve offered by a switch is simply the non-queuing latencies induced by its internal components. Note that delays introduced by other necessary physical components, such as Ethernet cables, should also be included in practice. In our approach, we rely on network calculus to find the worst-case delays caused by queuing. However, we still need to factor in non-queuing delay components, which are hard to determine merely based on technical specifications of switches. The latency component models several categories of non-queuing delays, including propagation delay on Ethernet cables, serialization delay at input/output interfaces, as well as processing delays (e.g., checksum verification) caused by various internal components of an Ethernet switch.

In our approach, we propose to capture non-queuing delays by taking measurements: By injecting sampled value messages at low rates and measuring the delays they experience, we use the maximum delay observed as a good approximation to the latency component because few messages are enqueued under light workloads and they are dequeued rapidly. Note that such

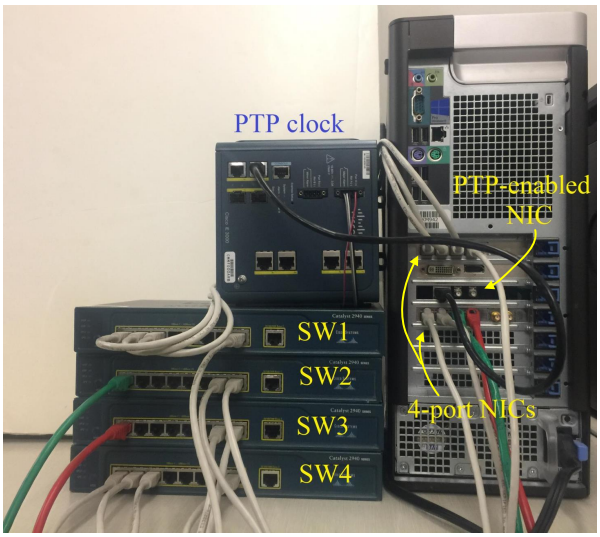



Figure 2.8: Components and organization of our test bed.

a measurement-based approach essentially relies on the assumption that additional delays observed during instantaneous traffic burstiness (e.g., simultaneous arrivals of SVMs) are caused by queuing. If this assumption holds for Ethernet switches (at least under normal workloads observed on process bus networks [58]), measuring non-queuing delays intrinsic to the switches will enable us to find an appropriate latency component for their service curves. To verify the validity of this assumption, we conduct experiments on four Ethernet switches of the same model. All the switches come with eight full-duplex 100 Mbps interfaces and one full-duplex 1 Gbps interface. For all the experiments conducted in this work, we only utilize the 100 Mbps interfaces.

### Test Bed Settings

The components and organization of our test bed is shown in Fig. 2.8. We install a PTP-enabled network interface card [123] on a computer server and connect it to a PTP master clock. Two four-port network interface cards (NICs) are installed on the server to inject and receive timestamped packets. To improve system throughput, we write our packet sender and receiver programs using the netmap fast packet I/O framework [107]. To measure the delay experienced by a packet, timestamps are generated when the packet is sent and received. The difference between the two timestamps is used to represent the delay induced by the network.

Table 2.1: Delay measurements from one of the switches under test (SUT) serving MU1 or MU2 in microseconds.

| SVMs per second | MU1 | | | MU2 | | |
|---|---|---|---|---|---|---|
| | min. | avg. | max. | min. | avg. | max. |
| 1 | 15.2 | 16.3 | 17.2 | 15.3 | 16.1 | 17.1 |
| 10 | 15.1 | 15.9 | 16.7 | 14.6 | 16.0 | 17.3 |
| 100 | 16.0 | 16.6 | 16.9 | 15.5 | 16.1 | 17.4 |
| 1000 | 15.7 | 16.2 | 17.0 | 14.9 | 15.8 | 16.9 |
| 2000 | 15.4 | 16.3 | 17.1 | 15.4 | 16.6 | 17.6 |
| 3000 | 15.2 | 16.3 | 17.5 | 15.8 | 16.2 | 17.0 |
| 4000 | 14.9 | 16.1 | 17.4 | 15.3 | 15.9 | 17.4 |
| 4800 | 15.6 | 16.5 | 17.6 | 15.1 | 16.4 | 16.8 |

**Measurement Procedures**

We inject sampled value messages into the input interface of an Ethernet switch to impose the traffic pattern shown in Fig. 2.7. The length of the sampled value messages is 126 bytes, and the message layout is detailed in [58]. To emulate broadcast traffic pattern on process bus, we set to destination MAC address of all injected messages to $FF:FF:FF:FF:FF:FF$. For each switch under test (SUT), we take the following measurements step by step:

1. Choose the SVM flow generated by merging unit $i$ as our flow of interest, where $i \in \{1, 2, \ldots, 6\}$. Inject SVMs at an initial rate, which should be low enough. All the other MUs are not activated (i.e., they do not generate SVMs and simultaneous arrival at the SUT is avoided). In our experiment, we start with one SVM per second. This step is run for five minutes and the delays experienced by all the injected SVMs are recorded.

2. Increase the data rate of the selected MU and then repeat Step (1). We increase the data rate up until $60 \times 80 = 4800$ SVMs per second, which is the data rate used by the MUs in the process bus networks studied in our case studies.

3. Choose another flow of interest and repeat Steps (1) and (2) until interfaces utilized by all the MUs are tested.

The measurement results for one of the SUTs, with MU1 or MU2 activated separately, are summarized in Table 2.1. We observe that the variances of the delays introduced by the

SUT at different data rates are small. At each rate, the majority of the measured values are close to the mean. The results observed on interfaces utilized by the other MUs are identical to those of MU1 and MU2. Similarly, if we change the interfaces utilized by MUs and relays (e.g., interchanging the interfaces for MU1 and PR1) while maintaining the same traffic pattern between sources (i.e., MUs) and sinks (i.e., relays), the distributions of the delays observed have similar characteristics. In addition, if an extra MU is activated to emulate simultaneous arrival, the worst-case delays increase significantly (see Sec. 2.5). Furthermore, this observation is consistent across all four SUTs. The fact that the maximum delays do not increase significantly at different data rates suggests that the SUTs offer sufficient processing capacity under the imposed workloads. The maximum latencies observed for all four SUTs are about the same, so we use the maximum value from Table 2.1, i.e., 17.6 $\mu$s, as the latency component for the service curves of all the SUTs. We choose the maximum value here because Theorem 2.5.1 requires that the service curve lower bounds the processing capacity of a switch. Choosing an aggressively small latency component may lead to underestimating the delay bounds.

It should be noted, however, different values may be chosen for individual switches if there are significant differences among the maximum latencies observed (e.g., the Ethernet switches employed by an SCN come from different manufacturers or have different switching fabric implementations). Moreover, we do not consider the scenarios where MUs inject SVMs at wire rate because some of the SVMs are discarded by the output interfaces of switches due to buffer overflow. The dropped SVMs effectively experience infinite network-induced delays. In a process bus network, we pay special attention not to cause SVM drops because successful delivery of SVMs is of critical importance to the implementation of protection and control tasks. Additionally, we note that Theorem 2.5.1 is not applicable under such scenarios because the assumption of infinite buffer size is violated (see Sec. 2.5.1).

## 2.5 Evaluation on A Feedforward Single-Switch Process Bus

### 2.5.1 Worst-Case Delay Analysis of Feedforward Process Bus Networks

Once arrival curve of aggregate input to an Ethernet switch and its service curve are constructed, the classic result from network calculus gives the upper bound on the worst-case delay incurred by the switch [16, 26]:

**Theorem 2.5.1.** *Suppose that a traffic flow $F^{in}(t) \sim \alpha(t)$ passes through a network node with service curve $\beta(t)$ and an infinite-size buffer and that the corresponding output flow is represented by $F^{out}(t)$. The delay experienced by the data bit entering the node at time $t$ is denoted by $d(t) = \inf\{u \geq 0 : F^{in}(t) \leq F^{out}(t+u)\}$ and we have*

$$d(t) \leq \inf\{u \geq 0 : \alpha(s) \leq \beta(s+u), 0 \leq s \leq t\} \equiv h(\alpha(t), \beta(t)).$$

According to this theorem, the delay bound is simply the maximum horizontal distance between the arrival curve and the service curve, i.e., $h(\alpha(t), \beta(t))$. Note that this theorem is only applicable if the buffer of the node can accommodate all the incoming data at all times. To ensure that such an assumption of infinite buffer size holds in practice, we check the input and output traces in our experiments to ensure that no SVMs are dropped. We observe that our switches occasionally discard SVMs when the data rate at its output interface is close to wire rate. Such scenarios are avoided in our experiments since a process bus network design leading to SVM loss should be further refined before commissioning.

Theorem 2.5.1 also explains why we want to obtain tight arrival curves (see Sec. 2.3) and service curves (see Sec. 2.4). A loose arrival curve results in overestimating the worst-case delays and so does an overly tight service curve (e.g., naively assuming that the latency component is zero). By constructing realistic arrival and service curves, existing methods for

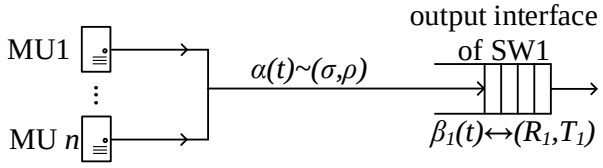



Figure 2.9: Simplified network diagram of the single-switch process bus network in Fig. 2.6.

worst-case delay analysis, such as those reviewed in Sec. 2.2.3, can be applied to feedforward SCNs and generate accurate delay bounds.

### 2.5.2 Worst-Case Delay of a Single-Switch Process Bus Network

To validate the quality of the delay bounds generated by our proposed combination of network calculus and measurements, we first study the simple single-switch process bus network depicted in Fig. 2.4 and Fig. 2.6 with up to seven merging units. Note that the traffic pattern on such a process bus is always feedforward because there is only one possible path with a single node.

#### Network-Calculus-Based Analysis

As proposed in Sec. 2.3.1, we construct the arrival curve for the aggregate input to SW1. We use our test bed to emulate $n$ merging units monitoring a 60-Hz power system at 80 samples per cycle, for $1 \leq n \leq 7$. The traffic pattern of this process bus can be simplified into the network diagram shown in Fig. 2.9. The length of the SVMs we use is 126 bytes. To use our traffic model, we take the 8-byte preamble and 4-byte CRC trailer required by Ethernet frames into account. The effective message size is hence $L = (8+126+4) \times 8 = 1104$ bits, and the length of the sampling cycle is $T = \frac{1}{60 \times 80} \approx 208.33 \mu$s. Therefore, the burstiness component of the aggregate input is $\sigma = n \times 1104$ bits, and the average rate is $\rho = \frac{n \times 1104}{208.33}$ Mbps. According to our discussion in Sec. 2.4.1 and Sec. 2.4.2, we take $R_1 = 100$ Mbps and $T_1 = 17.6$ $\mu$s for SW1. From Theorem 2.5.1, we find that the worst-case delay experienced by SVMs sent by any of the $n$ MUs is given by

$$d_{\max} = T_1 + \frac{n \cdot L}{R_1}. \tag{2.2}$$

#### Measurement-Based Evaluation

In our evaluation, we start with $n = 1$. After injecting emulated SVMs and recording the delays for 10 minutes, we increase $n$ by 1 and repeat the experiment until $n = 7$. Fig. 2.10 compares the measured delay values against the analytically derived bounds given by Eq. 2.2. Our results show that the delay bounds derived using our proposed combination of network

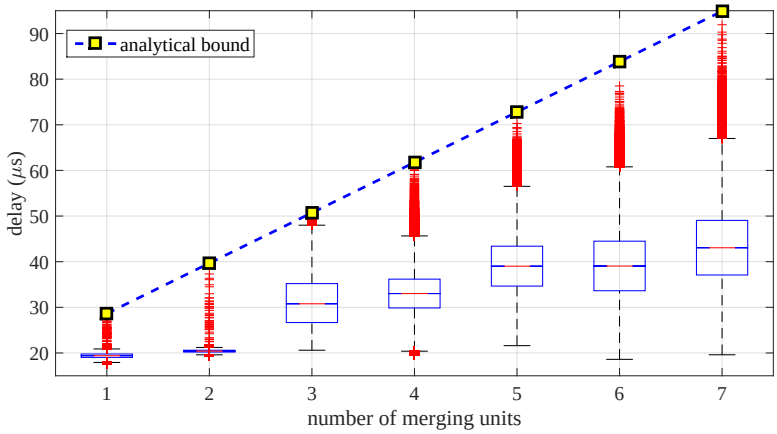



Figure 2.10: Analytically derived delay bounds and measurement results for the single-switch process bus network in Fig. 2.6 with different numbers of merging units activated.

calculus and measurements are sufficiently tight: For all tested values of $n$, the analytical bounds are at most 3.5% larger than the maximum delays we observe. We note that the traffic pattern carried by this process bus is rather simple, and the rest of this chapter will focus on non-feedforward traffic patterns.

## 2.6 Converting A Non-Feedforward Network into Feedforward Ones

As sampled value messages from multiple merging units converge toward Ethernet switch output interfaces connecting protective relays or other switches, we can model an individual Ethernet switch by $m$ work-conserving links, each of which corresponds to one of its utilized output interfaces. Take the process bus illustrated in Fig. 2.3 as an example. We re-draw this network into the simplified network diagram in Fig. 2.11 using the modeling technique in Sec. 2.4.1. Note that most of the paths among merging units are omitted. Only the path from MUs 4∼6 to MU1 is shown for the purpose of discussion. By introducing multiple work-conserving links, the SVM flows within the switching fabric of each switch are completely decoupled because the bottlenecks are the output interfaces. If we assign unique integer identifiers to the work-conserving links, it is now possible to find an assignment scheme such that all the paths can be represented by monotonically increasing sequences. For instance,

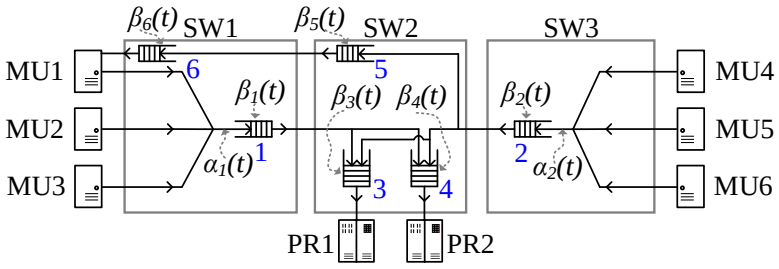



Figure 2.11: Network diagram of the process bus in Fig. 2.3, simplified for worst-case delay analysis.

suppose that we use the assignment scheme shown in Fig. 2.11. The path (SW3, SW2, SW1) in Fig. 2.3 can now be represented by the increasing sequence (2, 5, 6), and it is evident that the non-feedforward traffic pattern is converted into a feedforward one.

In switch Ethernet, each switch interface is utilized by a particular device (e.g., merging units, protective relays, and another switch). Therefore, the following theorem can be proved for switched Ethernet:

**Theorem 2.6.1.** *By introducing work-conserving links for individual switch output interfaces, any non-feedforward traffic pattern on a switched Ethernet can be converted into a feedforward one.*

*Proof.* To prove that the network is feedforward after introducing work-conserving links, all we need to show is that there exists an integer identifier assignment scheme such that all paths of the originally non-feedforward network can be represented by monotonically increasing sequences of identifiers. Let us represent Ethernet switches by virtual network nodes with an infinite number of embedded work-conserving links (i.e., output interfaces). For a network with $m$ switch and $n$ sinks, without loss of generality, suppose that each sink has only one input interface. Similarly, each pairwise physical connections between two switches uses up an output interface of the upstream switch. Any particular path connecting $k$ $(1 \leq k \leq m)$ nodes consists of $k$ pairwise physical connections among the nodes (i.e., $k-1$ connecting the switches and the last one connecting a switch to a sink). For any non-feedforward network that has paths that cannot be represented by increasing node (i.e., switch) identifiers, we can find an integer identifier assignment scheme for the work-conserving links that makes the network feedforward as follows. First, we choose an arbitrary path $p$ and assign integers such

that its representation is an increasing sequence. Then, we select a new path $p'$. We say that a pair of paths are in conflict if all possible assignment schemes enabling one of them to be represented by an increasing sequence make the representation of the other not monotonically increasing. If $p'$ is not in conflict with $p$, we extend the assignment scheme used for $p$ such that $p'$ is also represented by an increasing sequence. In this case, $p'$ may or may not introduce new work-conserving links. If $p'$ is in conflict with $p$, then it cannot be implemented using only the physical pairwise connections of $p$. This can be shown by contradiction: If $p'$ can also be implemented by a subset of the work-conserving links utilized by $p$, its representation must be a certain subsequence of the representation of $p$, which contradicts the definition of a conflicting path. In this case, $p'$ introduces certain new physical pairwise connections between the nodes by occupying work-conserving links that have not been utilized. Evidently, these new connections are introduced by the part of $p'$ that cannot be represented by subsequence of the representation of $p$. We can then walk along $p'$ and assign new integers to the newly utilized work-conserving links, by prepending, inserting, and/or appending new integers, and then adjust the scheme such that both $p$ and $p'$ are represented by increasing sequences. By repeating this procedure, we can eventually represent all paths by increasing sequences. □

Note that if the number of output interfaces of a switch is finite, only the number of possible paths that need to pass through this procedure is constrained. To sum up, for a process bus based on switched Ethernet with a non-feedforward traffic pattern, we can always convert it into a feedforward network by introducing one work-conserving link per utilized switched output interface: For any particular flow of interest, we start introducing work-conserving links from the last node on its path and walk backward. At each node, if the output interface utilized by the flow of interest also serves other flow(s), we also need to trace backward along the paths of these "interfering" flows and introduce work-conserving links accordingly. Since a non-feedforward traffic pattern does not have any cycle and Theorem 2.6.1 shows that such a network of work-conserving links is feedforward, we are able to reach the source nodes of these paths eventually. At this point, we obtain a network diagram allowing us to analyze the worst-case delay experienced by the flow of interest. Note that this technique has been extensively used in prior work on feedforward networks [18, 19, 21], and our proposed combination of

network calculus and measurements extends its application to non-feedforward patterns on switched Ethernet. It should also be noted that the feedforward traffic pattern created by introducing work-conserving links at switch output interfaces can be used to created multiple network diagrams for different flows of interest, with work-conserving links as network nodes.

## 2.7 Evaluation on Non-Feedforward Process Bus Networks

To evaluate the delay bounds derived by our approach for non-feedforward process bus networks, we study two multiple-switch examples. Note that the switches used in our experiments are the ones tested in Sec. 2.4.2.

### 2.7.1 Case Study I – A Three-Switch Process Bus

In this experiment, we analyze the network-induced delays for the network shown in Fig. 2.3. As discussed in Sec. 2.6, the originally non-feedforward traffic pattern is first converted to a feedforward one. The flows of our interest and notations for arrival/service curves are explicitly shown in Fig. 2.11. All six merging units are activated, and we have $\alpha_1(t) \sim (3 \cdot L, \frac{3 \cdot L}{T})$ and $\alpha_2(t) \sim (3 \cdot L, \frac{3 \cdot L}{T})$. As for service curves, we have $\beta_i(t) \leftrightarrow (R_1, T_1)$, where $R_1 = 100\ Mbps$ and $T_1 = 17.6\ \mu$s, for $1 \leq i \leq 6$.

#### Network-Calculus-Based Analysis

The worst-case delay bounds for the SVM flows from MUs $1 \sim 3$ to PR1 and PR2 as well as from MUs $4 \sim 6$ to PR1 and PR2 are the same due to symmetry. Note that an implicit assumption that enables us to exploit such symmetry is that the switching fabrics have sufficient capacity to replicate SVM messages for broadcasting/multicasting, which is verified in Sec. 2.4.2. Let us analyze the path from MUs $1 \sim 3$ to PR1. At node 1, Theorem 2.5.1 suggest that the worst-case delay it introduces is $d_1 = T_1 + \frac{3 \cdot L}{R_1}$. At node 3, we use $\alpha_1(t)$ and $\alpha_2(t)$ to approximate the arrival curves of the output of node 1 and node 2, respectively (see Sec. **??**). By applying Theorems 2.3.2 and 2.5.1, the worst-case delay introduced by node 3 is $d_3 = T_1 + \frac{6 \cdot L}{R_1}$. The worst-case delay bound for this flow is thus $d_{\max} = d_1 + d_3$.

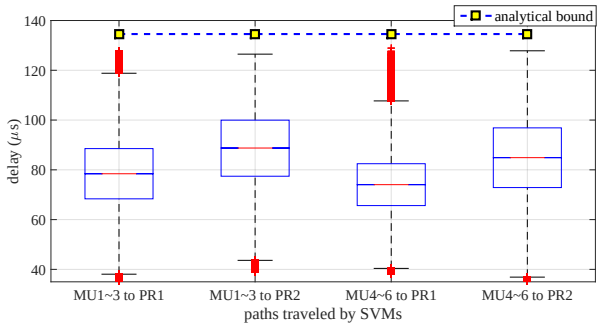



Figure 2.12: Analytically derived delay bounds and measurement results for SVM flows between merging units (MU) and protective relays (PR) on the process bus network in Fig. 2.3.

For the flow from MUs $4\sim6$ to MU1, note that nodes 2, 5, and 6 serve only this particular flow. Therefore, we can first apply Theorem 2.4.2 to find the concatenated service curve offered by the tandem of the three nodes. Then, we apply Theorem 2.5.1 and find that the delay bound for this flow is $d'_{\max} = 3 \cdot T_1 + \frac{3 \cdot L}{R_1}$. Note that this delay bound is not of practical interest in SASs, and we derive this bound simply for the purpose of evaluation.

### Measurement-Based Evaluation

We set up the network in Fig. 2.3 and let it execute for 10 minutes. For the flows between merging units and protective relays, the measured delays and the derived bounds are shown on Fig. 2.12. We observe that all measured delays are upper bounded by the analytically derived bounds, which are at most 6.4% greater than the maximum observed values. The maximum delay we observe for the flow from MUs $4\sim6$ to MU1 is 83.2 $\mu$s, whereas its derived delay bound is 85.92 $\mu$s. The results of this experiment show that network calculus provides accurate delay bounds for non-feedforward IEC 61850-9-2 process bus network once we convert it into a feedforward one.

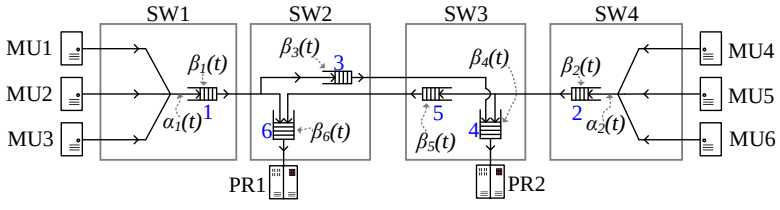



Figure 2.13: Network diagram of the four-switch process bus studied in Sec. 2.7.2.

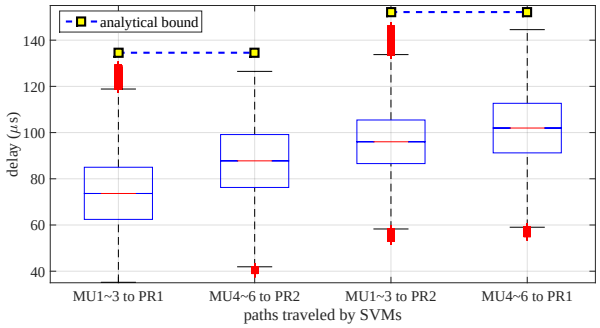



Figure 2.14: Analytical bounds and measurement results for SVM flows received by protective relays in the four-switch process bus network shown in Fig. 2.13.

### 2.7.2 Case Study II – A Four-Switch Process Bus

To further validate our proposed approach, we construct another process bus network consisting of four Ethernet switches. In contrast to the three-switch network studied in the previous case study, the SVM flow from MUs $1\sim3$ to PR2 travels through three work-conserving links (which is also the case for the flow from MUs $4\sim6$ to PR1). In [58], it is shown through measurements that inserting an additional Ethernet switch into the feedforward path traveled by SVMs introduces an extra 15-$\mu$s latency (using switch of a particular model, which is not reported). This is because SVMs simultaneously arriving at the first switch (i.e., SW1 in Fig. 2.2) are serialized and do not experience queuing delays at succeeding switches (i.e., SW2$\sim$5). This case study shows that the same observation also applies to non-feedforward traffic patterns. The network diagram of this process bus is shown in Fig. 2.13. In this experiment, we focus on worst-case delays experienced by SVMs traveling toward protective relays, and paths between merging units are all omitted. The configuration of the MUs remains unchanged. By introducing work-conserving links at the output interfaces along our flows of interest, the network in Fig. 2.13 is transformed into a feedforward one. Using the models discussed in previous sections, we have $\alpha_1(t) = \alpha_2(t) \sim (3 \cdot L, \frac{3 \cdot L}{T})$ and $\beta_i(t) \leftrightarrow (R_1, T_1)$ for $1 \leq i \leq 4$.

#### Network-Calculus-Based Analysis

Exploiting symmetry, we derive the delay bounds for the SVM flows from MUs $1\sim3$ to PR1 and PR2. First, let us analyze the flow from MUs $1\sim3$ to PR1. At node 1, Theorem 2.5.1 suggests

that the worst-case delay it induces is $d_1 = T_1 + \frac{3 \cdot L}{R_1}$. At node 6, we use the arrival curves $\alpha_1(t)$ and $\alpha_2(t)$ to approximate the serialized output from node 1 and node 5, respectively. By applying Theorem 2.3.2 and then Theorem 2.5.1, we have $d_6 = T_1 + \frac{6 \cdot L}{R_1}$.

For the flow from MUs 1∼3 to PR2, we note that it solely enjoys the service curves of nodes 1 and 3. Therefore, Theorem 2.4.2 can be applied to obtain their concatenated service curve. Then, Theorem 2.5.1 gives the worst-case delay incurred by these two nodes, which is $d_{1,3} = 2 \cdot T_1 + \frac{3 \cdot L}{R_1}$. At node 4, we again use $\alpha_1(t)$ and $\alpha_2(t)$ to approximate the serialized output from node 1 and node 2. By applying Theorem 2.3.2 and then Theorem 2.5.1, we have $d_4 = T_1 + \frac{6 \cdot L}{R_1}$.

#### Measurement-Based Evaluation

We set up the four-switch process bus using our test bed and let it execute for 10 minutes. The measured delays and the analytical bounds for the SVM flows received by the protective relays are shown in Fig. 2.14. All the measured delays are upper bounded by the corresponding analytical bounds. In fact, the delay bounds are at most 6.3% greater than the observed maximum delays. These analytical and measurement results suggest that the worst-case delay experienced by SVMs increases by about 17.6 $\mu$s if an additional switch is inserted, which is consistent with the observation for feedforward SCNs in [58]. Both case studies in this section show that our proposed combination of network-calculus-based worst-case delay analysis and measurement-based switch modeling produces realistic delay bounds that are close to measured values.

## 2.8 Discussion

### 2.8.1 Incorporating Network-Calculus-Based Analysis into SAS Project

The proposed approach complements existing SAS design tools and significantly reduces the efforts required to find out accurate worst-case delay bounds. Network-calculus-based analysis can be applied throughout multiple phases of an SAS project. At early stages of an SAS design when measurements are not available (e.g., models of Ethernet switches are not finalized

or plans for equipment procurement are pending review), arrival and service curves can be constructed from system specifications. As discussed in Sec. 2.3.1, arrival curve for the SVM flow generated by a particular merging unit can be derived based on its designed sampling rate, operating frequency of the power system, as well as the size of its SVMs. Based on site-specific constraints, such as the locations of primary power system equipment, it is also possible to determine how MUs should be grouped together. Arrival curves for aggregate SVM flows can then be found. Similarly, service curve can be roughly established by setting the latency component to 0 (i.e., temporarily using an overly tight, rate-based service curve). The analytical procedures demonstrated in Sec. 2.7 can then be applied to obtain estimated delay bounds. Although these analytical results provide insights into the worst-case delay performance of candidate design alternatives, it should be noted that the impacts of non-queuing delays are not factored in and underestimation will probably occur.

At later stages (e.g., when Ethernet switches are procured), more realistic service curves can be established by taking measurements following the steps outline in Sec. 2.4.2. In addition, arrival curves may also need to be updated to reflect system changes made after the last time network-calculus-based analysis is conducted. Using newly updated arrival and service curves, accurate delay bounds can be derived and whether the deterministic delay performance requirements specified in IEC 61850-5 [62] are satisfied can be verified analytically. For legacy SASs to be renovated with IEC 61850, service and arrival curves established in previous projects using similar devices and configuration may be exploited to analytically assess the worst-case delay performance of process bus networks to be deployed.

### 2.8.2 Applicability of the Proposed Approach

In our experiments, the maximum rate at which SVMs are injected into a particular input interface of an Ethernet switch is limited to about 5.3 Mbps. Although our evaluation shows that the analytical bounds are sufficiently tight at this rate, this does not suggest that our approach is always applicable. In theoretical treatments of network calculus [16, 26], it is usually assumed that network nodes have infinitely sized buffers. Under such an assumption, worst-case backlog bounds can be derived for individual nodes using network calculus, addressing

the problem of buffer dimensioning. In our proposed approach, Theorem 2.5.1 applies as long as the assumption of infinite buffer size holds. Suppose that the aggregate input to a network node with a service curve $\beta(t) \leftrightarrow (R, T)$ has an arrival curve $\alpha(t) \sim (\sigma, \rho)$. In general, the theoretical stability condition $\rho \leq R$ suffices to guarantee that there exists some constant $C$ that upper bounds the maximum backlog sizes of all nodes [16, 19].

However, what we observe on our switches is that certain frames will be discarded when the data rate at an output interface approaches its wire rate. To ensure that a particular SCN design does not violate the infinite buffer size assumption, simply checking the theoretical stability condition is insufficient. Measurements must be taken to establish the "safe operating regions" for different switch models in terms of maximum data rates and frame sizes at output interfaces: Using a particular frame size, we increase the data rate of a certain output interface, which can easily be achieved by utilizing multiple input interfaces. The data rate at which packet loss is first observed may be used to establish the boundary of the operating region, which can then be used in future SAS design to assess whether certain output interfaces are heavily loaded or not.

### 2.8.3 Application to Networked Cyber-Physical Systems

As digital computing and communication technologies become increasingly integrated into physical systems, it is of urgent necessity to design network infrastructure supporting reliable, low-latency communication among sensors and actuators distributed in physical environment. Recently, switched Ethernet has become increasingly popular in networked cyber-physical systems (NCPS) such as power substation automation systems [29] and avionic systems [39, 120]. In an Ethernet-based NCPS, network-connected devices realize a wide range of tasks by exchanging miscellaneous and sometimes mixed-criticality information (e.g., sensor readings, control commands, and synchronization messages) with each other. To ensure reliable control and operation, network-induced delays for time-critical and/or delay-sensitive applications of an Ethernet-based NCPS must be carefully examined.

In many Ethernet-based NCPSs, publisher-subscriber messaging model is adopted to enable flexible, real-time communication among sensors, actuators, and controllers. NCPS ap-

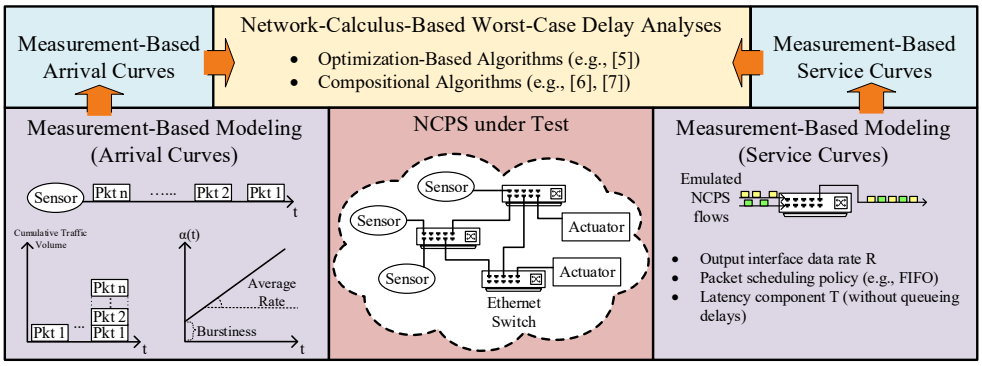



Figure 2.15: Overview of the extended framework.

plications employing the publisher-subscriber model includes power substation automation system based on IEC 61850 [29], Avionics Full-Duplex Switched (AFDX) Ethernet [39, 120], and automotive networks [70]. In a practical Ethernet-based NCPS, one or more communication protocols/standards may be implemented for different applications. When network-based applications are implemented, network traffic pattern on an NCPS (i.e., how each device is connected to other devices, flows generated by each device, and traffic profile of each flow) can also be determined (e.g., based on application and device specifications). To ensure that real-time performance of various time-critical applications are met, it is of vital importance to find the worst-case delay bounds on network-induced delays experienced by their network traffic flows. However, existing worst-case delay analysis algorithms (e.g., [13, 22]) are evaluated using specification-based models, and little has been done to facilitate the evaluation of existing and emerging methods in realistic NCPS settings.

To facilitate worst-case delay analysis of Ethernet-based NCPSs, we improve and extend the framework described in earlier sections. In contrast to tightening delay bounds through designing sophisticated delay bounding algorithms, our framework finds tighter bounds on network-induced delays through the use of realistic, measurement-based models. Therefore, both existing and emerging network-calculus-based delay bounding algorithms can be integrated into our framework to derive delay bounds for real-world NCPSs, enabling NCPS architects to choose proper method(s) to balance between tightness of bounds and computational complexity.

Given the traffic pattern of an Ethernet-based NCPS, our proposed framework (see Fi-

gure 2.15) first extracts traffic characteristics of network-connected devices and establishes arrival curve models. Then, emulated traffic flows are injected into networking devices of the NCPS (e.g., Ethernet switches) and network-calculus-based device models (i.e., service curves) are constructed. Finally, network-calculus-based analytical method(s) will be chosen and measurement-based models will be integrated to find worst-case delay bounds.

**Measurement-Based Modeling**

To perform network-calculus-based analysis, arrival curves of network-connected devices and service curves of networking devices need to be established. To establish the arrival curve of a device (e.g., a leaky-bucket curve $\alpha(t)=\sigma+\rho \cdot t$, where $\sigma$ is the burstiness component and $\rho$ is the average rate), two different ways of taking measurements are supported by our framework:

1. *Direct Measurements.* Using Ethernet taps or taking measurements from test equipment with the same configuration as the device of interest, we obtain network packets generated by the device and timestamps at the traffic capture device. By applying the definition of arrival curve, burstiness and average rate components can be straightforwardly derived (see Figure 2.15).

2. *Specification-Based Emulation.* When actual measurements cannot be easily taken (e.g., in early stage of NCPS design where devices and equipment have not been procured, or a physical device is in operation and cannot be interrupted), we can emulate the device based on its control system specifications (e.g., sampling rates of sensors and communication protocol utilized). Then, direct measurements are taken from the emulated device and arrival curve model is established.

To obtain service curve of a networking device (e.g., a rate-latency curve $\beta(t)=\max\{0, R(t-T)\}$, where $R$ is the rate component and $T$ is the latency component), rate component can be determined from device specifications. However, the latency component must be measured by injecting packets at low rates such that queueing does not occur in the device. This is because $T$ models non-queueing delays incurred by the device. Packet sizes of different traffic sources can be taken into account if differences in packet sizes are significant (e.g., an

NCPS simultaneously transmitting packets as short as 64 bytes and as long as 1500 bytes). This can be done by incorporating a packetizataion term in the service curve model, e.g., $\beta(t)$=max$\{0, R(t-T)-l_{max}\}$, where $l_{max}$ is the maximum packet size of the flow of interest.

**Network-Calculus-Based Analysis**

In practical NCPS design, real-time requirements for all the time-critical tasks need to be verified. Once a specific network-calculus-based delay analysis algorithm (e.g., [13, 22, 135]) is chosen, delay bounds are computed for traffic flows associated with all the time-critical applications. If delay bound for a particular flow exceeds its required worst-case delay, the NCPS design should be further refined. It should be noted that early work on network-calculus-based analysis finds that certain algorithms can generate formally provable bounds that are overly pessimistic. Therefore, extra network measurements based on application-specific traffic pattern can be taken (e.g., at Ethernet switch output interfaces [135]) and models refined with measurements can be adopted to replace intermediate network-calculus models and produce realistic worst-case delay bounds.

**Evaluation**

We emulate an NCPS with a publisher-subscriber traffic pattern shown in Figure 2.16, which has been seen in various application domains (e.g., [29, 39, 70]). Both Ethernet switches are equipped with 100-Mbps interfaces. Three publishers (i.e., Pub. 1, Pub. 2, and Pub. 3) generate time-critical traffic flows and two subscribers consume the data (e.g., the subscriber on the left subscribes to flows from Pub.1 1 and Pub. 2). We use netFPGA NICs to perform traffic capture and packet time-stamping. Network-calculus-based analytical method used in

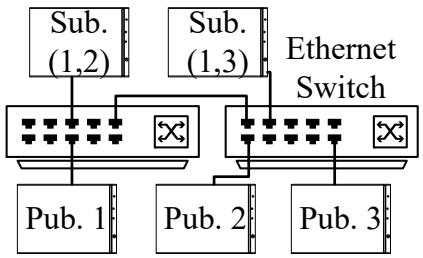
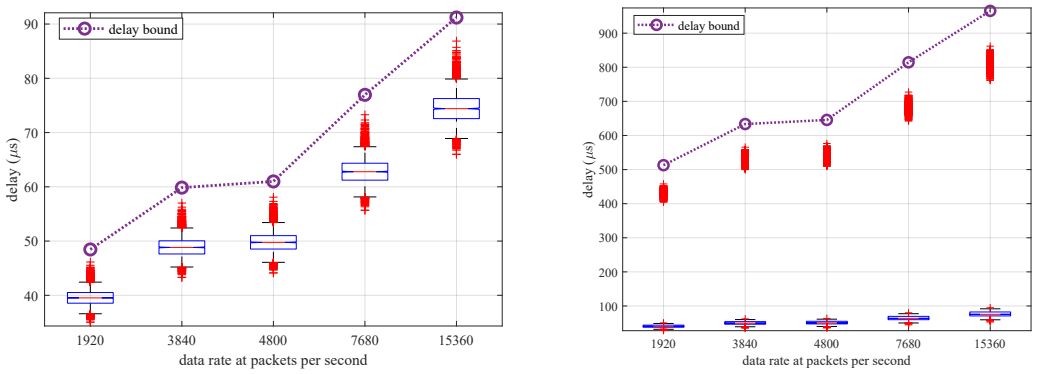



Figure 2.16: Network topology and traffic pattern of the NCPS in our experiments.




(a) Experiment I (b) Experiment II

Figure 2.17: Evaluation results obtained from emulated NCPS depicted in Fig. 2.16.

our experiments is the same as [135].

Currently, two experiments are conducted to evaluate the quality of delay bounds obtained by our framework. In Experiment I (see Figure 2.17a, all network-connected devices generate packets with the same size (192 bytes). Our results show that the proposed approach can generate sufficiently tight bounds, which is only slightly greater than the worst-case delays observed. In Experiment II (see Figure 2.17b), packet size for each network-connected devices is variable (between 64 and 1500 bytes) and packetization effect is taken into account. Although the derived bounds are acceptably tight, it is evident that these bounds deteriorate due to intrinsic pessimism that comes with the packetized models.

### 2.8.4 Application to Avionics Full-Duplex Switched (AFDX) Ethernet

As the de-facto standard for the transmission of network traffic flows of critical avionics applications, Avionics Full-Duplex Switched (AFDX) Ethernet interconnects end systems (e.g., sensor and actuator nodes) with Ethernet switch(es) based on First-in First-out (FIFO) scheduling [121]. For such a networked cyber-physical system (NCPS), it is of vital importance to guarantee that stringent real-time requirements of critical avionics applications are met. Therefore, an analytical tool facilitating the evaluation of worst-case delay performance (i.e., the worst-case end-to-end communication delays experienced by all time-critical flows) of practical AFDX networks is urgently needed.

As a theoretical tool for worst-case delay analysis, network calculus has been applied to

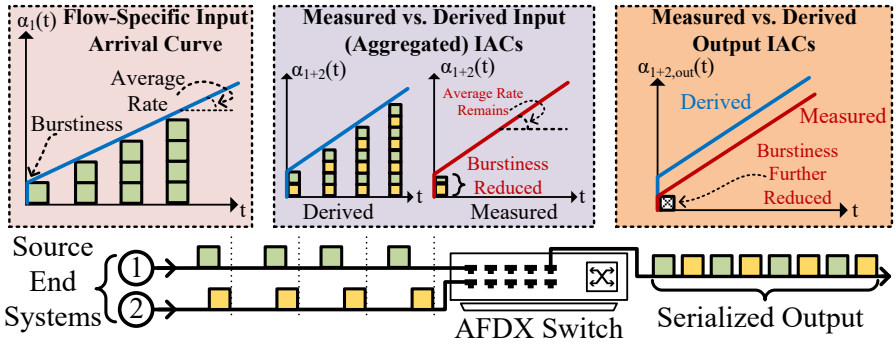
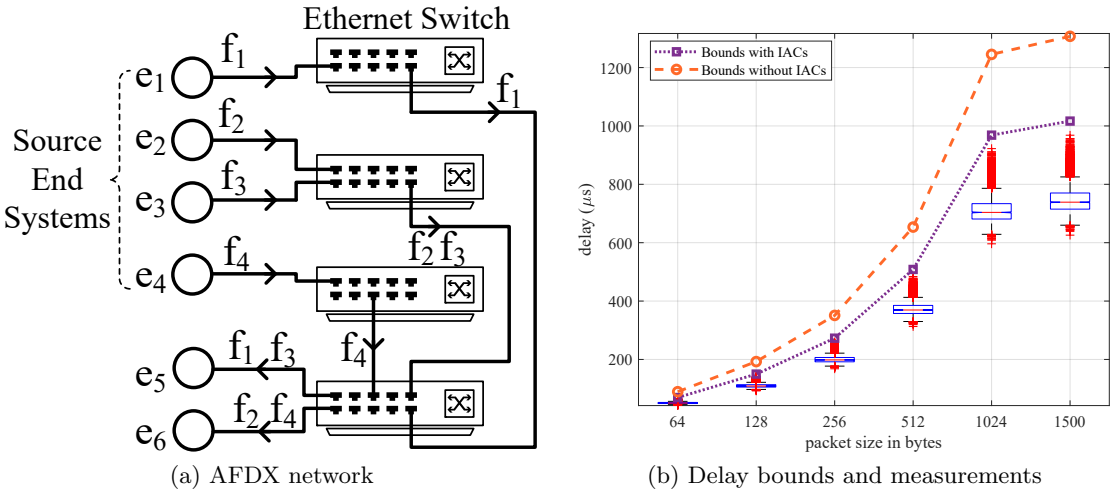



Figure 2.18: Measurement-based vs. derived models.

FIFO networks to derive formally provable delay bounds [23]. However, specification-based network-calculus models (e.g., constructed based on sampling rates and data formats of source end systems) are typically adopted during analysis, and little has been done to validate existing analytical methods in real-world AFDX networks. In fact, it is shown in [135] that delay bounds obtained from network-calculus-based analysis can be further tightened through taking network measurements. To evaluate whether network-calculus-based analysis can provide accurate delay bounds for practical AFDX networks, we combine network measurements with network-calculus-based analysis in this section and examine the quality of delay bounds obtained under application-specific traffic patterns of AFDX systems. By modeling traffic sources and networking devices (e.g., Ethernet switches) through measurements, we establish accurate, realistic traffic and device models for network-calculus-based analysis. Using our approach, different network-calculus-based analytical methods can be leveraged, allowing AFDX architects to construct a proper delay analysis framework to suit application-specific needs (e.g., balancing between computational complexity and tightness of bounds).

In existing analytical methods, intermediate arrival curves (IACs) for traffic flows seen at Ethernet switches must be properly characterized through measurements (e.g., using netFPGA and Ethernet tap). As shown in Figure 2.18, by taking into account serialization effects at any output interface of the switch, burstiness component of the output flow can be reduced. In addition, statistical multiplexing among source end systems may also result in reduced burstiness (at the input end). Therefore, measurement-based models should be properly integrated into existing analytical methods.




(a) AFDX network

(b) Delay bounds and measurements

Figure 2.19: Experiment settings and preliminary results.

In this section, we use the optimization-based analytical approach proposed in [23] and the measurement method outlined in [135]. Our preliminary results show that analytical bounds can be further improved for practical AFDX networks by plugging in measurement-based models and replacing intermediate arrival curves (IACs) with those constructed from measurements.

We emulate an AFDX network shown in Figure 2.19a. We use netFPGA NICs to perform traffic capture and timestamp all the packets. Derived delay bounds are compared with observed maximum delays to help assess their quality (i.e., tightness). In our experiments, all source end systems generate packets with the same size, ranging from 64 bytes to 1500 bytes. Interfaces of all switches operate at 100 Mbps. We choose flow $f_2$ as our flow of interest. Note that in practical AFDX design, worst-case end-to-end delays for all time-critical flows should be evaluated.

As shown in Figure 2.19b, analytical bounds given by the approach in [23] with and without intermediate arrival curves (IACs) incorporated are indeed upper bounds of the end-to-end network-induced delays observed from measurements illustrated in box plot. Furthermore, integrating IACs does improve the quality (i.e., tightness) of application-specific delay bounds for AFDX systems, resulting in bounds that are only slightly greater than the worst-case delays observed. In fact, as shown in Figure 2.19a, flow $f_2$ travels through two switches to reach end system $e_6$. Intermediate arrival curves are measured before and after $f_2$ passes through the first

and the second switches on its path. Through using the measured arrival curve at the input interface of an Ethernet switch, intrinsic pessimism of network-calculus traffic aggregation (i.e., the arrival curve of a flow aggregated from multiple sub-flows is simply the sum of those of the sub-flows) is effectively removed, and characteristics of aggregated traffic flows are accurately taken into account. At the output end of an Ethernet switch, serialization effect is modeled through the use of measured output arrival curve, which has a reduced burstiness component and will thus lead to a tighter input flow (or sub-flow) model for downstream Ethernet switch(es).

## 2.9 Conclusion

In this work, we show that Ethernet switches can be modeled by multiple work-conserving links provided that they have sufficient processing capacity to support dedicated interconnections for all their input/output interface pairs. To analyze a process bus network, we propose the combination of network calculus and measurements, facilitating the construction of realistic service curve models and producing accurate delay bounds that can be validated against measurements. Though experiments on emulated test bed, we also show that our proposed approach can be applied to Ethernet-based NCPSs with hard-real-time performance requirements.

# Chapter 3

# Delay Performance Analysis of Wireless PRP Infrastructure for Networked Cyber-Physical Systems

## 3.1 Background

Due to their low cost and ease of deployment as well as maintenance, wireless communication technologies, such as Wireless Local Area Networks (WLANs), have become an attractive communication solution for industrial control systems (ICSs) [99,125]. However, many wireless communication protocols are not initially proposed for ICSs requiring reliable and low-latency communication. For instance, the IEEE 802.11 family of WLANs are primarily designed to offer high throughput and continuous connectivity. Reliability and latency issues caused by radio interference have become a major deterrent to the application of WLANs in ICSs with latency-critical tasks [79]. To address these concerns, the Parallel Redundancy Protocol (PRP) [53] is leveraged to construct dependable wireless network infrastructure for real-time and safety ICS applications. As an example, parallel redundant WLANs [105, 106] are proposed to compensate the effects of stochastic channel fading by parallel operation of diverse wireless channels, resulting in a fault-tolerant wireless network infrastructure for ICSs with latency-critical and safety applications. Fig. 3.1 presents a simplified view of a parallel re-

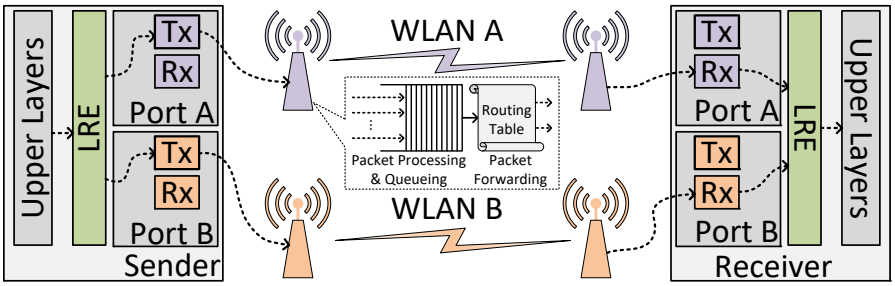



Figure 3.1: A simplified view of a parallel redundant WLAN.

dundant WLAN with unidirectional network traffic from a sender to a receiver. When the sender generates a data frame, its link redundancy entity (LRE) layer duplicates the frame and forwards it via both WLANs A and B. At the receiver side, the LRE layer removes the duplicate that arrives late, forwarding only a single copy to the upper layers. Using PRP as splitter and selection combiner, parallel redundant WLANs offer an improved wireless communication infrastructure with high stochastic reliability. Although wireless PRP networks such as parallel redundant WLANs provide a viable and reliable wireless communication solution for networked control systems (NCSs), application-specific delay performance evaluation is still required (e.g., [48, 93]) because different ICS tasks and applications have diverse delay performance requirements (e.g., constraints on worst-case network-induced delays for various power substation automation tasks are specified in [55]).

Application-specific delay performance of wireless PRP infrastructure is heretofore studied through discrete-event simulations (e.g., [48,104]) and/or measurements (e.g., [24,93]). These approaches enable ICS architects to evaluate the feasibility and usability of wireless PRP infrastructure for specific ICSs of interest during the planning phase of ICS design, but new simulation/experiment must be conducted when important modifications to the ICS design (e.g., changing the sampling frequencies of sensors) are introduced or the same set of equipment are employed in new ICS designs. To reuse measurement and simulation results and reduce the cost of delay performance evaluation, it is of vital importance to develop a theoretical framework that allows ICS architects to quickly assess whether certain latency-critical applications can be run on wireless PRP networks.

In this chapter, we propose a network-calculus-based framework to analyze worst-case

network-induced delays of wireless network infrastructure based on PRP, facilitating its adoption in miscellaneous ICSs with diverse delay performance requirements. Contributions of this work are as follows:

- *Modeling Wireless PRP Networks with Network Calculus.* Leveraging the deterministic network calculus (DNC) theory, we model wireless PRP networks to capture the impacts of various components of network-induced delays, including processing delays, transmission delays, and queueing delays. The proposed models enable ICS architects/designers to reuse application-specific delay measurements or simulation results and estimate worst-case delay performance of wireless PRP networks in new ICS designs.

- *Delay Bounding on Non-Feedforward Networks.* Practical wireless PRP networks support bidirectional communication between senders and receivers (see Sec. 3.3.5), which results in non-feedforward traffic patterns that cannot be handled by state-of-the-art delay bounding techniques designed for feedforward traffic pattern (i.e., all traffic flows on a network can be represented by increasing sequences after unique integer identifiers are assigned to all network nodes [20]). In this work, we address this challenge by introducing stopped sequences into our worst-case delay analysis framework and make our proposed approach applicable to realistic application scenarios of wireless PRP networks.

- *Derivation of Closed-Form Worst-Case Delay Expressions.* We demonstrate that our proposed approach is able to derive closed-form expressions of worst-case network-induced delays under general, non-feedforward network traffic patterns. The closed-form expressions can be re-used in different phases of ICS design as long as the given network traffic pattern remains the same. The proposed framework is thus a valuable tool for ICS architects aiming to design wireless PRP network infrastructure for latency-critical ICSs.

## 3.2 Related Work

### 3.2.1 Wireless PRP Network Infrastructure

Proposed and evaluated in [105, 106], parallel redundant WLANs have been shown to offer satisfactory reliability for real-time and safety applications in industrial control systems. In [24, 25], seamless redundancy principles of PRP are directly applied to WLAN links and additional improvements such as duplication avoidance mechanisms are designed in the proposed Wi-Red protocol. To ensure that communication delays induced by parallel redundant WLANs can be tolerated by critical ICS tasks, discrete-event simulations and measurement-based experiments have been conducted for various industrial control applications, such as power system state estimation using phasor measurement data [93] and wireless networked control systems for industrial workcell [48]. However, network-induced delays obtained through simulation or measurement are typically not the worst-case results because the testing scenarios are constructed based on representative communication workloads but do not exhaustively cover boundary cases that can lead to significant increase in network-induced delays. Furthermore, when modifications to an ICS design result in traffic pattern changes or a new ICS is to be designed, previously obtained results cannot be re-used and new experiments/simulations must be conducted. This problem can be resolved by the development of a theoretical framework for worst-case delay analysis of wireless PRP networks, which is the major contribution of this work.

In addition to WLANs, PRP has also been combined with other wireless communication technologies. Leveraging 4G Long Term Evolution (LTE) cellular protocol, a wireless seamless redundant communication infrastructure is proposed for railway communication systems in [7]. To facilitate flexible factory automation and control, seamless redundancy principles of PRP are applied to IEEE 802.15.4 wireless networks in [67]. Though the combination of wireless communication technologies and PRP leads to flexible and reliable wireless communication infrastructure for ICSs, these solutions need to be carefully evaluated for different latency-critical tasks before adoption/deployment. The worst-case delay analysis framework proposed in this chapter can be applied to different wireless PRP networks, ranging from wi-

reless redundant WLANs to those based on 4G LTE or IEEE 802.15.4, reducing the cost of delay performance evaluation for ICS design.

### 3.2.2 Worst-Case Delay Performance Analysis Using Network Calculus

In contrast to queueing theory, network calculus [17, 27, 33] studies worst-case performance of queueing networks. To apply network-calculus theorems and derive delay bounds for networks with different topologies, many algorithms have been proposed and implemented. In [115], three different algorithmic approaches to applying network-calculus theorems, namely total-flow analysis (TFA), separated-flow analysis (SFA), and pay-multiplexing-only-once (PMOO) analysis, are developed for wireless sensor networks (WSNs). It has later been shown in [113] that these algorithms may not always generate tight delay bounds. To derive tighter delay bounds by combining the TFA, SFA, and PMOO approaches, a new delay bounding method [14] has been proposed. However, computational cost of this approach is prohibitively expensive. Alternatively, the problem of worst-case delay analysis can be formulated as optimization problems under network-calculus constraints [20, 113]. However, the computational costs of the optimization-based approaches may be prohibitively high [20] if exact (i.e., tight) delay bounds are desired.

Existing algorithms of network-calculus-based worst-case delay analysis focus on tightening the delay bounds for feedforward networks. A network is said to be feedforward if all its traffic flows can be represented by increasing sequences after unique integer identifiers are assigned to all the network nodes. If no feasible assignment scheme can be found (i.e., at least one of the traffic flows cannot be represented by an increasing sequence under any assignment scheme), the network is regarded as non-feedforward. Traffic patterns resulted from bidirectional communication on wireless PRP networks are non-feedforward, making existing algorithms inapplicable. In this work, we propose to apply network calculus theorems to non-feedforward traffic patterns, making it possible to derive delay bounds for practical wireless PRP networks.

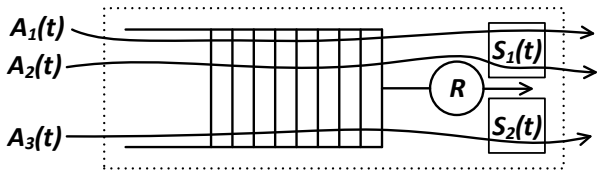



Figure 3.2: Network-calculus model of a single wireless access point on a wireless PRP network.

## 3.3 Modeling Wireless PRP Networks with Network Calculus

The key to applying network calculus to wireless PRP networks is to properly model all the traffic forwarding devices (e.g., wireless access points, 4G LTE base stations, or IEEE 802.15.4 sensor nodes). Throughout the remainder of this chapter, we present our proposed framework using wireless redundant WLANs as an example. We note that the same modeling approach can be applied to other wireless PRP network infrastructure (see Sec. **??**) as well.

A wireless access point (AP) in parallel redundant WLANs (e.g., see Fig. 3.1) can be modeled by a constant-capacity work-conserving link with ideal routers attached to its output end. At each wireless access point, a packet must go through certain processing tasks (e.g., validating its CRC bytes). If the communication channel is temporarily not available for transmission, the packet needs to be queued. When the communication channel becomes available, the packet is forwarded to the destination according to one of the routing table entries. In our proposed framework, packet processing and queueing at an access point is modeled by a work-conserving link, whereas the routing table entry designating the forwarding of packets in a traffic flow toward a certain destination node is modeled by an ideal router. Since a traffic flow originating from a source node may be forwarded to multiple destinations, multiple ideal routers are attached to the work-conserving link to represent all relevant routing table entries.

As an example, let us consider the AP model in Fig. 3.2. The work-conserving link has capacity $R$ and receives three individual input flows $A_1(t)$, $A_2(t)$, and $A_3(t)$. The control functions of the two ideal routers are $S_1(t)$ and $S_2(t)$, respectively. Each control function can be thought as a routing table entry, which specifies whether a packet can pass through the router or not. The ensuing subsections introduce the important theorems and concepts from network calculus that are used in our proposed analytical framework. Formal proofs of the

theorems can be found in classic textbooks on network calculus, such as [17, 27].

### 3.3.1 Definitions

In network calculus, arrival and departure processes (i.e., traffic flows) are modeled by non-negative cumulative functions. A cumulative function $A(t)$ is a non-negative and non-decreasing function defined for $t \geq 0$ with $A(0)=0$. If an arrival process is modeled by $A(t)$, then $A(t_1)$ represents the cumulative traffic volume seen from this process starting from $t=0$ up until $t=t_1$.

In addition, arrival curves are used in network calculus to characterize traffic flows. For a traffic flow modeled by cumulative function $A(t)$, a function $\alpha(t)$ is the arrival curve for $A(t)$ if and only if

$$A(t) - A(s) \leq \alpha(t-s), \text{ for } \forall t \geq s \geq 0.$$

A widely-used type of arrival curves is the leaky-bucket arrival curve, which is of the form $\alpha(t)=\rho t+\sigma$. If a traffic flow $A(t)$ has an arrival curve $\alpha(t)=\rho t+\sigma$, then it is said to be $(\sigma, \rho)$-upper constrained, which is denoted by $A \sim (\sigma, \rho)$. Parameter $\rho$ represents the long-term average rate of $A(t)$, whereas $\sigma$ represents the (maximum) instantaneous burstiness of $A(t)$.

In network calculus, networking devices such as wireless access points are modeled by network elements offering certain service curves. Suppose that a network element has a single arrival process $A(t)$ and a single departure process $D(t)$. The network element is said to offer a service curve $\beta(t)$ if and only if

$$D(t) \geq \inf_{0 \leq s \leq t} \{A(s) + \beta(t-s)\} = (A \otimes \beta)(t), \text{ for } \forall t \geq 0.$$

The operator $\otimes$ denotes the min-plus convolution operation. If a work-conserving link has constant capacity $R$, then we say that its service curve is $\beta(t)=R \cdot t$.

An input flow may be routed (i.e., demultiplexed) to multiple output paths. An ideal router is a network element with traffic input $A(t)$, output $D(t)$, and control input $F(t)$. The input-output relationship between the input flow to an ideal router and its output flow governed by $F(t)$ is expressed as $D(t)=F[A(t)]$. Note that an input flow can pass through

multiple ideal routers when its packets have different destinations.

### 3.3.2 Modeling Traffic Forwarding Devices

To facilitate back-of-the-envelop calculation, we model a wireless access point in parallel redundant WLANs using a constant-capacity work-conserving link with ideal routers attached to its output end. This subsection introduces two key operations used in the construction of the model illustrated in Fig. 3.2 for a wireless access point.

#### Traffic Aggregation

Suppose that multiple input flows are fed into a single wireless access point (e.g., multiple senders running different control tasks instead of single sender shown in Fig. 3.1 can be connected to the same wireless access point). If $A_1 \sim (\sigma_1, \rho_1)$ and $A_2 \sim (\sigma_2, \rho_2)$, then the aggregation of $A_1(t)$ and $A_2(t)$, namely $A_1(t) + A_2(t)$, is said to be $(\sigma_1 + \sigma_2, \rho_1 + \rho_2)$-upper constrained. In general, we have the following lemma for traffic aggregation (also known as traffic flow multiplexing).

**Lemma 3.3.1.** *Suppose that a network element has $n$ input flows $A_i(t), i=1, 2, \ldots, n$. If each individual flow $A_i$ is $(\sigma_i, \rho_i)$-upper constrained, then the aggregation of the $n$ input flows is represented by $A(t) = \Sigma_{i=1}^{n} A_i(t)$ and is $(\Sigma_{i=1}^{n}\sigma_i, \Sigma_{i=1}^{n}\rho_i)$-upper constrained.*

When two traffic flows merge, their data bits will be serialized according to a certain multiplexing policy. Note that this serialization effect is not modeled by Lemma 3.3.1. Combining traffic flows sharing the same ideal router or destined for the same next-hop wireless access point using this lemma simplifies the derivation of worst-case delay bounds for non-feedforward networks and makes back-of-the-envelop calculation possible.

#### Left-Over Service Curve

The notion of left-over service curve has been extensively used in delay-bounding algorithms for feedforward networks such as those proposed in [14,113,115]. In our analytical framework, we also leverage the following theorem for left-over service curve.

**Theorem 3.3.2.** *Suppose that a work-conserving link has constant capacity $R$ and two inputs $A_1(t)$ and $A_2(t)$. If $A_1 \sim (\sigma_1, \rho_1)$, then the service curve offered by this work-conserving link to $A_2(t)$ can be denoted by $\beta_2(t)$, where $\beta_2(t) = (Rt - \rho_1 t - \sigma_1)^+$, and $(\cdot)^+$ is defined as* $\max\{\cdot, 0\}$.

As illustrated in Fig. 3.2, packet-processing elements of a wireless access point are modeled using a single work-conserving link with constant capacity $R$ and infinite buffers (so that no packets are dropped due to buffer overflow). We note that this single-link model is conservative because parallelisms inside the wireless access point are ignored. Instead, we assume that the wireless access point processes packets one by one when multiple packets arrive simultaneously or when its buffer is not empty. Theorem 3.3.2 allows us to find out the service curve solely used by an individual flow (though this flow itself may be an aggregation of multiple flows). The service curve obtained from this theorem is known as left-over service curve [113].

### 3.3.3 Output Burstiness and Worst-Case Delay

**Theorem 3.3.3.** *Suppose that a network element offers service curve $\beta(t)$ for input flow $A(t)$. The output flow corresponding to $A(t)$ is denoted as $D(t)$. If $A(t)$ has an arrival curve $\alpha(t)$, then $D(t)$ has an arrival curve $\alpha'(t)$, where*

$$\alpha'(t) = \sup_{t,s \geq 0} \{\alpha^*(t+s) - \beta(s)\} = (\alpha^* \oslash \beta)(t).$$

Note that $\alpha^*(t)$ is the sub-additive closure of $\alpha(t)$ and the $\oslash$ operator denotes the deconvolution operation. Definition of sub-additive closure can be found in [17,27]. For a leaky-bucket arrival curve $\alpha(t) = \rho t + \sigma$, we have $\alpha^*(t) = \alpha(t)$.

In addition, we have the following theorem for worst-case delay experienced by a single flow passing through a networking element that offers service curve $\beta(t)$.

**Theorem 3.3.4.** *Suppose that a network element offers service curve $\beta(t)$ to its input flow $A(t)$, which has an arrival curve $\alpha(t)$. Let $d(t)$ denote that maximum delay experienced by flow $A(t)$ up to time $t$, the worst-case delay bound can be found as follows*

$$\forall t \geq 0 : \ d(t) \leq \inf\{t \geq 0 : \ (\alpha^* \oslash \beta)(-t) \leq 0\}.$$

### 3.3.4 Ideal Routing

The behavior that a wireless access point routes packets from its input flow to different destinations can be modeled using ideal routers. To model bidirectional communications on parallel redundant WLANs, we need to consider multiple traffic flows passing through a wireless access point and arriving at different destinations. As shown in Fig. 3.2, we attach two ideal routers to the output end of the work-conserving link. Each router routes part of the flow passing through it toward a wireless access point. The following lemma can be applied when ideal routers are introduced.

**Lemma 3.3.5.** *Suppose that an ideal router has a single input $A(t)$, which is $(\sigma, \rho)$-upper constrained. If the routing control function $F(t)$ is $(\delta, \gamma)$-upper constrained, then its output flow $D(t)$ is $(\gamma\sigma+\delta, \gamma\rho)$-upper constrained.*

We only show the ideal routers relaying packets towards other wireless access points in Fig. 3.2 because our flows of interest are the aggregated flows passing through two wireless access points to travel from a sender to a receiver (see Fig. 3.1).

### 3.3.5 Non-Feedforward Traffic Patterns and Stopped Sequences

In practical applications of wireless PRP networks, bidirectional communications between a sender and a receiver typically result in a non-feedforward traffic pattern. For any given network, we can assign unique integer identifiers to all its network nodes and represent all the traffic flows as integer sequences. If no feasible assignment scheme can be found such that all the flows can be represented as monotonically increasing sequences, the network has a non-feedforward traffic pattern.

#### Modeling a Single Wireless Path of Wireless PRP Networks

Fig. 3.3 illustrates the model for a wireless path of a parallel redundant WLAN with a non-feedforward traffic pattern. In particular, $C_1(t)$ and $D_1(t)$ models the aggregated traffic flows passing through individual wireless access points (APs) and getting forwarded toward destinations other than the two APs connecting the sender and the receiver (e.g., the sender may send

data frames to APs other than the ones connecting the sender and the receiver, and so does the receiver). On the path of $A_1(t)$, two ideal routers are inserted so that $A_5(t)$ represents aggregated traffic flow consumed by the receiver connected to wireless AP2. Similarly, two ideal routers are inserted so that $B_5(t)$ models the traffic flow consumed by the receiver connected to the wireless AP1 (since we consider practical scenarios with bidirectional communications). Note that the sender connected to AP1 is one of the sources contributing to the aggregated input flow $A_1(t)$. Similarly, the sender connected to AP2 is among the sources that generate $B_1(t)$. Evidently, the non-feedforward traffic pattern is caused by the bidirectional communications between the sender and the receiver: To determine the left-over service curve offered by AP1 for $A_1$, we have to determine the arrival curve for $B_3(t)$, which in turn requires the determination of the left-over service curve offered by AP2 for $B_1(t)$. However, to determine this left-over service curve for $B_1(t)$, we need to obtain the arrival curve of $A_3(t)$, which in turn requires the determination of the left-over service curve offered by AP1 for $A_1(t)$. Such intricacies will not be encountered if the traffic pattern is feedforward (i.e., all traffic flows on the network can be represented by a set of increasing sequences by properly assigning integer identifiers to all the networking nodes [20]), but it is necessary to derive worst-case delay bounds on non-feedforward networks because practical parallel redundant WLANs are generally non-feedforward.

**Stopped Sequences**

To analyze such a non-feedforward traffic pattern, we use the technique of introducing stopped sequences. For a cumulative function $A(t)$, cumulative function $A^\tau(t)$ is the stopped sequence of $A(t)$ at time $\tau \geq 0$ and is defined as

$$A^\tau(t) = \begin{cases} A(t), & \text{if } t \leq \tau, \\ A(\tau), & \text{otherwise.} \end{cases}$$

The following two lemmas enable us to analyze non-feedforward traffic patterns and find out whether worst-case end-to-end delay bounds exist.

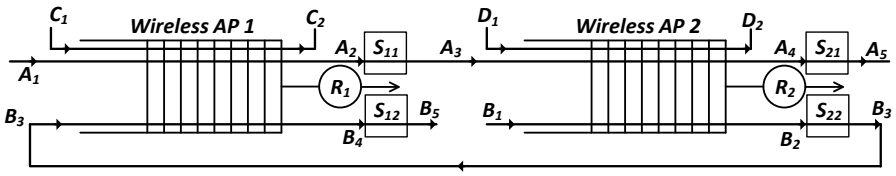



Figure 3.3: Modeling of a wireless path of a wireless PRP network with a non-feedforward traffic pattern imposed by bidirectional communications between sender and receiver attached to different wireless access points (APs).

**Lemma 3.3.6.** *For every chosen $\rho$, a stopped sequence $A^{\tau}(t)$ is $(\sigma(\tau), \rho)$-upper constrained, where*

$$\sigma(t) = \sup_{0 \le t \le \tau} \sup_{0 \le s \le t} \{A(t) - A(s) - \rho(t-s)\}.$$

**Lemma 3.3.7.** *If the stopped sequence $A^{\tau}(t)$ in Lemma 3.3.6 is also found to be $(\sigma, \rho)$-upper constrained, then we have $\sigma(\tau) \le \sigma$.*

## 3.4 Worst-Case Delay Analysis for Wireless PRP Network Infrastructure

With the concepts, theorems, and lemmas introduced in Section 3.3, we now fully parameterize the model depicted in Fig. 3.3 and show the existence of upper bounds on the end-to-end delays induced by this wireless path. In particular, we derive the worst-case delay experienced by packets that are sent to the receiver connected to AP2 by the sender connected to AP1. Leveraging symmetry of the non-feedforward traffic pattern, worst-case delays experienced by packets sent by the sender at AP2 to the receiver at AP1 can be derived similarly. Note that for a practical parallel redundant WLAN, two wireless paths need to be modeled and analyzed. The parameters for the models will have different values, which can be obtained from prior measurements or simulations.

Suppose that all input flows have leaky-bucket arrival curves. Without loss of generality, suppose that we have $A_1 \sim (\sigma_{A1}, \rho_{A1})$, $B_1 \sim (\sigma_{B1}, \rho_{B1})$, $C_1 \sim (\sigma_{C1}, \rho_{C1})$, and $D_1 \sim (\sigma_{D1}, \rho_{D1})$ for the input flows. We also assume that both APs have sufficient processing capacity, i.e., $R_1 > \rho_{A1} + \rho_{B1} + \rho_{C1}$ and $R_2 > \rho_{A1} + \rho_{B1} + \rho_{D1}$. To model bidirectional communications, the output end of each work-conserving link is attached with two ideal routers. $S_{11}$, $S_{12}$, $S_{21}$, and $S_{22}$ are

the control functions of the ideal routers, respectively. Suppose that we have $S_{11}\sim(\delta_{11},\gamma_{11})$, $S_{12}\sim(\delta_{12},\gamma_{12})$, $S_{21}\sim(\delta_{21},\gamma_{21})$, and $S_{22}\sim(\delta_{22},\gamma_{22})$. We note that values of the parameters introduced here can be obtained either from prior knowledge of traffic profiles of different ICS applications or from measurements and simulations.

Since this traffic pattern is non-feedforward, we introduce stopped sequences for all the traffic flows. For instance, $A_2^\tau$ and $B_2^\tau$ are the stopped sequences for flows $A_2$ and $B_2$, respectively. Applying Lemma 3.3.6, we can have $A_2^\tau\sim(\sigma_{A2}(\tau),\rho_{A1})$ and $B_2^\tau\sim(\sigma_{B2}(\tau),\rho_{B1})$. Applying Lemma 3.3.5 at ideal routers $S_{11}$ and $S_{22}$, we have $A_3^\tau\sim(\gamma_{11}\sigma_{A2}(\tau)+\delta_{11},\gamma_{11}\rho_{A1})$ and $B_3^\tau\sim(\gamma_{22}\sigma_{B2}(\tau)+\delta_{22},\gamma_{22}\rho_{B1})$. At AP1, we apply Theorem 3.3.2 and Lemma 3.3.1 to obtain the left-over service curve for $A_1$, which is denoted by $\beta_{A1}^\tau(t)$ and

$$\beta_{A1}^\tau(t)=(R_1t-(\rho_{C1}t+\sigma_{C1}+\gamma_{22}\sigma_{B2}(\tau)+\delta_{22}+\gamma_{22}\rho_{B1}t))^+.$$

Similarly, we also have

$$\beta_{B1}^\tau(t)=(R_2t-(\rho_{D1}t+\sigma_{D1}+\gamma_{11}\sigma_{A2}(\tau)+\delta_{11}+\gamma_{11}\rho_{A1}t))^+.$$

Then, by applying Theorem 3.3.3 on $A_1$ and $\beta_{A1}^\tau(t)$, we have $A_2^\tau\sim(\sigma'_{A2}(\tau),\rho_{A1})$, where

$$\sigma'_{A2}(\tau)=\sigma_{A1}+\rho_{A1}\cdot\frac{\sigma_{C1}+\gamma_{22}\sigma_{B2}(\tau)+\delta_{22}}{R_1-\rho_{C1}-\gamma_{22}\rho_{B1}}.$$

Similarly, we also have $B_2^\tau\sim(\sigma'_{B2}(\tau),\rho_{B1})$, where

$$\sigma'_{B2}(\tau)=\sigma_{B1}+\rho_{B1}\cdot\frac{\sigma_{D1}+\gamma_{11}\sigma_{A2}(\tau)+\delta_{11}}{R_2-\rho_{D1}-\gamma_{11}\rho_{A1}}.$$

Using Lemma 3.3.7, we have $\sigma_{A2}(\tau)\leq\sigma'_{A2}(\tau)$ and $\sigma_{B2}(\tau)\leq\sigma'_{B2}(\tau)$. Solving these two inequalities, we can find that $\sigma_{A2}(\tau)\leq\sigma''_{A2}$, where

$$\sigma''_{A2}=\frac{\sigma_{A1}+\gamma_{A1}(\sigma_{C1}+\delta_{22})+\gamma_{A1}\gamma_{22}(\sigma_{B1}+\gamma_{B1}(\sigma_{D1}+\delta_{11}))}{1-\gamma_{11}\gamma_{B1}\gamma_{22}\gamma_{A1}},$$

and $\gamma_{A1}=\frac{\rho_{A1}}{R_1-\rho_{C1}-\gamma_{22}\rho_{B1}}$, $\gamma_{B1}=\frac{\rho_{B1}}{R_2-\rho_{D1}-\gamma_{11}\rho_{A1}}$. Note that $\sigma''_{A2}$ is independent of $\tau$. Similarly,

we can derive an upper bound on $\sigma_{B2}(\tau)$, which is also independent of $\tau$ and can be denoted by $\sigma''_{B2}$.

After applying the technique of introducing stopped sequences and finding burstiness bounds for $A_2(t)$ and $B_2(t)$ that are independent of $\tau$, we are now able to find out the burstiness bound on $A_4(t)$. The left-over service curve offered to $A_3$ by AP2 is denoted by $\beta_{A3}(t)$ and given by

$$\beta_{A3}(t) = (R_2 t - (\sigma_{D1} + \rho_{D1} t + \sigma_{B1} + \rho_{B1} t))^+.$$

Therefore, we have $A_4 \sim (\sigma_{A4}, \rho_{A1})$, where

$$\sigma_{A4} = \gamma_{11}\sigma''_{A2} + \delta_{11} + \gamma_{11}\rho_{11}\frac{\sigma_{D1} + \sigma_{B1}}{R_2 - \rho_{D1} - \rho_{B1}}.$$

It is trivial to find that $A_3(t) \sim (\gamma_{11}\sigma''_{A2} + \delta_{11}, \gamma_{11}\rho_{A1})$. In fact, worst-case buffer requirement (i.e., deterministic upper bound on backlog) at each individual AP can be found using our proposed framework. Furthermore,our approach allows for a fine-grained understanding of the worst-case buffer requirements for individual APs since the impacts of and interactions between different flows (i.e., $A_1(t), B_1(t), C_1(t)$, and $D_1(t)$) can now be quantitatively assessed.

Finally, the worst-case delay $d_{A1}$ experienced by packets from $A_1$ that passes through both APs is upper bounded by the sum of the delay bounds induced by AP1 and AP2, i.e.,

$$d_{A1} = d^{A1}_{left} + d^{A1}_{right}.$$

At AP1, the left-over service for $A_1(t)$, which needs to be independent of $\tau$, can now be derived. Let us denote this left-over service curve by $\beta_{A1}(t)$. By applying Theorem 3.3.2, we obtain

$$\beta_{A1}(t) = (R_1 t - (\rho_{C1} t + \sigma_{C1} + \gamma_{22}\sigma''_{B2} + \delta_{22} + \gamma_{22}\rho_{B1} t))^+.$$

Then, we can apply Theorem 3.3.4 and obtain

$$d^{A1}_{left} = \frac{\sigma_{A1} + \sigma_{C1} + \gamma_{22}\sigma''_{B2} + \delta_{22}}{R_1 - \rho_{C1} - \gamma_{22}\rho_{B1}}.$$

At AP2, we have already obtained the left-over service curve for $A_3(t)$ and its arrival curve. By applying Theorem 3.3.4 again, we have

$$d_{right}^{A1} = \frac{\gamma_{11}\sigma''_{A2} + \delta_{11} + \sigma_{D1} + \sigma_{B1}}{R_2 - \rho_{D1} - \rho_{B1}}.$$

Note that the delay bound $d_{A1}$ is independent of $\tau$, i.e., a closed-form expression of the worst-case delay bound for the wireless path is found. Since the closed-form expressions given by our proposed approach can be formally proved using network calculus theorems (see Sec. 3.3), the derived bounds are reliable as long as the assumptions of our framework are carefully met and values for the parameters of our models are properly selected.

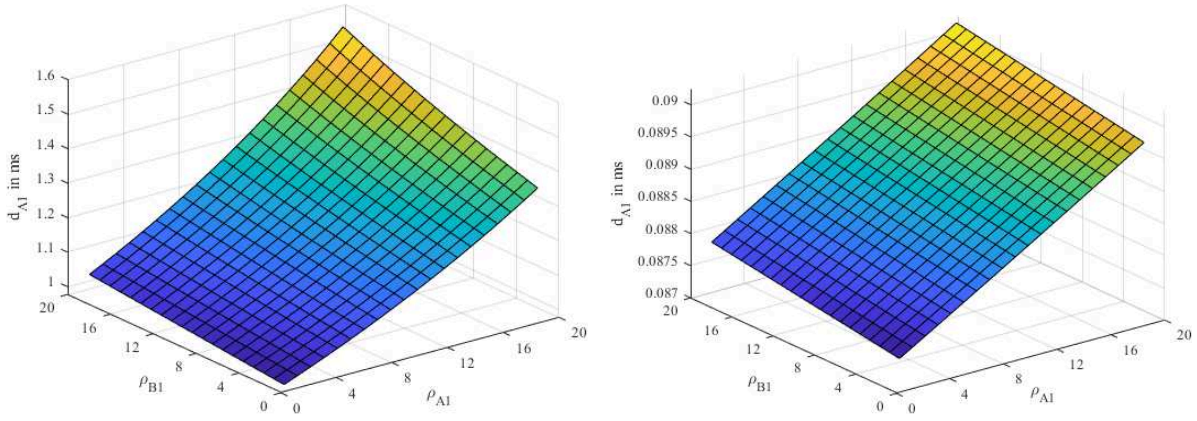
## 3.5 Numerical Evaluation

### 3.5.1 Experiment Settings

To demonstrate the use of our proposed framework, we consider a wireless PRP network designed for streaming phasor measurements. In [93,102], it is reported that the size of phasor measurement packets is 300 bytes, with one packet every 20 ms. The wireless PRP network is depicted in Fig. 3.1 and its network calculus model is shwon Fig. 3.3. Our flow of interest is $A_1(t)$. Suppose that flow $B_1(t)$ is also a stream of phasor measurements with the same packet size and rate. If no other tasks generate any network traffic, i.e., $A_1(t)$, $B_1(t)$ only contain phasor measurements and $C_1(t)$, $D_1(t)$ contain no packets, we can model this application scenario using the following parameter values: $\sigma_{C1}$=$\sigma_{D1}$=0, $\rho_{C1}$=$\rho_{D1}$=0, $\gamma_{11}$=$\gamma_{12}$=$\gamma_{21}$=$\gamma_{22}$=1, $\delta_{11}$=$\delta_{12}$=$\delta_{21}$=$\delta_{22}$=0, $\sigma_{A1}$=$\sigma_{B1}$=2400 bps, and $\rho_{A1}$=$\rho_{B1}$=120000 bps. We assume that IEEE 802.11b is used [93], and the raw data rates (i.e., values of $R_1$ and $R_2$) are 1 Mbps, 5.5 Mbps, or 11 Mbps.

Table 3.1: Worst-Case Delay Bounds at Different Raw Data Rates.

| Raw Data Rate | 1 Mbps | 5.5 Mbps | 11 Mbps |
|---|---|---|---|
| Delay Bound ($A_1t$) | 0.9776 ms | 0.1751 ms | 0.0874 ms |




(a) $R_1$=$R_2$=1 Mbps. (b) $R_1$=$R_2$=11 Mbps.

Figure 3.4: Impacts of the number of phasor measurement units on worst-case network-induced delays.

### 3.5.2 Worst-Case Delays at Different Raw Data Rates

Under the scenario described in Sec. 3.5.1, worst-case delay bounds at different raw data rates are listed in Table 3.1. The delay bounds given by our proposed framework show that the wireless PRP network in Fig. 3.1 is suitable for streaming phasor measurement data, which typically requires worst-case end-to-end delays of 3 ms. The theoretical results agree with measurements obtained in [93]. With delay bounds derived by our framework, ICS architects/designers can choose proper raw data rates under various design constraints (e.g., end-to-end delay requirements, power consumption constraints).

### 3.5.3 Impacts of the Number of Phasor Measurement Units

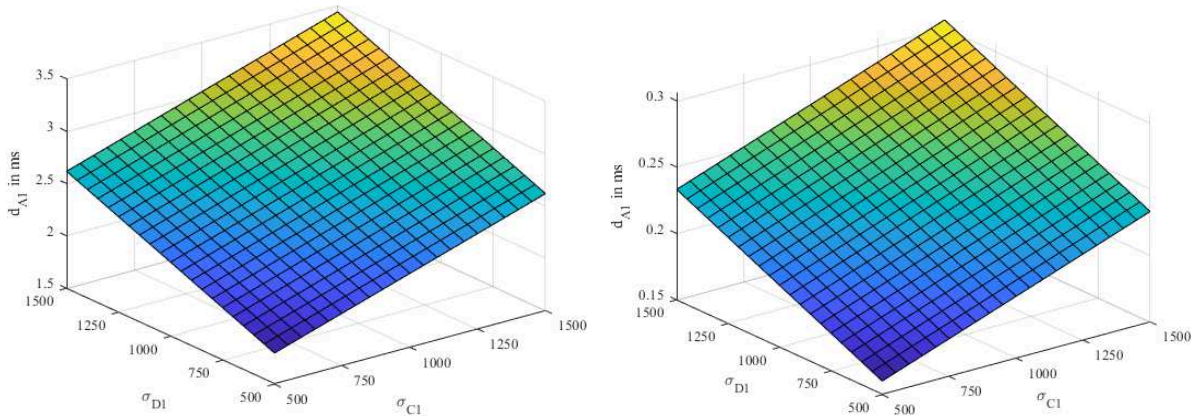
Suppose that we want to deploy more phasor measurement units into the system and that traffic profile of each unit remains the same. Fig. 3.4 shows the worst-case delay bounds when the number of phasor measurement units connected to each wireless access point varies from 1 to 20. As shown in Fig. 3.4b, worst-case latency for phasor measurement packets is as low as 0.0902 ms when each wireless access point has 20 phasor measurement units and the raw data rate is 11 Mbps. On the other hand, Fig. 3.4a shows that the worst-case delay bound is 1.5547 ms when the raw data rate is reduced to 1 Mbps. At this low rate, our framework can help ICS architects find out that the number of phasor measurement units cannot exceed

37 at each access point if the worst-case delay requirement for the phasor data streaming application is 3 ms. When the raw data rate is low, it is therefore important to analyze worst-case delays of wireless PRP network infrastructure for applications with stringent worst-case delay requirements.

### 3.5.4 Impacts of Non-Latency-Critical Tasks

In addition to phasor data, network traffic generated by non-latency-critical tasks may also be carried by the wireless PRP network. Suppose that phasor measurement units at each wireless access point are occasionally reconfigured by the ICS system operators via the wireless PRP network. This task is not latency-critical and occurs infrequently, but its bursty traffic (i.e., large network packets containing configuration data) can impose additional queueing delays for phasor measurement data steams. If two operators reconfigure the phasor measurement units at both wireless access points simultaneously, we can model the generated traffic with $C_1(t)$ and $D_1(t)$. Since this task is infrequent, we approximate the average rates with $\rho_{C1}$=$\rho_{D1}$= 0. The burstiness components, $\sigma_{C1}$ and $\sigma_{D1}$, model the bursty network packets containing configuration data. Suppose that size of these configuration data packets varies from 500 to 1500 bytes and that each wireless access point has only one phasor measurement unit connected to it. Fig. 3.5 shows the worst-case delay bounds derived by our framework with raw data rates of 1 Mbps and 11 Mbps.

At 11 Mbps, the newly introduced configuration task causes the worst-case delay bound for the flow of interest to increase from 0.0874 ms to 0.3061 ms, which is still far below the worst-case delay requirement of 3 ms. However, when the raw data rate is reduced to 1 Mbps, it is shown in Fig. 3.5a that a 1500-byte bursty configuration data packet causes a significant increase in worst-case delay for the flow of interest from 0.9776 ms to 3.4366 ms, which exceeds the worst-case delay requirement. Therefore, traffic regulation mechanism (e.g., constraining the maximum packet size to 1000 bytes) may need to be introduced when the wireless PRP network carries both phasor measurement data and configuration data at raw data rates of 1 Mbps.




(a) $R_1$=$R_2$=1 Mbps. (b) $R_1$=$R_2$=11 Mbps.

Figure 3.5: Impacts of bursty traffic from non-latency-critical tasks on worst-case network-induced delays.

## 3.6 Conclusion and Future Work

In this work, we apply deterministic network calculus to model wireless PRP network infrastructure and show that closed-form expressions of network-induced worst-case end-to-end delays can be found for practical scenarios with non-feedforward traffic patterns, which has not been derived in prior research. Our current work focuses on wireless path consisting of two APs, which are widely employed and studied in different ICS applications (e.g., [7,67,93]). Giving closed-form expressions of delay bounds, our approach facilitates back-of-the-envelope derivation and provides guidance on refining wireless PRP infrastructure designs.

# Chapter 4

# Intrusion and Botnet Detection for Supervisory Control and Data Acquisition (SCADA) Networks

## 4.1 Detecting Peer-to-Peer (P2P) Botnets in SCADA Networks

### 4.1.1 Introduction

Supervisory Control and Data Acquisition (SCADA) systems have been deployed to monitor and control electrical power grids for decades. As conventional power grids around the globe evolve into smart grids, an increasing number of SCADA systems get connected to remote SCADA systems, corporate networks, and even the Internet [133, 140]. To support the ever-growing series of smart-grid applications, the integration of state-of-the-art information technologies, such as cloud computing, has also been proposed [11, 111], which will introduce more computers into SCADA systems. The increasing interconnectivity, deployment of computers, and use of commercial-off-the-shelf (COTS) computing devices [98] expose SCADA systems to a vast assortment of cyber attacks, including denial-of-service (DoS) attacks, data interception, data alteration, and false data injection. Vulnerabilities of cyber-enabled compo-

nents will lead to not only cyber but also disastrous physical consequences. For instance, the Stuxnet worm infected field devices in a nuclear facility and forced centrifuge speeds outside the normal operating range [91]. Among all the current threats to SCADA systems, botnets are at the top of the list as it is an effective way to launch miscellaneous attacks and disseminate malicious software [117]. Peer-to-peer (P2P) botnets are an emerging threat to SCADA systems because they are more resilient to existing takedown measures [94]. To secure the increasingly interconnected SCADA systems with more and more cyber-enabled components, it is important to detect P2P botnets.

Unlike early botnets relying on a centralized command and control (C&C) server, a P2P botnet has a decentralized C&C infrastructure, allowing bots to exchange C&C messages in a P2P manner [117]. Several notable examples of P2P botnets, such as Sality, Kelihos, and ZeroAccess, have been alive in the wild for a long time and can be leveraged to launch cyber attacks on SCADA systems. Recently proposed detection schemes (e.g., [96, 132]) are typically designed and evaluated under Internet environment, but little work has been done for P2P-botnet detection in SCADA systems. Some existing methods require the installation of sophisticated, operating-system-specific software to monitor system calls, leading to a high deployment barrier and/or operating costs for SCADA systems. Furthermore, SCADA traffic has several distinct characteristics that are not present in the Internet traffic [9]. Both the unique characteristics of SCADA traffic and the potentially devastating impacts of cyber attacks entail the design of a P2P-botnet detection algorithm for SCADA systems.

In this section, we design a P2P-botnet detection method for SCADA systems leveraging the traffic monitoring capabilities of the SCADA network infrastructure. Our evaluation results show that our method achieves high detection accuracy with very few false positives under different scenarios. The contributions of this work are listed as follows:

- We study the problem of P2P botnet detection for SCADA systems, which have distinct traffic characteristics and diverse architectures. We show that P2P bots (i.e., cyber-enabled components infected by a certain P2P botnet) can be identified in SCADA systems by analyzing flow statistics collected by networking devices.

- As many SCADA systems organize its sensors, actuators, and protection devices in a

decentralized fashion, the corresponding traffic patterns bear certain resemblances to P2P communication. To filter out non-P2P hosts and reduce the data volume for botnet detection, we incorporate a simple feature test in the preprocessing stage of our method to identify hosts engaging in (benign or botnet) P2P communication.

- To set apart P2P bots from benign P2P hosts, the bot identification stage of our method jointly considers their flow-based and connectivity-based behaviors. We propose an unsupervised approach in this stage, which not only identifies known P2P botnets but also alerts system administrator to newly emerged ones.

- In our evaluation, we consider the scenario where some bot-infected hosts simultaneously run P2P SCADA applications and show that our method is able to separate hosts infected by different types of P2P botnets from those running benign P2P applications.

### 4.1.2 Related Work

#### SCADA System Architecture and Traffic Patterns

Many SCADA systems implement sensing, computation, and control tasks through a decentralized or distributed architecture. For example, in smart grid substations, sensors and actuators are deployed near power system equipment (e.g., transformers and circuit breakers). Data collected by sensors is transmitted through the SCADA network infrastructure to multiple SCADA hosts (e.g., protective relays). Deriving the states of the power system from the data, different SCADA hosts cooperate to implement control functionality by exchanging information with each other and sending commands to actuators. SCADA traffic patterns bear certain resemblances to P2P communication, which is also decentralized in nature.

SCADA systems exhibit traffic characteristics that are significantly different from the Internet, which may render existing detection schemes targeting Internet environment less effective. In [9], traffic characteristics of SCADA systems in two water treatment and distribution facilities are studied. Unlike Internet traffic that is strongly correlated with human activity, it is found that SCADA traffic is not self-similar and does not present obvious diurnal or nocturnal patterns. Instead, the studied SCADA systems are dominated by periodic traffic generated

by sensors, resulting in a large number of flows with nearly constant throughputs. From the perspective of network traffic monitoring and analysis, a SCADA system may generate the following categories of traffic patterns to support its various tasks and services:

- *Periodic data transmission.* Sensors can be configured to transmit data to other SCADA hosts periodically and autonomously. A sensor may support multiple sampling frequencies simultaneously to facilitate tasks requiring different levels of data granularity.
- *Periodic polling.* A SCADA host (e.g., a protective relay) can periodically request data from a set of sensors it specifies. A task relying on periodic polling is activated at regular time intervals.
- *Unsolicited response.* In this mode, sensors can spontaneously initiate message transmission to report status change or event occurrence without receiving any polling command. This mode is supported by DNP3.
- *Event-triggered commands.* Using the data collected from sensors, a SCADA host can determine whether certain commands need to be sent. It is common to observe multiple command sequences on a SCADA network during the execution of event-triggered tasks.
- *P2P data transmission.* To increase system resilience in the presence of faulty or compromised devices, P2P communication is also leveraged by SCADA systems. In [44], a self-organized structured P2P overlay is deployed on top of SCADA network, taking full advantage of path redundancy and data replication that are inherent to P2P technology. In the P2P convergecast application [43], data sources and sinks can be located on various hierarchical levels of an interconnected SCADA system. These communicating peers may be geographically far apart from each other.

### Data Collection for Botnet Detection

The life cycle of a P2P botnet consists of multiple stages (or phases), i.e., infection stage, rally stage, waiting stage, and execution stage [95]. One of the most essential characteristics of P2P botnets is their C&C channel. In the rally stage, an infected bot performs peer discovery and joins the P2P botnet. In the waiting stage, it waits for commands from the botmaster, which

are delivered in a P2P fashion. Without relying on a centralized C&C server, P2P botnets are found to be resilient to takedown measures targeting centralized C&C infrastructure [94, 109]. Many P2P-botnet detection techniques distinguish C&C communication patterns from regular ones over the Internet.

To identify a botnet through its C&C communication patterns, data needs to be collected by monitoring the behaviors of individual hosts and/or network traffic flows. In MalFlow [131], host-level data flow profiles are constructed to identify malign network domains by intercepting network-related system calls. However, such a host-based approach monitors the behaviors of processes running on individual hosts, which is not feasible for SCADA systems because the incurred overhead can affect the performance of safety-critical real-time SCADA tasks.

Another approach to data collection is to capture packet traces on individual network hosts. Packet traces are then converted into a conversation-based or flow-based data set. A conversation is uniquely identified by a two-tuple (i.e., source and destination addresses). Time series of packet sizes and packet inter-arrival timestamps of individual conversations can be extracted for feature selection [95, 96]. A network flow is identified by a unique five-tuple (i.e., source and destination IP addresses, source and destination ports, and protocol). A recently proposed detection system [69] extracts flow-based statistical features from time series of packet sizes and inter-arrival timestamps of packets. Collecting packet traces on individual SCADA hosts may also introduce significant system overhead and affect the performance of critical SCADA tasks.

A third way to collect network-level data is to leverage the traffic monitoring capabilities built into switches and routers (e.g., NetFlow records). In addition to the five-tuple that uniquely identifies a network flow, a flow record can include flow arrival timestamp, flow duration, as well as summarized information such as the number of packets and the number of bytes transmitted. Implemented in many networking devices deployed in SCADA systems, traffic monitoring allows system operators to configure how flow records should be collected. When a flow terminates or its pre-specified timer expires, the corresponding flow record is exported to traffic analysis server designated by the system administrator. As collecting flow records does not incur extra overhead on existing SCADA hosts, sensors, and actuators, a

botnet detection method leveraging flow records has a low deployment barrier for SCADA systems. In PeerClean [132], NetFlow records are collected at the edge routers of a campus network and supervised multi-class SVMs are trained to label P2P bot clusters. In contrast, our approach is unsupervised and is able to differentiate newly emerged bots provided that they exhibit distinct flow-based and/or connectivity-based characteristics.

### 4.1.3 System Overview

The primary objective of this work is to design a P2P-botnet detection method that can be easily integrated into existing and emerging SCADA systems. Figure 4.1 shows the architecture of our proposed P2P-botnet identification system.

**Data Collection and System Deployment:** Network flow records are collected by networking devices (e.g., Ethernet switches and routers) of a SCADA system. A server is set up to collect the flow records and execute our P2P-botnet detection algorithm. For a hierarchically organized SCADA system, multiple detection servers can be distributedly deployed in subsystems at different levels (e.g., substation automation systems and control center network). To minimize data transmission workload between subsystems, each detection server is responsible for identifying P2P bots within its own subsystem. A detection server can be configured to signal the system administrator on site and/or send alarms to a remote location.

**Detection Algorithm:** Our two-stage detection algorithm is deployed on the detection server. In the first stage, flow records for both non-P2P and P2P hosts are used as input. We extract the good connection ratios of individual hosts and perform a simple feature test to filter out non-P2P SCADA hosts. This step identifies P2P hosts on the SCADA system and significantly reduces the data volume for the second stage. Our feature test uses good connection ratios because SCADA hosts engaging in P2P communication generate many failed

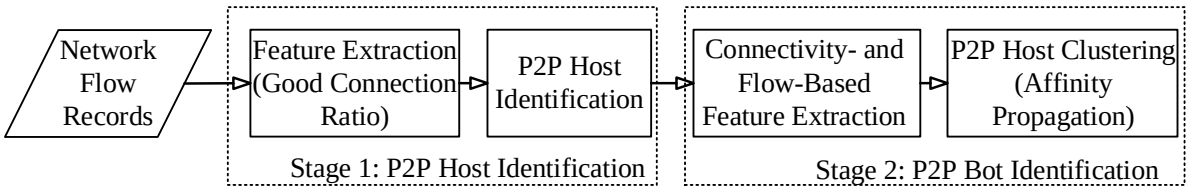



Figure 4.1: System architecture overview

connections during peer discovery. P2P hosts identified by the first stage may include those running P2P SCADA application (i.e., benign P2P hosts), non-P2P hosts infected by P2P botnet, and hosts running both P2P bot processes and P2P SCADA applications. In the second stage, we use an unsupervised learning algorithm, i.e., affinity propagation clustering, to classify benign P2P hosts and different types of known P2P bots. By jointly considering flow-based and connectivity-based features that characterize the C&C communication patterns of botnets, our algorithm can not only set apart known P2P bots but also detect P2P bots of unknown types. Our detection method is executed at the end of each monitoring period to identify P2P bots. Various lengths of monitoring period are supported provided that P2P bots generate enough network flows for C&C communication during that period. System operator can configure the length of detection periods so that P2P bots are identified in a timely manner.

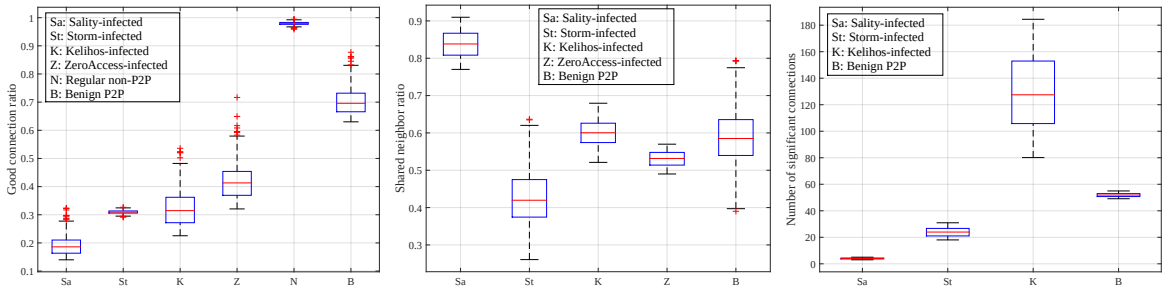
### 4.1.4 System Design

To design an effective method for P2P-botnet detection, we select features that differentiate P2P bots from other hosts in SCADA systems. In this section, we describe our design of the two stages of our algorithm and discuss the rationale behind their feature extraction modules.

#### P2P Host Identification

A SCADA system generates a huge volume of control and monitoring traffic. The majority of these flows are associated with regular non-P2P SCADA applications. Only a few tasks requiring data redundancy and resilience to sensor faults leverage P2P communication. Therefore, we first filter out hosts running only non-P2P applications. To ensure the scalability of this step, we design a simple feature test using the good connection ratio feature.

**Good Connection Ratio:** A pair of hosts share a "good" TCP connection if they complete a three-way handshake (i.e., SYN, SYN/ACK, and then ACK). They share a "good" UDP connection if UDP flow generated by one host is "responded" by the other. During monitoring period $T_i$, the good connection ratio for a host pair is defined as the ratio of the number of good connections to the total number of connections between them. As a




(a) Good connection ratio feature (b) Shared neighbor ratio feature (c) Significant connections

Figure 4.2: Connectivity-based features.

SCADA host communicates with multiple other hosts, a set of pairwise good connection ratios is associated to it. We compute the mean and standard deviation over the set of good connection ratios of a host $j$ and denote this feature vector as $G_j^i$. Figure 4.2a shows the pairwise good connection ratios associated to different types of hosts in a simulated SCADA system over 24 hours (see Section 4.1.5). Note that regular SCADA hosts only running non-P2P applications have significantly higher good connection ratios than P2P hosts. Hosts in SCADA systems typically assume well-defined roles in various tasks: Sensors monitor power-system equipment and send data to hosts that require the states of these equipment. These hosts respond to sensors to confirm data delivery, making connections from sensors good connections. Protective relays detect power-system events and send commands to actuators (e.g., circuit breakers), which have to respond to these commands by sending updated states of the power-system equipment being controlled. Therefore, the majority of connections between actuators and protective relays are also good ones. On the other hand, both P2P SCADA applications and P2P bots need to perform peer discovery so as to join a P2P network, which results in bad connections if some of the peers are not active or in states other than peer

Table 4.1: Connectivity-based statistical features

| **Feature** | **Description** |
|---|---|
| Good connection ratio | mean and standard deviation of per-host pairwise good connection ratios |
| Shared neighbor ratio | mean and standard deviation of per-host pairwise shared neighbor ratios |
| Number of significant connections | number of significant connections for each host |

discovery (e.g., the execution stage of P2P bots). We note that the good connection ratio feature is also used in the next stage to help differentiate P2P bots from benign P2P hosts. Both the mean and standard deviation (i.e., the $G_j^i$ feature vector) are used in the second stage.

**P2P Host Identification:** We observe that the hourly average good connection ratios associated to individual hosts can be used to identify P2P hosts. We inspect the 24-hour training data set (see Figure 4.2a) and empirically determine a threshold of 0.94. The P2P host identification stage consists of a threshold-based feature test: If a host has an hourly average good connection ratio over 0.94, we label it as a non-P2P SCADA host at the end of the monitoring period. Otherwise, it is labeled as a P2P host for further processing in the next stage. Our experiment (see Section 4.1.5) shows that all the non-P2P hosts can be filtered out by this simple feature test, which significantly reduces the data volume for further analysis.

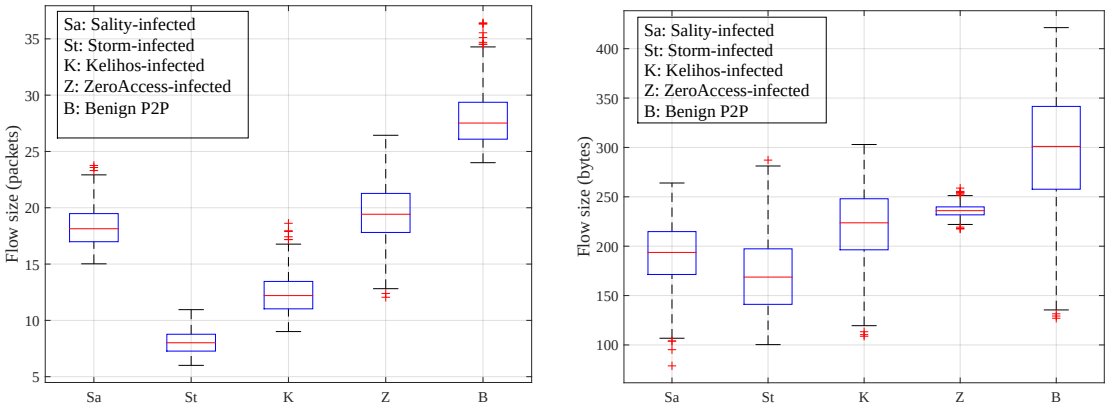
### P2P Bot Identification

To identify hosts infected by P2P botnet, we jointly consider two sets of features to distinguish P2P-botnet C&C traffic patterns from those generated by benign P2P hosts. The first set of features includes connectivity-based features listed in Table 4.1. We use these feature to capture the unique characteristics of connectivity patterns of various P2P botnets.

**Good Connection Ratio:** As shown in Figure 4.2a, good connection ratio can be further exploited to differentiate P2P bots and benign P2P hosts. P2P SCADA applications are typically used in a small set of hosts whose tasks require data redundancy and/or resilience to interruption. After the first several rounds of peer discovery, each host will retain an up-

Table 4.2: Flow-based statistical features for P2P bot identification

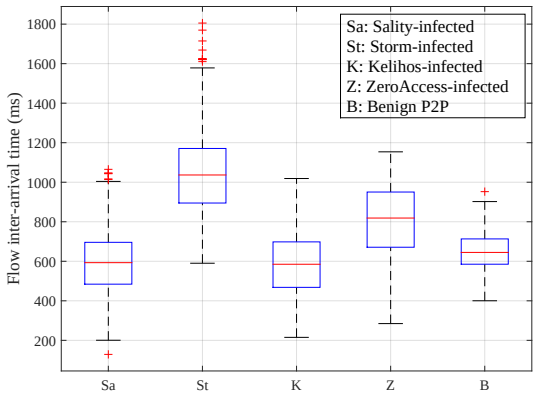
| Feature | Description |
|---|---|
| Packet-oriented flow size | mean and standard deviation of per-host flow sizes in number of packets |
| Byte-oriented flow size | mean and standard deviation of per-host flow sizes in number of bytes |
| Flow inter-arrival time | mean and standard deviation of flow inter-arrival timestamps for each host |
| Flow duration | mean and standard deviation of flow durations for each host |



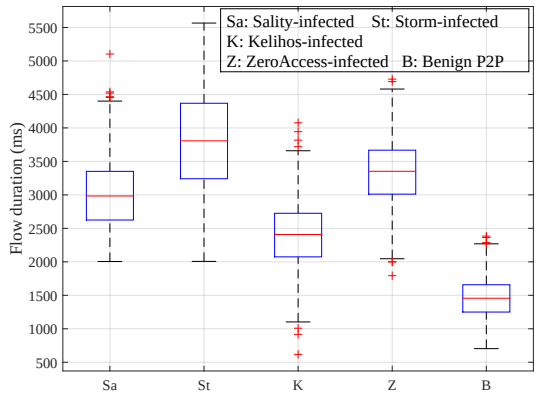

(a) Packet-oriented flow size feature




(b) Byte-oriented flow size feature




(c) Flow inter-arrival time feature




(d) Flow duration feature

Figure 4.3: Flow-based statistical features

to-date list of peers, leading to a higher good connection ratio afterwards. Using mean and standard deviation of per-host pairwise good connection ratios, we can also set apart different P2P bots. Although most P2P bots have good connection ratios lower than those of benign P2P hosts, their respective C&C communication patterns lead to distinctive characteristics of good connection ratios.

**Shared Neighbor Ratio:** During monitoring period $T_i$, we find the set of P2P hosts host $j$ has contacted and denote this set by $n_j^i$. For any host $k \in n_j^i$, we find the set $n_k^i$. The pairwise shared neighbor ratio of host pair $\{j,k\}$ is defined as $n_{(j,k)}^i = \frac{||n_j^i \cap n_k^i||}{||n_j^i \cup n_k^i||}$. Mean and standard deviation of the set of pairwise shared neighbor ratios of host $j$ is used as a feature for $j$. We denote this feature vector by $N_j^i$. Figure 4.2b shows the pairwise shared neighbor ratios of different P2P hosts over 24 hours. This feature vector can help us effectively differentiate different types of P2P bots. In particular, Sality-infected hosts have high shared

neighbor ratios because they share the same bootstrap list of peers. As all SCADA hosts must remain in operation, contents of the peer lists of Sality-infected hosts gradually evolve to become similar.

**Number of Significant Connections:** At the end of monitoring period $T_i$, we count the number of significant connections of individual P2P hosts. For host $j$, this scalar feature is denoted by $S_j^i$. Figure 4.2c shows the significant connection counts of different P2P hosts over 24 hours. We observe that ZeroAccess-infected hosts have virtually no significant connections, which indicates that ZeroAccess is designed to evenly distribute its C&C workload. As shown in Figure 4.2c, this feature can effectively set apart Sality bots, Storm bots, Kelihos bots, and benign P2P hosts.

The second set of features includes flow-based statistical features listed in Table 4.2. We use these features to characterize the traffic flows generated by P2P bots of different types.

**Packet-Oriented Flow Size:** At the end of monitoring period $T_i$, we compute the mean and standard deviation of flow sizes in number of packets for every individual host. Note that inbound and outbound flows are considered together. For host $j$, we denote its packet-oriented flow size feature vector as $P_j^i$. Figure 4.3a shows the packet-oriented flow size statistics for different types of P2P hosts over 24 hours. Packet-based flow sizes can be used to set apart benign P2P SCADA hosts from bot-infected hosts. P2P SCADA applications transmit sensor data to other hosts exploiting the inherent path redundancy of P2P technology. The number of packets transmitted by a benign P2P host outnumbers that of a P2P bot because its P2P connections are used to transmit multiple data packets sampled at high rates: A sensor may need to collect as many as 256 samples per cycle (i.e., 1/60 second) and each sample has to be sent immediately to ensure control system responsiveness. In contrast, P2P bots transmit fewer packets over established connections. These packets contain botmaster commands that need to be disseminated among bots. The differences in packet-oriented flow sizes among P2P bots can be attributed to their peer discovery strategies and command sets. Bots with a richer command set tend to have more possible states and more complicated control sequences.

**Byte-Oriented Flow Size:** At the end of monitoring period $T_i$, we compute the byte-oriented flow size feature vector $B_j^i$ for host $j$, which includes mean and standard deviation

of inbound and outbound flow sizes in number of bytes. Figure 4.3b shows the byte-oriented flow size statistics of different P2P hosts over 24 hours. P2P SCADA applications are used to transmit sensor data packets which are normally in several hundred of bytes, whereas many C&C packets are of shorter lengths. The differences in traffic volumes of flows of various P2P bots reflect their unique characteristics of both C&C sequences and encoding schemes of commands.

**Flow Inter-Arrival Time:** In addition to flow sizes, we characterize the temporal patterns in which a P2P host communicates with its peers. For monitoring period $T_i$, we collect and sort the inbound and outbound flow arrival timestamps of host $j$, which results in a time series of flow arrival timestamps. We subtract timestamp $T_i$ from the first element of the time series and then generate a time series of flow inter-arrival timestamps. Mean and standard deviation of inter-arrival timestamps are extracted as a feature vector for host $j$, which is denoted as $I_j^i$. In a SCADA system, remote servers can collect data from sensors within the system via the remote terminal unit (RTU) [43]. Although sensors may aggregate data sampled at different time intervals before transmission, many safety-critical SCADA tasks require low latency in data delivery, limiting the amount of data accumulated. This also explains the smaller standard deviation of flow inter-arrival timestamps for benign P2P hosts in Figure 4.3c. In addition, Figure 4.3c indicates that this feature vector is not very effective in differentiating Sality-infected bots from Kelihos-infected bots because the ranges of their flow inter-arrival time almost completely overlap.

**Flow Duration:** For monitoring period $T_i$, we collect inbound and outbound flow durations for host $j$ to form a time series of flow durations. Mean and standard deviation of this time series are then computed and used as a feature vector for host $j$. We denote this feature vector by $D_j^i$. Flow durations of benign P2P hosts are relatively shorter than those of P2P bots. This can be attributed to the fact that data packets generated by a sensor can be transmitted through different P2P connections: During each cycle, a sensor collects multiple data samples, demultiplexes them over different established connections, and then tears down the connections. In contrast, flow durations of P2P bots are typically longer because these bots are deliberately designed to communicate stealthily to evade the radars of existing detection

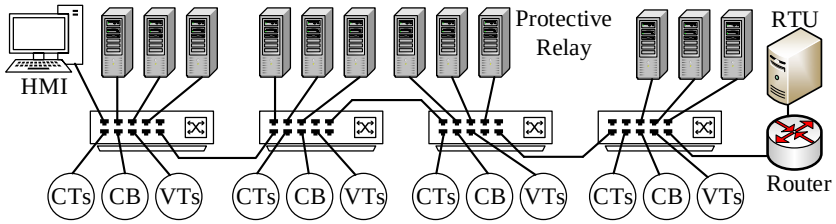



Figure 4.4: A substation SCADA system simulated using OMNeT++

systems.

**P2P Botnet Clustering.** We apply feature normalization to all the elements in feature vector $x_j^i$: For feature element $x_j^i(l)$, we compute $\frac{x_j^i(l)-x_{l,\min}}{x_{l,\max}-x_{l,\min}}$, where $x_{l,\min}$ and $x_{l,\max}$ are the minimum and maximum feature element values among all hosts during period $T_i$, respectively. Similarities among all the pairs of P2P hosts, which are computed based on normalized features, are used by the affinity propagation algorithm to automatically find the proper number of clusters for a presented data set. Suppose that the normalized features for hosts $j$ and $k$ during monitoring period $T_i$ are denoted by vectors $\bar{x}_j^i$ and $\bar{x}_k^i$, respectively. The similarity $s^i(j,k)$ between the two host is defined as the negative Euclidean distance, i.e., $s^i(j,k) = -||\bar{x}_j^i - \bar{x}_k^i||^2$. As our bot identification algorithm is unsupervised, we expect that it can identify both known and unknown P2P bots.

### 4.1.5 Evaluation

Without loss of generality, we construct two evaluation scenarios by simulating substation SCADA systems.

**Experiment I – Detecting Unknown Bots:** To demonstrate the effectiveness of our proposed method, we first evaluate whether our system can identify known and unknown bots in a SCADA system. The SCADA system simulated in OMNeT++ is depicted in Figure 4.4. Current sensors (CT), voltage sensors (VT), and circuit breakers (CB) are deployed near power-system equipment (e.g., transformers). Traffic patterns between protective relays and sensors can be periodic data transmission, periodic polling, or unsolicited response. Traffic patterns between protective relays and circuit breakers are periodic polling and/or event-triggered commands. A remote control center communicates with the simulated substation

Table 4.3: P2P Botnet Identification Performance (A-Accuracy, P-Precision, R-Recall)

| **Bot Type** | **Experiment I (A/P/R)** | **Experiment II (A/P/R)** |
|---|---|---|
| Sality | 98.6%/95.6%/91.7% | 97.1%/95.6%/89.6% |
| Kelihos | 98.1%/87.0%/97.9% | 96.7%/90.0%/93.8% |
| ZeroAccess | 99.3%/94.1%/100% | 98.8%/94.1%/100% |
| Storm | 98.3%/90.1%/93.8% | 97.9%/92.2%/97.9% |

via the RTU using multiple protocols including P2P convergecast [43]. System operators can also monitor and control the substation through the human-machine interface (HMI). There are 8 SCADA hosts generating regular non-P2P traffic. In addition, there are 10 P2P hosts where there are 2 Sality-infected hosts, 2 Kelihos-infected hosts, 2 ZeroAccess-infected hosts, 2 Storm-infected hosts and 2 P2P hosts generating benign P2P traffic. All the bots generate malicious P2P traffic while simultaneously running non-P2P SCADA applications. We generate 24-hour training data without activating the Storm bots and train the clustering algorithm. Then, we collect testing data by running the simulation for another 24 hours and activating the storm bots 8 hours after simulation starts. The second column in Table 4.3 summarizes the detection performance of our proposed method on the testing data set over 24 monitoring periods (i.e., length of $T_i$ is set to 1 hour). We note that the simple feature test employed in the first stage filters out all the 8 regular non-P2P hosts without incorrectly including any P2P hosts. Before Storm bots are activated, 4 clusters are generated by the clustering algorithm at the end of each monitoring period: 1 for benign P2P hosts, and 3 for the bots seen during training. After Storm bots are activated, a new cluster containing Storm-infected hosts is generated. This indicates that our method can be used to identify previously unseen P2P botnets and signal system administrator to take necessary measures. We further note that our method does not classify any bot as benign P2P hosts.

**Experiment II – Detecting P2P hosts infected by P2P bots:** We further consider the scenario where all the hosts run P2P SCADA applications. A SCADA system with 10 P2P SCADA hosts are simulated: There are 2 Sality-infected hosts, 2 Storm-infected hosts, 2 Kelihos-infected hosts, 2 ZeroAccess-infected hosts, and 2 benign P2P hosts communicating with remote data collection servers. After training on a 24-hour data set, we evaluate our method on a 24-hour testing set and summarize the detection performance in the third column

of Table 4.3. At the end of each monitoring period, the clustering step generates 5 clusters. Although the first stage of our algorithm does not filter out any host, the second stage is still able to achieve high detection accuracy with very few false positives and no false negatives.

## 4.2 Detecting Attacks on the DNP3 Protocol

### 4.2.1 Introduction

Monitoring and controlling critical infrastructure of industrial control system (ICS), Supervisory Control and Data Acquisition (SCADA) network interconnects field devices, engineering workstations, human-machine interfaces (HMIs), and various servers to ensure reliable system control and operation. As interconnectivity among SCADA hosts continues to increase, *network-based cyber attacks* (i.e., cyber attacks that exploit SCADA network vulnerabilities to realize their malignant purposes) have become one of the primary cyber threats to SCADA systems. As SCADA networks utilize both widely-used network protocols from the IT domain and specialized SCADA network protocols, SCADA hosts can become the target of both *conventional network-based attacks* (e.g., a malware targeting Windows-based engineering workstations) and *specialized attacks on SCADA protocols* (e.g., [40,45]). In recently reported cyber-security incidents (e.g., [73, 77]), it is observed that network-based cyber attacks are a major component of sophisticated, multiple-stage attacks on critical ICS infrastructure. However, intrusion detection systems (IDSs) for IT networks cannot be directly ported into SCADA networks because traffic characteristics of SCADA hosts are shown to be distinctly different from hosts in corporate/campus networks [10]. To protect modern ICSs, it is therefore vital to design SCADA network intrusion detection solution that takes into account both operation needs and distinct traffic characteristics of SCADA systems.

Real-world attacks on SCADA systems tend to follow a step-by-step method to compromise critical system devices [73,85,97], and network-based attacks may be launched at different steps to achieve malicious goals. Take the SCADA system in Fig. 4.5 as an example. Suppose that the ultimate goal of attackers is to induce severe damages to the physical process monitored and controlled by the SCADA system. If the attackers launch the Stuxnet attack [73],

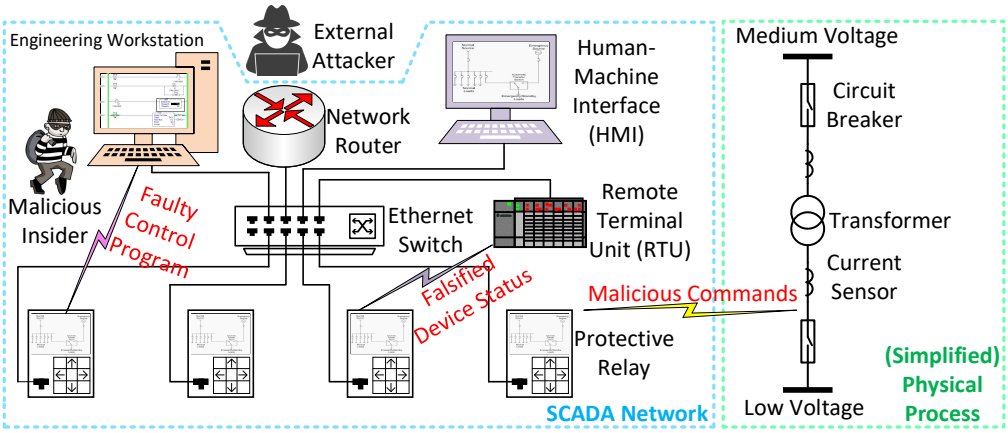



Figure 4.5: Network-based attacks on SCADA system.

the Windows-based engineering workstation will first be compromised. This step can be accomplished by a malicious insider with a flash drive containing the Stuxnet malware. Then, the attackers can download malicious or faulty control programs to field devices such as protective relays. The compromised field devices may issue malicious network commands (e.g., tripping circuit breakers when no faults occur) or generate falsified device status reports to mask the execution of malicious commands. In the CrashOverride attack [2, 77], knowledge on SCADA communication protocols is exploited, and external attackers can even directly control system equipment without relying on the software within the engineering workstation. As shown in [2, 77, 129, 142], emerging cyber attacks on SCADA systems tend to be composed of multiple "primitive" attacks, many of which are network-based attacks. For instance, traces of both conventional attacks such as passive reconnaissance and active service/host discovery, and specialized attacks on SCADA protocols (e.g., falsified control commands), are found in the CrashOverride attack. Hence, network intrusion detection solution for SCADA systems must be able to combat both conventional as well as specialized network attacks.

In this section, we propose a deep-learning-based network intrusion detection system for SCADA networks, detecting network-based cyber attack primitives. The contributions of this work are as follows:

- *Deep-learning-based IDS for SCADA networks.* Instead of relying on hand-crafted features of individual network packets, our proposed approach employs a convolutional neural network (CNN) to characterize salient temporal patterns of network behaviors

of SCADA hosts. Leveraging traffic monitoring capabilities of existing SCADA networking devices, our approach inspects network traffic without interrupting SCADA system operation, which significantly lowers the barrier to deploying our solution.

- *Detection of SCADA network attack primitives.* Detecting both conventional network attacks and those purposely crafted for SCADA network protocols, our proposed approach will facilitate the detection of sophisticated, real-world attacks that launch different network attack primitives at different stages.

- *Capability to adapt to different SCADA environments.* Our solution provides a retraining scheme, enabling SCADA system operators to refine our neural network models with site-specific network attack traces and facilitating the rapid adaptation of these models to various SCADA environments that are subjected to different cyber attacks.

- *Evaluation on realistic data sets.* We design and evaluate our proposed network IDS using traffic data collected from different energy-delivery system test beds with real field devices running DNP3 protocol. Our results show that our approach achieves high detection accuracy with few false positives, making it well-suited for network intrusion detection in SCADA systems.

### 4.2.2 Related Work

#### Network-Based Cyber Attacks on SCADA Systems

In recently reported cyber-security incidents [2, 73, 77, 92], it is found that many sophisticated cyber attacks exploit certain network attack primitives to penetrate from corporate networks into SCADA networks, determine roles of different SCADA hosts, and cause damages to physical systems. *Primitive network-based cyber attacks* can be classified into two categories, i.e., conventional attacks exploiting vulnerabilities of IT network protocols [142] and specialized attacks on SCADA network protocols. Taxonomies and analyses of attacks on SCADA network protocols can be found in surveys on ICS security, such as [37, 129]. To detect real-world attacks, a SCADA network intrusion detection system must take into account both conventional and specialized attacks. In this work, we focus on SCADA systems running DNP3

protocol [54], which is widely adopted in SCADA systems across the world.

#### Intrusion Detection for SCADA Systems

SCADA hosts exhibit traffic characteristics that are significantly different from hosts in corporate/campus networks [10]. Therefore, intrusion detection systems for IT networks do not work well in SCADA network environments. Recently, a survey on intrusion detection solutions for industrial control systems is presented in [50]. It is found that many existing solutions utilize widely-used network security mechanisms, such as white-listing, for network intrusion detection in SCADA networks. However, such approaches are not effective for purposely crafted attacks on SCADA network protocols because legitimate SCADA network packets and attack packets can share the same application-layer header fields (e.g., [37]). To detect specialized attacks, application-layer information needs to be analyzed in detail to distinguish malicious packets from benign ones [87]. However, a wide variety of SCADA-specific attacks reviewed in [129,142] cannot be detected by the approach proposed in [87] because of the lack of a mature way of modeling network behaviors of SCADA hosts. In [100], machine-learning-based algorithms are proposed to model network behaviors of SCADA hosts. However, the features selected in [100] cannot comprehensively model SCADA host behaviors (e.g., a robust model for temporal network behaviors is yet to be established) and thus many attacks reported in [129] are not considered. In this work, we analyze network packets up to their application-layer headers and comprehensively model the network behaviors of SCADA hosts using a convolutional neural network (CNN).

### 4.2.3 System Design Overview

#### Threat Model

In our network IDS solution for SCADA systems, we assume that an attacker is capable of compromising certain SCADA hosts or slipping in a new device to launch network-based cyber attacks. For instance, in sophisticated cyber attacks launched by external attackers (e.g., [73,77]), engineering workstations and field devices in SCADA systems are compromised. In addition to passive eavesdropping, some of the compromised hosts are exploited to issue

malicious control commands. As another example, a malicious insider may either directly install malware on SCADA hosts [40, 134] or plant a new device [86] to launch attacks. Once a SCADA host is compromised (or a new device is planted), it may be exploited to send malicious attack packets with or without simultaneously generating normal/regular SCADA network traffic. However, this host will probably not use any network protocols that have not been previously observed on the SCADA network. Otherwise, it will easily be flagged or blocked by existing ICS intrusion detection tools [50].

We also assume that a malicious SCADA host must generate network packets at a certain attack stage. This is because our proposed detection algorithm identifies malicious network activities by analyzing packets generated by all the SCADA hosts. A malicious host can, for example, issue malicious control commands to cause system failures, mask impacts of malicious control actions by sending manipulated device status updates, and learn about the functions of other hosts by reading their detailed configurations [37]. However, if a cyber attack does not generate any network packet revealing its malignant purposes (e.g., [3]), operators must resort to other intrusion detection solutions (e.g., host-monitoring-based methods [124, 126]) to protect the SCADA system.

Finally, we assume that network packets exchanged inside a SCADA site (e.g., a power substation) are not encrypted. This is the case because a SCADA site is typically secured physically and legacy field devices that do not support encryption/decryption are still widely in use.

### System Design

The architecture of our proposed network intrusion detection tool is depicted in Fig. 4.6. The input to our tool is raw SCADA network traces, i.e., network packets generated by all the SCADA hosts. In real-world SCADA systems, these packets can be collected by leveraging the traffic monitoring capabilities of network switches (e.g., the port mirroring feature [1] of Cisco switches). By properly configuring networking devices to duplicate all the network traffic to a monitoring port, our proposed approach will not interrupt normal system operation. The duplicated traffic is forwarded to a server, where our proposed detection algorithm is deployed.

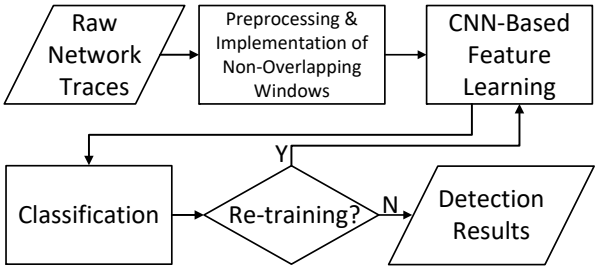



Figure 4.6: High-level architecture of the proposed network IDS solution.

Our detection algorithm first inspects each packet up to its application-layer header and extracts useful information for feature learning. Application-layer payload data are not analyzed by our algorithm, so an attack that manipulates process data will not be detected unless the corresponding application-layer header is also modified. The preprocessing stage extracts a 25-tuple from each packet and implements non-overlapping detection windows for the feature learning stage. Feature vectors of all packets contained in each detection window are passed to a convolutional neural network (CNN) for feature learning. At lower layers of the CNN stage, local characteristics of the tuple sequence are obtained. Then, at higher layers of the CNN stage, high-level salient patterns are extracted. The learned features are then fed to a classification stage to generate detection outputs. We attach a fully-connected softmax output layer to the feature-learning CNN for classification. In addition, outputs of the classification stage are also exploited as indicator for re-training. When a newly emerged attack is encountered, our algorithm will signal the SCADA system operator to inspect and label the corresponding time window. When the number of new attack instances are sufficiently large, a new class is added to the classification stage and the entire neural network is re-trained.

In essence, our detection method analyzes the temporal network behavior patterns of SCADA hosts and identifies time windows containing abnormal patterns. At the output end, our detection algorithm assigns a label to each detection window (e.g., "normal operation"). By inspecting detection windows containing anomalies, SCADA system operator can easily pinpoint the root cause and take proper measures to mitigate the attack impacts.

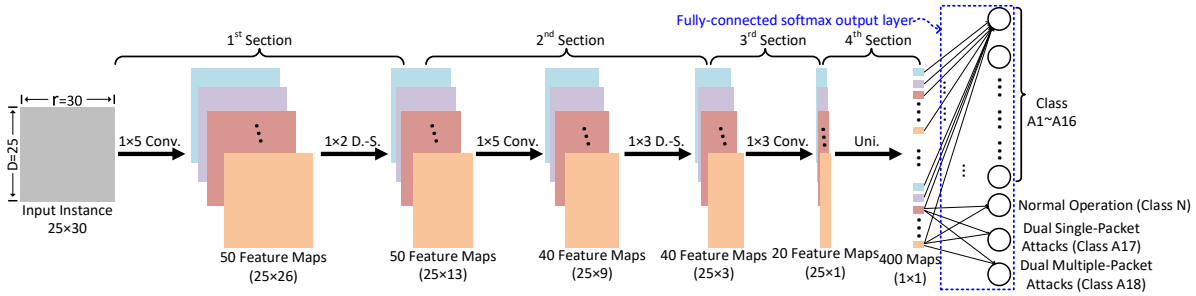



Figure 4.7: High-level architecture of the proposed neural network model with convolutional (Conv.) layers, down-sampling (D.-S.) layers, unification (Uni.) layer, and the classification layer.

### 4.2.4 Deep-Learning-Based IDS for SCADA Networks

#### Data Preprocessing

An operational SCADA network runs a set of network protocols for network connectivity and SCADA system operations. The data preprocessing stage extracts information from different types of network packets as follows:

- *Layer 2 packets.* For each Layer 2 packet, source and destination MAC addresses, as well as the EtherType value, are extracted. Size of the Layer-2 payload, excluding the 4-byte checksum, is also extracted.

- *Layer 3 packets.* Generally, IP and ICMP packets are present on SCADA networks. In addition to the 4 fields extracted from Layer 2, we extract source and destination IPs, protocol type, as well as the packet length field from IP packets. If the packet is an ICMP packet, the type and subtype fields are also obtained. If the packet is not an ICMP packet, both fields are set to 256.

- *TCP/UDP packets.* In addition to the 10 fields from Layers 2 and 3, we extract source and destination ports, TCP flags, sequence number, as well as acknowledgment number if a TCP packet is encountered. Size of TCP packet payload, excluding the TCP header, is also calculated and recorded. If the packet is a UDP packet, we extract source and destination port numbers. Additionally, UDP payload size excluding the UDP header is also computed. For a UDP packet, the TCP flags, sequence number, and acknowledgment number, are all set to 0.

- *DNP3 packets.* DNP3 packets can use either TCP or UDP. In addition to the 16 fields extracted from a DNP3 packet up to its TCP/UDP layer, we extract information from the DNP3 data-link, pseudo-transport, and application layers [54]. At the DNP3 data-link layer, we extract DNP3 source and destination addresses, the control byte value, as well as the length byte value. The one-byte pseudo-transport layer header is also extracted. At the DNP3 application layer, application control byte, function code byte, and internal indication field values are extracted.

Therefore, 24 fields are filled out for each packet at the preprocessing stage. In addition, *packet inter-arrival time*, i.e., the time interval between the arrival of the current packet and the arrival of its preceding packet is logged. For the first packet seen on the SCADA network, its inter-arrival time is set to 0.

For each packet, the output of the preprocessing stage is a 25-tuple. To capture temporal behavior patterns of SCADA hosts, we implement non-overlapping sliding windows on the sequence of SCADA packets collected from the monitoring port. The size of the detection window, in terms of the number of packets it contains, is denoted by $r$. These non-overlapping detection windows partition the multivariate sequence of packet feature vectors into small snippets. Each *input instance* to the feature learning and classification stages is a sequence of $r$ 25-tuples.

### Convolutional Neural Network (CNN) for Feature Learning

Each input instance used by the CNN stage is a two-dimensional matrix. Each matrix contains $r$ samples, and each sample has $D$ features. We choose $r$ to be the average number of packets observed within one second on a SCADA network over a long period of time (e.g., 24 hours). In our experiments (see Sec. 4.2.5), we choose $r = 30$. For SCADA systems based on DNP3, we choose $D$ to be 25 (see Sec. 4.2.4). In the training data set, the label of each instance (i.e., a $D \times r$ matrix) is determined by the type of network attack included in the corresponding detection window. Instances that do not include any attack packet share the same label, i.e., "normal operation". In the CNN, the $i$th layer's $j$th feature map is also a matrix. The element value at the $r$th column and the $d$th row is denoted by $v_{ij}^{d,r}, \forall d = 1, \cdots, D$. In the CNN stage,

multiple convolutional layers and pooling layers are employed.

**Convolutional Layers** At each convolutional layer, the input feature maps are convolved with several convolutional kernels, which are learned during the training process. The output of the convolutional operators is added by a bias (also learned during training) and then passed to the activation function to generate the feature maps for the next layer. The value $v_{ij}^{d,r}$ is thus given by

$$v_{ij}^{d,r} = \tanh\left(b_{ij} + \sum_{k}\sum_{p=0}^{P_i-1} \omega_{ijk}^{p} v_{(i-1)k}^{d,r+p}\right), \tag{4.1}$$

where $\tanh(\cdot)$ is the hyperbolic tangent function, $b_{ij}$ is the bias for the feature map, $k$ is the index for the feature map from the $i$–1 layer, $\omega_{ijk}^{p}$ is the value at position $p$ of the convolutional kernel, and $P_i$ is the convolutional kernel length.

**Pooling Layers** At each pooling layer, resolution of the input is reduced by pooling feature maps from the previous layer over their local temporal neighborhoods. A max pooling function is given by

$$v_{ij}^{d,r} = \max_{1\le q\le Q_i}\left(v_{(i-1)j}^{d,r+q}\right), \tag{4.2}$$

where $Q_i$ is the length of the pooling region. Pooling is essentially a down-sampling operation that increases the invariance of features to distortions on the inputs (e.g., caused by different TCP/UDP/DNP3 payload sizes).

Using the convolution and pooling operations defined above, we construct a CNN shown in Fig. 4.7. We group layers of the CNN into four *sections*. The first and the second sections share similar high-level architecture. First, a convolutional layer convolves the input or feature maps from the previous layer's output using a set of kernels, which are learned during the training process. Then, a rectified linear unit (ReLU) layer maps the output of the previous layer with the function $relu(v) = \max(v, 0)$. Next, a max pooling layer is deployed to pick out the maximum feature map over its temporal neighborhood. Finally, a normalization layer is

added to normalize the values in different feature maps from the previous layer with

$$v_{ij} = v_{(i-1)j}\big(\theta + \alpha \cdot \sum_{t \in M(j)} v_{(i-1)t}^2\big)^{-\beta}, \tag{4.3}$$

where $\alpha, \beta, \theta$ are hyper-parameters, and $M(j)$ is the set of feature maps used in normalization.

The third section consists of a convolutional layer, an ReLU layer, and a normalization layer. Pooling is not needed because the temporal dimension of each feature map becomes 1.

At the fourth section, a fully-connected layer is developed to unify the feature maps from the previous layer. Mathematically, the unification operation is defined by

$$v_{ij} = \tanh\big(b_{ij} + \sum_{k} \sum_{d=1}^{D} \omega_{ijk}^{d} v_{(i-1)k}^{d}\big). \tag{4.4}$$

This unification layer is followed by an ReLU layer and a normalization layer.

**Design of Classification Stage and Re-training Scheme**

The classification stage leverages the features learned from the CNN stage to label input instances. In this stage, a fully-connected output layer is attached to the CNN stage to map learned features to output classes. Output of this stage is governed by the softmax function

$$v_{ij} = \frac{\exp(v_{(i-1)j})}{\sum_{j=1}^{C} \exp(v_{(i-1)j})}, \tag{4.5}$$

where $C$ is the total number of output classes. An entropy cost function is constructed using the true labels assigned to the training instances and softmax function's probability output.

When one or multiple previously unseen attack instances are present in a detection window, output neurons in the classification stage will have output values that are low and possibly close to each other. This facilitates the design of a re-training scheme that allows SCADA system operator to refine our neural network models with site-specific network attack traces. During the detection (i.e., testing) process, if the classification stage outputs for all the existing classes fall below a threshold $R_{th}$ ($0 \leq R_{th} \leq 1$), we log the features extracted by the CNN stage for the corresponding detection window. Such a window will temporarily be label as "unknown",

indicating that it includes previously unseen network behavior patterns. When a sufficiently large number of "unknown" instances are collected, we perform k-means clustering based on Euclidean distance metric to find the number of distinct clusters and request the SCADA system operator to inspect and properly label these clusters. Labeled instances from these clusters are then pumped into the existing training set, and the entire neural network is re-trained with previously unseen attack classes added to the classification stage. Note that the value of $R_{th}$ needs to be empirically determined.

### 4.2.5 Evaluation

We evaluate our intrusion detection solution using SCADA traffic data collected from two SCADA cyber-security test beds, both emulating realistic operational SCADA networks for energy-delivery systems. One test bed is constructed by the National Center for Reliable Electric Power Transmission (NCREPT) at University of Arkansas. This test bed consists of a remote terminal unit (RTU), which controls multiple protective relays via serial connections. The RTU is connected to Ethernet, coordinating an HMI's access to the protective relays. From the HMI, an operator can periodically poll process data from the protective relays and/or issue control commands using DNP3. The other test bed is built by the cyber-security lab of the Electric Power Research Institute (EPRI). This test bed organizes a few dozens of protective relays from different vendors, RTUs, engineering workstations, and computing servers into a SCADA network. In addition to polling and sending commands, an engineering workstation can also update the configurations of protective relays using vendor-specific programming tools. SCADA hosts in both test beds have similar average DNP3 packet rates (i.e., about 120 packets per minute), and we monitor network traffic on both test beds for more than 8 hours.

#### Data Set Overview

After collecting realistic SCADA network traffic from different test beds, we create our data set as follows. First, packets generated by each SCADA host are extracted. Then, we edit the destination and source IP addresses of the packets so that SCADA hosts from different

test bed can reside on the same local area network. Next, a set of network-based attacks on SCADA systems are injected into the data set. We create the attack packet sequences according to [37, 129, 142]. Specifically, network packets associated with the following attacks are created:

- *ARP spoofing attacks (A1) [142].* Such attacks can be launched by a device planted by a malicious insider or a SCADA host compromised by an external attacker. In SCADA environments, ARP spoofing attack can be launched at the beginning of a man-in-the-middle attack to hijack communication sessions between normal, legitimate SCADA hosts. It can also be launched to link multiple IPs to a single SCADA host, overloading the target host or causing its failure.

- *SYN flooding attacks (A2) [142].* For legacy SCADA field devices with limited computation capacity, such attacks can devour their system resources and block other SCADA hosts from establishing legitimate connections.

- *TCP RST attacks on telnet (A3).* In the collected SCADA traffic traces, we find that protective relays from a popular vendor can be reconfigured by the engineering workstation via Telnet. A malicious SCADA host can falsify a TCP packet from the engineering workstation with the RST flag set, effectively shutting down the reconfiguration session between engineering workstation and protective relay.

- *UDP flood attacks (A4).* UDP flood attacks can also be launched to overwhelm legacy SCADA field devices, which have limited computation resources.

- *DNP3-specific attacks (A5∼A16) [37, 129].* DNP3 attacks reported in the literature are also created and injected into the data set. These attacks can be categorized into two types. The first type is single-packet attacks. Each type of these attacks can be implemented by modifying a single packet in a normal DNP3 communication session. Single-packet attacks include application termination attacks (A5), outstation DFC flag attacks (A6), function reset attacks (A7), unavailable function attacks (A8), address alteration attacks (A9), outstation write without reading (A10), clear object attacks (A11), outstation data reset attacks (A12), and configuration capture attacks (A13).

The second type is multiple-packet attacks, which can be realized by modifying multiple packets in a normal DNP3 session. Multiple-packet attacks include data-link layer length overflow attacks (A14), fragmented message interruption attacks (A15), and pseudo-transport layer sequence modification attacks (A16).

In addition to the "normal operation" class (i.e., class N) and attack classes A1∼A16, we also include two special classes (see Fig. 4.7), i.e., "dual single-packet attacks" (i.e., class A17) and "dual multiple-packet attacks" (i.e., class A18). A detection window with the label "A17" includes two single-packet attack instances, which can be any combination of instances from classes A5∼A13. A window with the label "A18" includes two multiple-packet attack instances, which can be any combination of instances from classes A14∼A16.

**Experiment Settings and Evaluation Results**

Throughout our experiments, we choose $\alpha = 3.5 \times 10^{-4}, \beta = 0.7, \theta = 1, R_{th} = 0.125$, and the size of $M(j)$ is 5. To choose the parameters in the CNN, we follow the guidelines introduced in [74] during the training process.

**Experiment I** We first train our proposed algorithm using a training data set containing attack instances from classes A1∼A9 and A14 (see Sec. 4.2.5). Table 4.4 summarizes the detection performance results when a testing data set with attack instances from the same classes as used in the training process is presented to our algorithm. We can observe that our algorithm is able to accurately identify instances of all the attack classes seen in the training process. The overall detection accuracy is 99.38%. In fact, only one detection window with no attack instance (out of 2200 total "normal operation" windows) is incorrectly labeled as an attack of type A3. We also observe that 7 instances (out of a total of 200 testing instances) from classes A1 and A2 are misclassified as "normal operation". Very few instances of classes A6∼A9 and A14 are incorrectly labeled as attacks from other classes, but they are never classified as "normal operation". As observed in real-world attacks (e.g., [2,142]), primitive attacks of types A1 and/or A2 usually need to be launched multiple times, so our detection algorithm can still help SCADA system operator detect such attacks. These results show that the number of false

Table 4.4: Detection Performance of Experiment I.

| Class | N | A1 | A2 | A3 | A4 | A5 |
|---|---|---|---|---|---|---|
| Precision | 99.8% | 100% | 100% | 99% | 100% | 100% |
| Recall | 99.9% | 96% | 97% | 100% | 100% | 100% |
| Class | A6 | A7 | A8 | A9 | A14 | |
| Precision | 94% | 94.2% | 96.8% | 100% | 94.9% | |
| Recall | 94.9% | 97% | 93.9% | 94% | 100% | |

positive instances generated by our tool is extremely low and only a few attack instances from classes A1 and A2 are misclassified as "normal operation". Therefore, our tool is well-suited for network intrusion detection in SCADA systems because most detection windows requiring the attention of the operator indeed contain primitive attacks.

**Experiment II** Next, we add previously unseen attack instances into the testing set of Experiment I and re-execute the detection (i.e., testing) process. When the output of the classification stage and the threshold value $R_{th}$ indicate that a detection window contains previously unseen network behavior patterns, we temporarily label it as an "unknown" class instance and record the features learned by the CNN stage. Once a sufficiently large number of "unknown" class instances are collected, we perform k-means clustering to find distinct clusters contained in the "unknown" class. Our tool then requests the SCADA system operator to inspect instances from different clusters and provide them with proper labels. Once the number of instances included in a cluster is sufficiently large, we pump these instances into the existing training set, add a new attack class in the classification stage, and re-train the entire neural network.

Table 4.5 summarizes the detection results obtained after our algorithm is re-trained to include classes A10∼A13 and A15∼A18 in the classification stage. The overall detection accuracy is 99.84%, which is sufficiently high for intrusion detection in real-world SCADA systems. Note that a sufficiently large number of instances of a newly emerged attack class must first be collected to create a representative training set. In our experiment, we require that 100 instances for each new attack type must be labeled before being added into the existing training set for re-training because our experience suggests that adding fewer instances into the

existing training set will result in significant deterioration of detection accuracy (e.g., previously unseen attacks may be incorrectly labeled as instances of other classes). Misclassifying previously unseen attacks may hinder SCADA system operator from becoming aware of newly emerged threats quickly and mislead him/her into taking ineffective countermeasures.

As shown in Table 4.5, our algorithm still achieves high accuracy with very few false positives once sufficient previously unseen attack instances are pumped into the training set. Although a few instances from classes A6∼A18 are misclassified as other attack classes, they are never classified as "normal operation". These results suggest that our re-training scheme facilitates rapid adaptation of our neural network models to SCADA environments that are subjected to previously unseen primitive attacks. These properties, coupled with the fact that our approach identifies primitive attacks without interrupting SCADA system operation, make our approach suitable for network intrusion detection in SCADA systems.

Table 4.5: Detection Results for Testing Phase in Experiment II.

| Actual Class | N | A1 | A2 | A3 | A4 | A5 | A6 |
|---|---|---|---|---|---|---|---|
| Total Instance | 3550 | 100 | 100 | 100 | 100 | 100 | 100 |
| Detected | 3549 | 96 | 97 | 100 | 100 | 100 | 94 |
| Misclassified (Class \| Count) | A3 \| 1 | N \| 4 | N \| 3 | | | | A14 \| 5 |
| | | | | | | | A15 \| 1 |

| Actual Class | A7 | A8 | A9 | A10 | A11 | A12 |
|---|---|---|---|---|---|---|
| Total Instance | 100 | 100 | 100 | 100 | 100 | 100 |
| Detected | 97 | 92 | 94 | 97 | 96 | 98 |
| Misclassified (Class \| Count) | A8 \| 3 | A7 \| 6 | A6 \| 6 | A11 \| 3 | A10 \| 1 | A11 \| 2 |
| | | A13 \| 2 | | | A12 \| 3 | |

| Actual Class | A13 | A14 | A15 | A16 | A17 | A18 |
|---|---|---|---|---|---|---|
| Total Instance | 100 | 100 | 100 | 100 | 100 | 100 |
| Detected | 97 | 93 | 97 | 94 | 97 | 98 |
| Misclassified (Class \| Count) | A8 \| 3 | A12 \| 3 | A12 \| 1 | A12 \| 1 | A6 \| 2 | A15 \| 1 |
| | | A15 \| 4 | A14 \| 2 | A14 \| 2 | A8 \| 1 | A16 \| 1 |
| | | | | A15 \| 3 | | |

## 4.3 Conclusion

In this work, we first design an unsupervised algorithm for P2P botnet detection in SCADA systems by capturing the flow-based and connectivity-based characteristics of C&C communication patterns of bots. Our method has a low deployment barrier because it leverages traffic monitoring capabilities that have already been built into existing networking devices. High detection accuracy is achieved in our evaluation, which indicates that the proposed approach is well-suited for securing SCADA systems. Since our designed system generates only a few false positives and does not interrupt normal system operation, it is well-suited for P2P botnet detection in SCADA systems.

In addition, we also design and implement a network intrusion detection solution for SCADA systems based on deep neural networks. CNN-based feature learning is particularly useful in developing IDS solution for SCADA systems when temporal network behavior patterns of SCADA hosts need to be reliably modeled. Achieving satisfactory detection performance and permitting adaptations with network traces of previously unseen attacks, our proposed approach is well-suited for network intrusion detection in SCADA networks where both conventional and specialized attacks may be present.

# Chapter 5

# Payload Attack Detection for Programmable Logic Controllers (PLCs)

## 5.1 Introduction

In industrial control systems (ICS), programmable logic controllers (PLC) play a critical role in process automation. As cyber attacks targeting ICS increase in sophistication, field devices, such as PLCs, are of particular concerns because they directly monitor and control physical processes. As shown in Figure 5.1, PLCs are typically deployed close to sensors and actuators, implementing local control actions (i.e., regulatory control). In addition of utilizing sensor data and controlling actuators locally, PLCs transmit real-time process data to operator workstations and execute their commands, facilitating the realization of supervisory control. Due to the unique and vital role of PLCs in critical ICS infrastructure [5], they are one of the major targets of cyber attacks. For example, the Stuxnet attack managed to silently sabotage centrifuges in a uranium-enrichment plant by reading and writing code blocks on PLCs from a compromised engineering workstation [38, 71]. By modifying a PLC's control program, severe damages (e.g., data loss, interruption of system operation, and destruction of ICS equipment) can be induced by attackers. In [47], it is shown that malicious code can easily be slipped

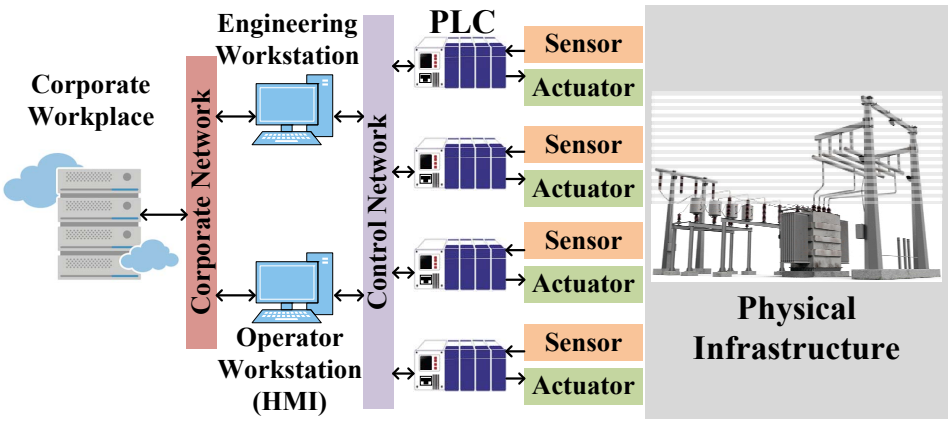



Figure 5.1: Architecture of industrial control systems and the role of PLCs.

into PLC control programs and evade the scrutiny of relay engineers from both academia and industry. Therefore, it is crucial to devise automated detection method against cyber attacks launched by modified PLC's control program.

As PLCs are special-purpose computers interfacing with various sensors/actuators and providing firmware support to run control programs (also known as "payload" programs [41, 88]) that emulate the behaviors of an electric ladder diagram [5, 110], attacks on PLCs can be launched by modifying or overwriting the PLC payload program. Such attacks are known as *PLC payload attacks*. A PLC control program is typically written by a team of PLC engineers using the suite of programming languages specified in IEC 61131-3 [52]. Such a control program is regarded as the payload of a PLC's firmware, which controls access to hardware resources (e.g., inputs, outputs, and timers) and repeatedly loops through the payload instructions. An attacker with PLC access (e.g., by gaining control of an engineering workstation running PLC development and monitoring software) can download malicious payload and gain full control over its sensors and actuators. In the Stuxnet attack, a component of Stuxnet is capable of launching payload attacks on PLCs by first infecting an engineering workstation and then downloading malicious code blocks [38]. Payload attacks can also be carried out by an insider (e.g., a disgruntled employee) with the help of tools such as SABOT [88], which generates malicious payload based on adversary-provided specifications. Since legitimate payload relies on PLC programming instructions implemented by the firmware to carry out control and monitoring tasks, a malicious payload program can execute any combination of these instructions

to sabotage the physical process.

In this chapter, we introduce runtime behavior monitoring into PLC firmware to detect payload attacks and protect ICS from severe physical damages. Based on control system specification provided by control system engineers, we establish runtime behavior profile of normal/legitimate payload program in terms of I/O access patterns, network access patterns, as well as payload program timing characteristics. When a newly updated payload program is downloaded into a PLC (either by an attacker or by a trusted control system engineer), its runtime behavior data is collected by the PLC firmware. When abnormal behaviors are observed by the firmware, execution of the payload program is terminated so that abnormal control signals will not be sent to actuators. The contributions of our work are as follows:

- We introduce runtime behavior monitoring into PLC firmware to enable automated detection of PLC payload attacks. In contrast to existing detection methods based on linear temporal logic, our proposed approach can identify attacks that violate real-time requirements of an ICS and does not require the introduction of bump-in-the-wire apparatus between engineering workstation and PLCs.
- We present a proof-of-concept implementation of the firmware-level payload attack detection scheme on ARM® Cortex®-M4F microcontrollers. Our evaluation results show that the proposed approach can detect a wide variety of payload attacks revealed by prior research [47] and reported cyber-security incidents.
- Furthermore, we evaluate the overhead of implementing the proposed detection method and find that it is feasible to incorporate our scheme on microcontrollers used by existing PLCs to detect payload attacks.

## 5.2 Related Work

### 5.2.1 Programmable Logic Controller (PLC) and Payload Program Execution Model

A programmable logic controller (PLC) is a special-purpose computer designed to replace relay panels and control a physical process [110]. Figure 5.2 presents the general hardware

and software architecture of PLCs. There are several important characteristics that distinguish PLCs from personal computers [101]: PLCs are designed to operate in harsh industrial environments and are programmed in relay ladder logic or other PLC programming languages [52]. In addition, a PLC executes a simple payload program in a sequential fashion. Once deployed in an ICS, a PLC continuously collects readings from sensors connected to its inputs, runs the PLC payload program, and generates outputs that control the physical process. As shown in Fig. 5.1, PLC control program can be developed on engineering workstations using programming software that supports ladder logic or other PLC programming languages and downloaded to target PLC for execution. Operator of an ICS may monitor and control the physical process via a human-machine interface (HMI), which communicates with PLCs to receive real-time process data and issue control commands.

To control and monitor physical process, a PLC's firmware implements input and output image tables as well as a program scan cycle [101,110]. A program scan cycle consists of input scan, program scan, output scan, and housekeeping phases, which are shown in Fig. 5.3. After system start-up, a PLC repeatedly walks through the four phases of the program scan cycle as follows: First, in the input scan phase, the PLC firmware samples the I/O pin values and writes them into the input image table. Then, in the program scan phase, instructions in the payload program are executed one by one using values stored in the input image table. Output values are generated during this phase and written into the output image table. Next, in the output scan phase, values in the output image table are transferred to the external output terminals, making control actions specified in the payload program take effect. Finally, in the housekeeping phase, internal checks on memory and system operation are performed. Additionally, communication requests originated from other hosts (e.g., the HMI) or generated by the payload program itself are also serviced before the next program scan cycle starts.

### 5.2.2 PLC Ladder Logic

Many widely-used PLC programming languages are standardized in IEC 61131-3 [52] and ladder logic is the most commonly used one [101] since it is straightforward to control system engineers who prefer to define control actions in terms of relay contacts and coils. Instructions

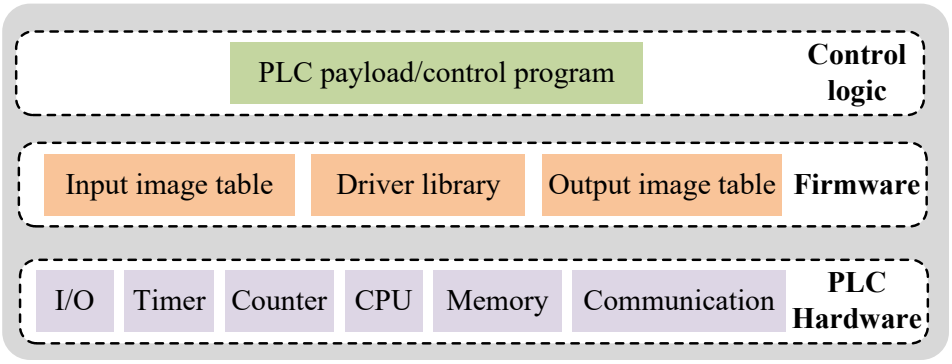



Figure 5.2: General PLC hardware and software architecture.

specified by ladder logic have their own symbolic representation. A PLC payload program written in ladder logic has one or more ladder-formatted schematic diagrams. Within each diagram, ladder logic instructions are organized into rungs. Each rung may contain multiple ladder logic instructions, which are evaluated from left to right. Instructions on the left of a rung test input conditions or outputs generated by other rungs, and instructions on the right generate rung outputs. Multiple input condition checks can be placed in tandem, and the input logic evaluates to true if and only if all input conditions are true. Parallel branches can be used on a rung to accommodate more than one input condition combinations. The rung logic is evaluated to true as long as one of the branches forms a true logic path. When multiple output branches are present on a rung, a true logic path controls multiple outputs.

Fig. 5.4 shows a sample subroutine of a ladder logic program consisting of three rungs. The "XIC" instruction on the first rung examines if an input is true. If so, the instruction evaluates to true. The "OTE" instruction energizes a specified output bit. Input condition

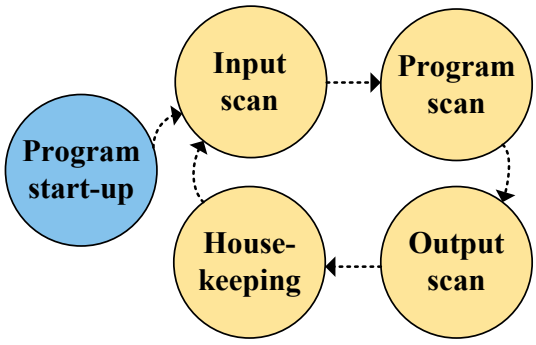



Figure 5.3: PLC payload program execution model.

of the first rung first checks if input bit I:0/4 or I:0/3 is true and then checks if I:0/0 bit is true. The output of this rung controls both output bits, i.e., O:2/1 and O:2/2. The second rung's input condition is always true, so the subroutine in file U:7 is executed. Note that the subroutine is essentially another ladder logic diagram. When the subroutine returns, the second rung completes and the third rung is evaluated, which signals the end of the payload program. Note that hierarchical addressing is used in ladder logic program to specify the data type, slot number, and bit position of PLC data and peripherals [101]. For example, I:0/4 is the fifth bit of binary input slot 0 (with the first bit being I:0/0). For analog I/Os, the hierarchical address is slightly different. For example, O:2.0 is an analog output on the output module installed on slot 2, and the output value is written to the first (zero-indexed) word of its allocated memory.

Ladder logic provides a wide range of instructions for PLC engineers to specify control actions. Bit instructions examine status of individual input/internal bit or control a single output bit. Word instructions, such as mathematical operations, data transfer, and logical operations, operate on data words or registers. Program control instructions, such as subroutine invocation and return, control the execution flow of the payload program. For control program of large and complex ICS, subroutines are frequently used to better organize the instructions and enhancement maintainability. In addition, communication instructions allow a PLC to communicate with other hosts via a particular ICS network protocol. From the perspective of PLC control program development, a malicious payload is essentially a combination of legitimate PLC programming instructions causing disastrous impacts on an ICS.

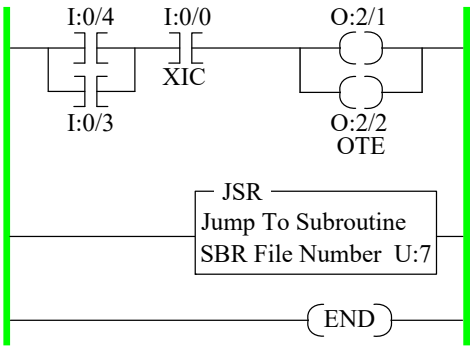



Figure 5.4: A sample ladder logic program with three rungs.

In this work, we focus on detecting payload attacks implemented via ladder logic, but the proposed techniques are applicable to attacks written in other languages [52] as well because different PLC programming languages can be used to implement the same control system specifications [101].

### 5.2.3 Firmware vs. Payload Attacks

As revealed by Fig. 5.2, both the PLC firmware and its payload program can become the target of cyber attacks. An attacker can reverse-engineer and modify the firmware on a PLC to launch firmware attacks. In this case, even though a legitimate payload program is downloaded to the PLC, its execution will still be monitored and/or intercepted by the modified firmware. In [4], a rootkit is developed on the CODESYS PLC runtime to intercept I/O operations of the payload program. When the payload wants to read or write a certain I/O pin, interrupt handler installed by the attacker is called first, within which the attacker can reconfigure the I/O pins or modify values to be read/written. In [41], a more advanced rootkit is developed for an Allen Bradley CompactLogix PLC firmware. In addition to intercepting PLC inputs and outputs at the firmware, it incorporates physical-process awareness and always presents modified sensor measurements, hoaxing ICS operator in front of the HMI to think that the system runs normally.

Firmware attacks typically requires detailed knowledge on target PLC's hardware components and reverse-engineering of its firmware because PLCs are closed-source embedded devices [32]. An attacker needs to install the rootkit on PLCs either via the built-in remote firmware update mechanism or by loading it via JTAG interface [41]. For firmware update process protected by cryptographic means (e.g., certificate in the X.509 standard), it is hard to install a modified version of the firmware on the PLC. Alternatively, an attacker can load modified PLC firmware via JTAG interface. However, such an approach will require physical access to the PLC and possibly disassembling it.

PLC payload attacks, on the other hand, are much easier to launch. An insider with proper privileges can easily download (e.g., a disgruntled control system engineer) a malicious payload program. As shown in Fig. 5.1, such an insider may download a malicious payload program

via the engineering workstation to one or multiple PLCs. Integrity checks on PLC payload program cannot effectively prevent such attackers from downloading malicious payload because warnings on payload program changes can always be overridden once proper privileges are acquired (e.g., a password allowing engineers to repeatedly download revised payload program for development and debugging purposes). Alternatively, sophisticated cyber attacks, such as Stuxnet [38, 71], may include payload attack as a component to induce physical damages on ICS. Partial knowledge on the physical process can be sufficient to create a malicious payload using automated tools such as SABOT [88]. In [47], a small-scale challenge shows that malicious code snippets are likely to evade the scrutiny of code reviewers. Therefore, it is necessary to develop automated payload attack detection mechanisms to protect physical infrastructure from PLC payload attacks.

### 5.2.4 Payload Attack Detection

As payload attacks can easily be launched by insiders or from compromised engineering workstations, several techniques that detect payload attacks have been proposed. In [84], a bump-in-the-wire device, called PLC guard, is introduced to intercept the communication between an engineering workstation and a PLC, allowing engineers to review the code and compare it against previous versions. Features of the PLC guard include various levels of graphical abstraction and summarization, which makes it easier to detect malicious code snippets. In [66], an external runtime monitoring device (e.g., a computer or an Arduino microcontroller board) sits alongside the PLC, monitors its runtime behaviors (e.g., inputs, outputs, timers, counters), and verifies them against ICS specifications converted from a trusted version of the PLC payload program and written in interval temporal logic. It is shown that functional properties of payload program can be verified against ICS specifications, but the types of payload attacks that can be detected by this approach remain to be explored.

In [89, 143], a trusted safety verifier is introduced as a bump-in-the-wire device that automatically analyzes payload program to be downloaded onto a PLC and verifies whether critical safety properties are met using linear temporal logic. However, linear temporal logic implicitly assumes that states of the systems are observed at the end of a set of time intervals.

In the case of PLC payload program, snapshot of system states is taken at the end of each program scan cycle. As a result, real-time properties that does not span multiple program scan cycles cannot be checked by the trusted safety verifier. For example, a legitimate payload program is required to energize its output immediately when a certain input pin is energized. An attacker can inject malicious code and prolong the program scan cycle to cause real-time property violation while evading code analytics based on linear temporal logic. In [108], the timer on-delay (TON) ladder logic instruction is modeled using linear temporal logic. The TON instruction starts a timer when its input condition evaluates to true and energizes its output (i.e., the "Done" bit) when the timer reaches the preset value. It is shown in [108] that TON behavior can be approximated with the combination of liveness and fairness properties: Either TON instruction is not used or TON output bit will eventually be energized. However, linear temporal logic cannot verify whether the TON output bit is energized at the exact program scan cycle designated by control system engineers. Therefore, such an approximation does not capture critical real-time requirements of ICS.

In this work, we introduce runtime behavior modeling and monitoring of PLC payload in PLC firmware. Our proposed approach complements existing detection techniques and can detect violations of ICS real-time properties. In addition, our proposed approach does not require the introduction of any external apparatus that may introduce new vulnerabilities into ICS.

### 5.2.5 Runtime Behavior Monitoring for Anomaly Detection

The idea of detecting abnormal program behaviors by monitoring its execution at runtime has been applied to an rich array of computer systems. Runtime behavior monitoring techniques on operating systems such as Windows, Linux, and Android are reviewed in [34, 136]. However, these techniques cannot be directly applied to PLCs since PLCs are closed-source systems [32] running specialized firmware and payload programs. System calls utilized by existing techniques are not available in PLC systems. In [82], a runtime anomaly detector hardware design is proposed for embedded systems, which eliminates performance overheads incurred by software-based runtime monitoring methods. In [36], a timing-based PLC program

anomaly detector is designed. An external data collector is deployed to collect program execution time measurements and detect unauthorized modifications to the PLC system. In [130], runtime behaviors are monitored via dedicated hardware performance counters, which are not widely available in microcontrollers utilized by PLCs. To detect payload attacks in existing ICS, runtime behavior monitoring technique must utilize only the resources available on microcontrollers used in existing PLCs and does not require external apparatus (e.g., data collector proposed in [36]).

## 5.3 System Overview

### 5.3.1 Adversary Model

A malicious payload may be directly downloaded by an insider with PLC programming privilege. For instance, the insider can be a PLC programmer responsible for deploying tested PLC payload program. However, he/she downloads a different payload, which may be written anew or modified from the tested version. Since such an attacker has proper privilege to program PLCs, integrity checks on PLC payload program can be overridden and will not prevent malicious payload from being downloaded. For an external attacker, security flaw of other

Table 5.1: Control System Specifications and Legitimate PLC Control Logic.

| Control System Specification | Legitimate Control System Logic |
|---|---|
| Digital I/O pins, values & functionality | Control logic of binary inputs and outputs |
| Analog I/O pins, value ranges, & functionality | Sensor output and actuator input ranges, control logic of analog I/Os |
| Legitimate sequences and timing relationships of I/O operations | Control logic of I/Os, possibly controlled by counters and timers |
| Network data packet and timing relationships | Data from network for local control tasks or data required by remote hosts (e.g., HMI or other networked PLCs), and real-time requirements for these network events |
| Network commands and timing relationships | Control tasks mandated by operator workstation and their real-time requirements |

ICS components may be exploited to gain access to an engineering workstation, which allows he/she to download malicious payload. For example, in the Stuxnet attack [71], many potential attack vectors, including the PLC programming environment, are exploited to eventually compromise a PLC-connected engineering workstation.

We assume that the attacker is not capable of changing the PLC firmware, which requires either attacking the cryptographically protected firmware image or loading modified firmware directly via JTAG interface. Therefore, firmware-level detection mechanism proposed in this work is not tampered by the attacker. The goal of a payload attack is not limited to blocking legitimate outputs, causing system interruption, and destruction of system equipment. Sophisticated attacks such as the PLC blaster worm [122] which replicates itself to other PLCs can also be launched. However, such attacks download a payload program that are significantly different from the legitimate version in terms of program size and functionality, which can be identified by human operator monitoring the control system. In this chapter, we consider stealthy payload attacks that are modified from legitimate payload programs. Such attacks preserve certain legitimate payload properties (e.g., always sending sensor readings requested by HMI) while carrying out malicious tasks.

### 5.3.2 PLC Program Development Process and Control System Specifications

To develop PLC payload program for an ICS, the following process is typically adopted by PLC engineers:

1. *Specification Formulation.* Control tasks to be carried out by a PLC are identified and input/output signals required by these tasks are defined. The logical sequence of operations for the PLC are specified, e.g., in the form of sequence table, flow chart, or relay schematic [101].

2. *PLC Program Development.* At this step, PLC program is developed based on the formulated specifications. Although an engineering team usually has its own set of guidelines and best practices on program organization and documentation, the generated PLC payload always aims to accurately implement the specifications. At this stage, an

attacker (e.g., a disgruntled control system engineer) may collect legitimate payload program and modify it to generate malicious payload.

3. *Testing.* Before deploying the PLC program, PLC engineers need to test the program via simulation or under some test environment. Safety properties (e.g., a circuit breaker must trip if a fault is detected) can be provided by system operators and/or identified during specification formulation. In addition, different combinations of input values are fed to the PLC to ensure that correct responses are taken under different system operation scenarios. Although the test cases may not be exhaustive (e.g., it is hard to implement all test cases when analog inputs are used), important system properties, such as safety and real-time requirements, should always be validated.

4. *Maintenance.* After an initial version of the PLC control program is deployed, the ICS may go through hardware upgrades and design improvements. Accordingly, the specifications should be updated and the PLC program should be revised. After necessary testing, the new payload is downloaded to the PLC.

In this work, we assume that *control system specifications*, such as the number of I/Os, functionality of each I/O pin, and possible ranges of I/O values, are available. Such specifications are usually provided by the control system engineering team that develops the legitimate payload program. Table 5.1 summarizes the control system specifications required by our detection mechanism and the corresponding legitimate control system actions. For instance, when designing the legitimate payload, a digital output pin may be used to control a circuit breaker to trip. The engineering team knows whether a "0" or a "1" corresponds to the "trip" signal, so it is straightforward to generate control system specifications describing the functionality of this output pin. To implement control operation sequences (e.g., tripping a circuit breaker and then re-closing it), timers and counters are generally used. When the legitimate payload program is created, timers and counter must be properly configured to control the temporal behaviors of the payload program. These configurations can then be converted into timing relationships among I/O and network events.

### 5.3.3 Payload Attack Detection at PLC Firmware

Using control system specifications, runtime behavior model of legitimate PLC payload program is established and stored in the PLC firmware. The timing relationships between inputs and outputs, the number of network packets generated after different control actions, as well as timing relationships between I/O and network events, are modeled. By modifying the PLC firmware, runtime behaviors of the payload program (e.g., I/O and network access patterns) are time-stamped and compared against the established runtime behavior model. In addition, a backup version of the output image table is separately stored by the firmware at the beginning of each program scan cycle. If a certain abnormal runtime behavior is detected, the backup output image table is loaded to overwrite the output generated by the payload. As a result, any output related to the detected abnormal runtime behavior will not affect the physical system. For PLC payload sending/receiving network packets, network requests are also blocked when a runtime behavior anomaly is detected by the firmware.

## 5.4 System Design

### 5.4.1 PLC Payload Runtime Behavior Model

Given the control system specifications, it is possible to create a *runtime behavior model* for legitimate PLC payload. Suppose that we need to create control system specifications for the PLC shown in Fig. 5.5. In this figure, sample specifications for I/O terminals and the network port is provided. We note that timing relationships are not shown in Fig. 5.5. The information categorized in Table 5.1 allows us to create the runtime behavior model as follows: First, the number of (analog and digital) I/Os and their feasible values are determined. For instance, for digital input I:0/0 in Fig. 5.5, its legitimate values are "1" and "0". For analog input I:1.0 (note that the notation for analog I/Os is different from that for digital I/Os as mentioned in Sec. 5.2.2), the legitimate value ranges are 0∼3V and 12∼15V. In the PLC firmware, such information can be stored as a table (see Fig. 5.6 for an example), with each row storing the legitimate values/ranges of a particular pin. We call this table the *I/O event table*.

Next, the number of network packets received or sent by the legitimate payload are ex-

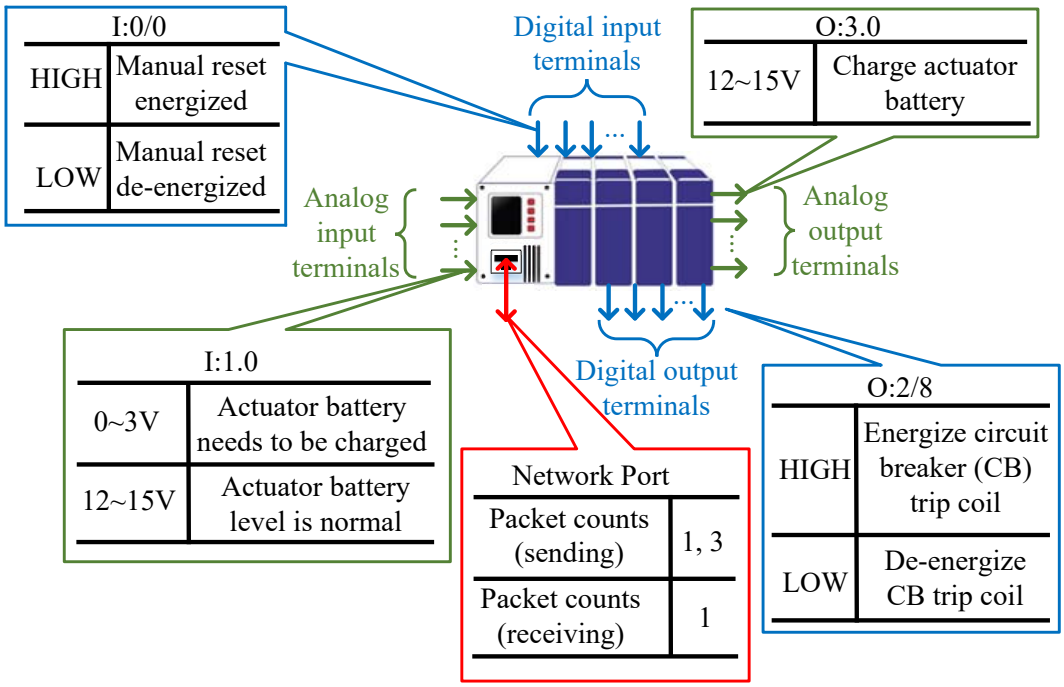



Figure 5.5: PLC wiring diagram with sample control system specifications for I/O and network events.

tracted from the specifications. Since PLC payload program is designed to control physical process, network packets are typically associated to specific I/O conditions. For instance, when an alarm signal is energized to sound a horn, the same alarm signal is usually transmitted via a network packet to the HMI at the same time. When a process data request from the HMI is received, the PLC generates process data response(s) to transmit the requested data. In the PLC firmware, network event information can be stored as a table with two rows (see Fig. 5.6 for an example). The first row lists the numbers of network packets that can be received, and the second row lists those that can be sent. We call this table the *network event table*. Using the I/O and network event tables, we are able to model the legitimate runtime behaviors of I/Os and network port(s) at any particular time instant.

Then, timing relationships between inputs, outputs, and network accesses are established. To store these relationships, a sparse matrix is created in the PLC firmware (see Fig. 5.6 for an example). We call this sparse matrix the *timing behavior matrix*. Both the rows and the columns of the matrix are indexed by legitimate I/O and network operations. For instance, the I:0/0:1 event in the matrix in Fig. 5.6 represents the I/O event where digital input pin I:0/0

is set to HIGH. Each column of the matrix represent a particular payload program action, whereas the rows with non-zero values represent its preconditions. For instance, the matrix in Fig. 5.6 indicates that there are four preconditions under which a network packet will be generated and sent by a legitimate PLC payload. Note that the non-zero value in the matrix represent the maximum time (in microseconds) within which a column event will occur.

Once all information provided in the control system specifications is converted into a runtime behavior model, three tables are stored into the PLC firmware (i.e., the I/O event table, the network event table, and the timing behavior matrix). These tables will only be updated if changes to the control system specifications are made (e.g., additions of new sensors/actuators). When a PLC payload is downloaded to a PLC, the PLC firmware assumes that its runtime behaviors match the ones specified in the supplied control system specifications. Any deviation from the encoded runtime behavior model will be regarded as an anomaly.

### 5.4.2 Payload Attack Detection at PLC Firmware

Our detection scheme introduces runtime behavior monitoring into the PLC firmware and compares the runtime behaviors of the currently deployed payload against the runtime behavior model established from control system specifications. To implement the proposed detection scheme, the following modifications to the PLC firmware are incorporated:

#### Logging Access to Input and Output Images

As introduced in Sec. 5.2.1, input image is updated before each run of the payload program, and output image is updated after each run. In existing PLC firmware, I/O reads move values from the input/output image to a designated memory location. When an output pin is written, value stored in a memory location is moved to the output image table. To receive/send a packet, receive/transmit queue is either explicitly (via ladder logic instruction) or implicitly (at the end of the housekeeping phase) queried. To monitor the I/O and network access patterns, we modify the implementation of PLC firmware to log the system time-stamp of these operations. This can be achieved by setting up the memory protection unit (MPU)

**I/O Event Table**

| I/O Event | Legitimate Values/Ranges |
|---|---|
| I:0/0 | 1 (HIGH), 0 (LOW) |
| ⋮ | ⋮ |
| I:1.0 | 0:3 (0~3V), 12:15 (12~15V) |
| ⋮ | ⋮ |
| O:2/8 | 1 (HIGH), 0 (LOW) |
| ⋮ | ⋮ |
| O:3.0 | 12:15 (12~15V) |
| ⋮ | ⋮ |

**Network Event Table**

| Network Event | Legitimate Packet Counts |
|---|---|
| receiving | 1 |
| sending | 1, 3 |

**Timing Behavior Matrix**

| | I:0/0:1 | I:0/0:0 | ⋮ | I:1.0:0:3 | I:1.0:12:15 | ⋮ | O:2/8:1 | O:2/8:0 | ⋮ | O:3.0:12:15 | ⋮ | Recv:1 | Send:1 | Send:3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I:0/0:1 | | | | | | | | | | | | | 2000 | |
| I:0/0:0 | | | | | | | | | | | | | | |
| ⋮ | | | | | | | | | | | | | | |
| I:1.0:0:3 | | | | 300 | | | | | | | | | | |
| I:1.0:12:15 | | | | | | | | | | | | | | |
| ⋮ | | | | | | | | | | | | | | |
| O:2/8:1 | | | | | | | | | | | | | 500 | |
| O:2/8:0 | | | | | | | | | | | | | 500 | |
| ⋮ | | | | | | | | | | | | | | |
| O:3.0:12:15 | | | | | | | | | | | | | | |
| ⋮ | | | | | | | | | | | | | | |
| Recv:1 | | | | | | | | | | | | | 1500 | 1500 |
| Send:1 | | | | | | | | | | | | | | |
| Send:3 | | | | | | | | | | | | | | |

Figure 5.6: A sample runtime behavior model established based on control system specifications in Fig.5.5, consisting of two tables and a sparse matrix.

to enter interrupt when the user program accesses the input/output images or the network queues. In existing PLC firmware, a separate system timer is typically supported. This timer provides the time-stamps for the I/O and network events to be monitored. If I/O images are accessed, the interrupt handler decodes the I/O pin address and log the time-stamp of the operation. Suppose that the same input pin is accessed multiple times during a single program scan cycle, only the time-stamp of the first read operation is logged. For an output pin, both the first read and the last write operations are time-stamped. For access to network queues, the number of packets received/sent is logged and time-stamped. Time-stamps of I/O and network operations are stored in a separate table (known as the *runtime time-stamp table*) in the PLC firmware. Each entry of the table corresponds to a particular I/O event (e.g., a legitimate I/O value is observed) or network event (e.g., a legitimate number of packets are sent).

In our current implementation, the maximum number of time-stamps logged by the runtime time-stamp table is 10 for each I/O event. If more than 10 time-stamps are collected, newly generated time-stamps will be discarded. We log the time-stamp for the first I/O read operation and last output operation within each program scan cycle because control system specifications typically use the observation of an I/O value on the physical process as precon-

dition. Take the output pin O:2/8 in Fig. 5.5 as an example. Even if the payload program operates on O:2/8 multiple times during a program scan cycle, it is the last value written into the output image that will actually take effect. For each legitimate network event, our current implementation logs a maximum of 20 time-stamps. Newly collected time-stamps will be discarded if there are already 20 time-stamps pending in the table.

### Validating Runtime Behaviors

When time-stamping I/O and network events, any event that is not included in the I/O and network event tables is regarded as an abnormal runtime event. In addition, a separate sparse matrix (known as the *runtime sparse matrix*) is created and maintained in the PLC firmware to keep track of the timing relationships at runtime. The sparse matrix is also updated in the MPU interrupt handler. Runtime behaviors specified in the timing behavior matrix are validated in the output scan phase before the values in the output image are transferred to external output terminals. If any of the preconditions specified by the runtime behavior model are met, the timing relationships are checked. If an event occurs but none of its preconditions are active, a runtime behavior anomaly is detected. Take the timing behavior matrix in Fig. 5.6 as an example. Suppose that during a program scan cycle, we observe two occurrences of the event "Send:1". For the first time-stamp of "Send:1", we check the all the time-stamps for its preconditions. If any of the timing relationships is met, the corresponding entry in the runtime sparse matrix is cleared. In the runtime time-stamp table, the oldest time-stamp for the corresponding precondition event is removed. If a violation of the timing relationship is detected, a runtime behavior anomaly is found and the execution of the payload program should be terminated. Then, for the second time-stamp of "Send:1", previously cleared precondition fields are set if the corresponding entries in runtime time-stamp table have pending time-stamps. The timing relationships for "Send:1" are then validated again.

### Backing Up the Output Image

At the beginning of each program scan cycle (i.e., in the input scan phase), a backup version of the output image table is separately stored by the PLC firmware. Values in this backup image

are simply the output of the preceding program scan cycle. If runtime behavior anomaly is detected at the current program scan cycle, the backup image is used to overwrite the output image generated by the payload program. In this way, output values corresponding to illegitimate payload program behaviors are blocked.

### Canceling Network Send/Receive Requests

There are two scenarios where network send/receive requests generated by ladder logic instructions are processed: Network send/receive requests generated by a payload program are always processed in the housekeeping phase. To block these packets, we modify the firmware so that all pending network requests are cleared in the output scan phase if runtime behavior anomaly is detected. Alternatively, a subset of network-related ladder logic instructions can request the PLC firmware to service pending network tasks immediately. To prevent such network access, the implementation of MPU interrupt handler is further modified to check the preconditions of requested network operations. Suppose that a network-related ladder logic instruction is executed, after the network requests are generated (e.g., four packets will be retrieved from the receive queue), the firmware first enters the MPU interrupt handler and checks the preconditions of the requested network event. If any of the preconditions is met yet the corresponding timing relationship is violated, the network requests will not be executed because a runtime behavior anomaly is detected.

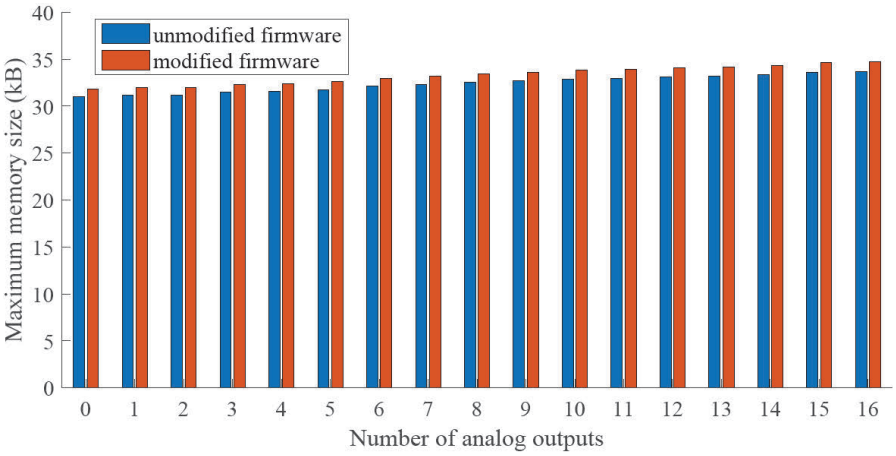



Figure 5.7: Maximum memory utilization of unmodified and modified PLC firmware running PLC payload programs with different numbers of utilized analog outputs.

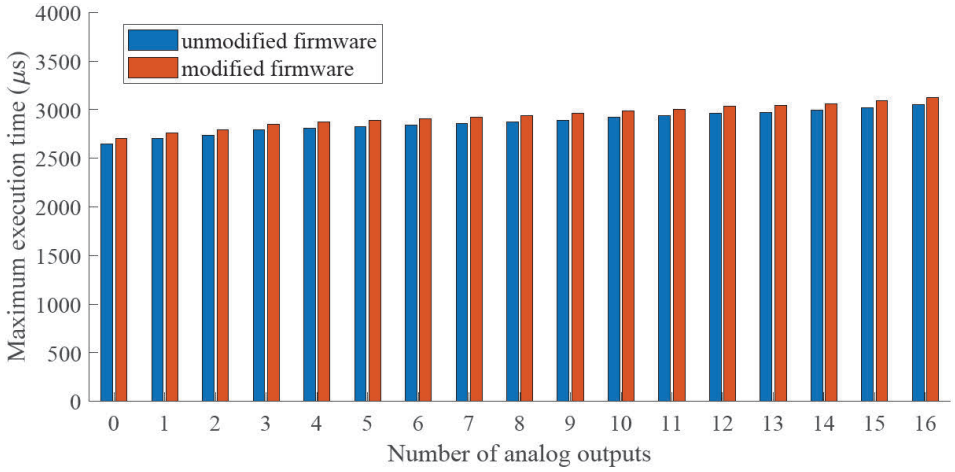



Figure 5.8: Maximum execution time of PLC programs with different numbers of utilized analog outputs.

It should be noted that our proposed detection scheme can easily be customized to notify ICS operators of the detection of PLC payload attacks. Suppose an on-site operator is to be notified, an extra output pin can be energized to set up an alarm during the output scan phase when runtime behaviors are examined. It is also possible to send out an alarm message to a remote HMI during this phase after the runtime behavior validation is done.

## 5.5 Evaluation

We implement the proposed payload attack detection method on Texas Instruments TM4C12x ARM® Cortex®-M4F core-based microcontrollers. Payload attacks are written in ladder logic, which are converted into machine code and loaded onto the PLC prototype. Hardware resources of the chosen microcontroller series are the currently active equivalents to the microcontrollers used by existing PLCs [41]. Memory protection unit (MPU) and system timer are available to implement our proposed detection scheme. Runtime behavior data collected by the PLC firmware is read from an Universal Asynchronous Receiver/Transmitter (UART) module connected to a PC. We first evaluate the overhead of implementing the proposed detection mechanism and then its detection performance.

### 5.5.1 Memory Overhead

Memory overhead of implementing the proposed detection method comes from both the firmware and payload levels. In the PLC firmware, runtime behavior model converted from control system specifications needs to be stored. Extra tables and sparse matrix are required to timestamp and keep track of the runtime behaviors of the currently deployed payload. The sizes of these matrices and tables will grow as the number of I/O and network events specified in the control system specifications grows. In addition, interrupt handler for the MPU as well as initialization code for the system timer and MPU need to be added to the PLC firmware. In our prototype, these firmware modifications translate to about 200 lines of assembly code (compared to the unmodified PLC firmware with about 6000 lines of assembly code).

To evaluate whether the memory overhead of our proposed detection mechanism is acceptable, we create payload programs utilizing different numbers of I/Os and generating different numbers of network packets. Note that each of these payload programs generates two types of network events (i.e., sending two packets or receiving one packet within each program scan cycle) and utilizes 16 digital I/Os. The number of analog outputs utilized by these payload programs varies from 0 to 16. Each analog output has two legitimate value ranges. The timing relationships in the control system specifications all describe preconditions for analog outputs. These payload programs are then loaded onto our PLC prototype twice: First, unmodified PLC firmware is used to execute the payload programs and the maximum sizes of the PLC firmware in the RAM are logged. Then, PLC firmware with our payload detection mechanism is used and the maximum firmware sizes are also recorded. Fig. 5.7 shows the memory overhead of implementing our PLC payload attack detection method in our PLC prototype. For a PLC system with 16 analog outputs, the memory overhead (compared to unmodified PLC firmware) is about 1 kB, which translates to a 3% increase in memory size. This memory overhead is acceptable for existing PLC systems on the market, which typically have more than 32 kB of memory [101].

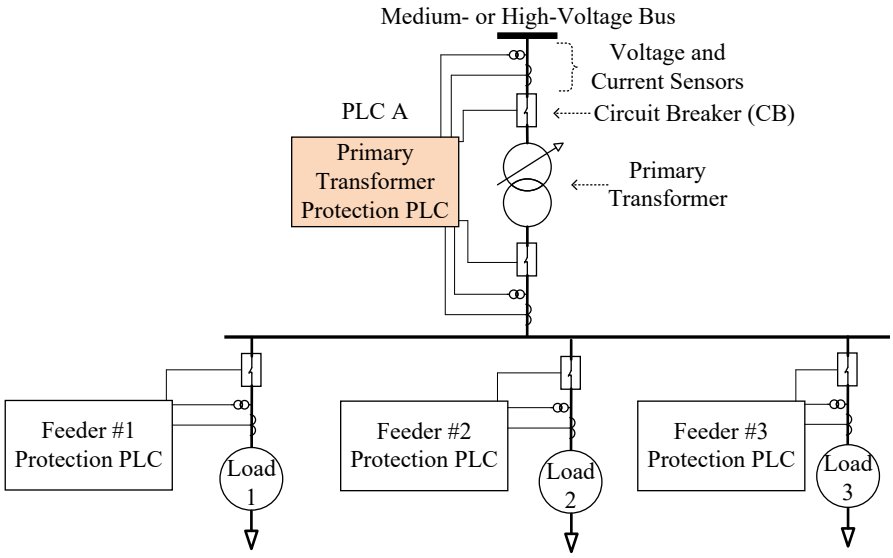



Figure 5.9: Sample power substation protection system implemented by multiple PLCs.

### 5.5.2 Execution Time Overhead

PLC payload program needs to satisfy execution time requirements in order to control physical process correctly. If a program scan cycle takes too long to complete, the PLC will not be able to track the changes of the physical process and generate control outputs timely. Since our payload detection mechanism incorporates runtime behavior monitoring and validations

Table 5.2: Attack Instances Implemented on PLC Prototype

| Attack Instance Group | Description |
|---|---|
| Illegitimate analog inputs (Group 1, 5 instances) | Scaling factors of analog input modules are modified by attacker to generate out-of-range input values. |
| Illegitimate network events (Group 2, 5 instances) | When trip coils are energized, the attack payload sends process data to multiple pre-specified destinations. When process data request is received, a packet containing intentionally modified process data is sent. |
| Illegitimate I/O event timing (Group 3, 5 instances) | Trip coils are not energized within 1000 $\mu$s when a voltage/current fault is detected. |
| Illegitimate network event timing (Group 4, 5 instances) | Packet containing up-to-date process data is not sent within 500 $\mu$s after process data request is received. |

in the PLC firmware, it is necessary the ensure that execution time of the program scan cycle does not significantly increase.

To evaluate the execution time overhead of the proposed detection mechanism, we measure the execution time of the payload program instances created in Sec. 5.5.1. Each payload program are executed for 1,000 program scan cycles on both unmodified and modified PLC firmware. Note that we added six extra assembly instructions in the PLC firmware to set up an extra output pin of the prototype PLC: At the beginning of each program scan cycle, this pin is set to HIGH. At the end of each program scan cycle, this pin is set to LOW. Fig. 5.8 shows the maximum execution time of the payload program instances. The average increase in maximum execution time is about 65 $\mu$s, which is far above the typical execution time of PLC payload programs (e.g., 1$\sim$10 ms [101]).

### 5.5.3 Detection Performance

To evaluate the detection performance of our proposed method, our PLC prototype emulates PLC A shown in Fig. 5.9. To implement the protection tasks assumed by PLC A, four analog inputs and two digital outputs are utilized. Our control system specifications require that both circuit breakers are tripped within 1000 $\mu$s once a voltage/current fault is detected on either side of the transformer. In addition, when process data request (sent by a PC emulating an HMI) is received, a packet containing up-to-date current and voltage readings must be sent within 500 $\mu$s. We create 20 different payload attack instances, which can be categorized into the four groups and are described in Table 5.2. Each payload attack instance is executed for 10 times (each run consisting of 1,000 program scan cycles). Table 5.3 shows the detection results when running the payload attacks on modified PLC firmware. 19 out of the 20 payload attack instances can always be detected during our evaluation, which shows that our proposed detection mechanism can help prevent PLC payload attacks without introducing external apparatus.

One of the attack instances (Group 2, Instance 2) cannot always be detected. This attack instance either generates illegitimate outputs or transmits modified process data as network packets. When it sends network packets, it simply modifies the process data values stored in

Table 5.3: Attack Instances and Detection Results

| Group/ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1/1 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 1/2 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 1/3 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 1/4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 1/5 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2/1 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2/2 | ✓ | ✓ | × | × | ✓ | ✓ | × | ✓ | × | ✓ |
| 2/3 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2/4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2/5 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 3/1 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 3/2 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 3/3 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 3/4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 3/5 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 4/1 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 4/2 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 4/3 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 4/4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 4/5 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

memory before they are encapsulated. The preconditions of this network events are still met and the timing relationships are not violated. Although this attack instance can sometimes evade our detection, it can be easily identified by existing detection methods against false data injection attacks [35].

## 5.6 Conclusion

In this chapter, we propose the detection of PLC payload attacks via runtime behavior monitoring in PLC firmware. Through modeling and monitoring the runtime behaviors, our proposed firmware enhancements can detect abnormal runtime behaviors of malicious payload. Using our proof-of-concept PLC prototype, we show that the proposed approach can identify a wide variety of PLC payload attacks revealed by prior research. In addition, our evaluations show that the execution time and memory overhead of the proposed detection

mechanism are acceptable for existing PLC firmware. Our proposed approach complements existing bump-in-the-wire solutions in that it can detect payload attacks that violate real-time requirements of ICS operations.

# Chapter 6

# Conclusion and Future Work

## 6.1 Summary

In this dissertation, we study two important aspects of smart grid infrastructure, i.e., worst-case delay performance and cybersecurity, and extend the application of some of our results to networked cyber-physical systems. In power substation automation systems (SASs) based on IEC 61850, conventional hardwired process connections are being replaced by switched Ethernet. To ensure system reliability and responsiveness, transmission of critical information required by protection and control tasks must satisfy hard delay constraints at all times. Therefore, delay performance conformance should be taken into consideration during the design phase of an SAS project. In addition, Ethernet-based network infrastructure is becoming increasingly popular in many industrial control systems. Deterministic delay performance must also be analyzed for these systems, which typically have non-feedforward traffic patterns and applications with different criticality levels.

Recently, wireless communication technologies, such as Wireless Local Area Networks (WLANs), have gained increasing popularity in industrial control systems (ICSs) due to their low cost and ease of deployment, but communication delays associated with these technologies make it unsuitable for critical real-time and safety applications. To address concerns on network-induced delays of wireless communication technologies and bring their advantages into modern ICSs, wireless network infrastructure based on the Parallel Redundancy Proto-

col (PRP) has been proposed. Although application-specific simulations and measurements have been conducted to show that wireless network infrastructure based on PRP can be a viable solution for critical applications with stringent delay performance constraints, little has been done to devise an analytical framework facilitating the adoption of wireless PRP infrastructure in miscellaneous ICSs. In addition, traffic patterns on wireless PRP network are non-feedforward, making existing analytical approaches based on network calculus inapplicable. To facilitate design and optimization of wireless PRP infrastructure with time-critical services, it is of vital necessity to devise an analytical method to find flow-specific worse-case network-induced delays.

Supervisory Control and Data Acquisition (SCADA) systems monitor and control critical infrastructure such as the smart grid. As SCADA systems become increasingly interconnected and adopt more and more cyber-enabled components, the risks of cyber attacks become a major concern. Due to their decentralized organization, peer-to-peer (P2P) botnets are resilient to many existing takedown measures and can be exploited as an effective way to launch cyber attacks on SCADA systems. However, little work has been done to detect P2P botnets in SCADA systems, which carry traffic flows with characteristics significantly different from the Internet. In addition, network-based cyber attacks on field devices are also one of the major cyber threats to ICS network infrastructure. Field devices and computing nodes in ICSs are subjected to both conventional network attacks and specialized attacks purposely crafted for SCADA network protocols. Proper methods and algorithms are needed to protect SCADA networks from emerging threats such as P2P botnets as well as attacks on SCADA protocols.

Besides network attacks, attacks on programmable logic controller (PLC), which is a critical component of industrial control system (ICS), also need to be properly addressed. Providing hardware peripherals and firmware support for control program (i.e., a PLC's "payload") written in languages such as ladder logic, PLCs directly receive sensor readings and control ICS physical process. An attacker with access to PLC development software (e.g., by compromising an engineering workstation) can modify the payload program and cause severe physical damages to the ICS. The ultimate purpose of many reported cyber attacks on SCADA systems is to gain access to PLCs and induce physical damages to the system process. Therefore, it is

of critical importance to protect field devices such as PLCs from payload attacks.

Addressing concerns on worst-case delay performance and cybersecurity of smart grid infrastructure, the contributions of this dissertation can be summarized as follows:

- *A worst-case delay performance framework for Ethernet-based network infrastructure with non-feedforward traffic patterns.* In Chapter 2, we study the worst-case delay performance of IEC 61850-9-2 process bus networks, which generally carry non-feedforward traffic patterns, through the combination of measurements and network-calculus-based analysis. Further refining our approach designed for Ethernet-based network infrastructure, we also apply it to other networked cyber-physical systems (NCPSs) and show that our method is able to derive useful bounds for NCPS applications with hard-real-time requirements.

- *A method of deriving closed-form expressions of worst-case delays for wireless PRP networks.* Leveraging the deterministic network calculus (DNC) theory, we propose in Chapter 3 to analytically derive worst-case bounds on network-induced delays for critical ICS applications. We show that the problem of worst-case delay bounding for a wireless PRP network can be solved by performing network-calculus-based analysis on its non-feedforward traffic pattern. Closed-form expressions of worst-case delays are derived, which has not been found previously and allows ICS architects/designers to compute worst-case delay bounds for ICS tasks in their respective application domains of interest. Our analytical results not only provide insights into the impacts of network-induced delays on latency-critical tasks but also allow ICS architects/operators to assess whether proper wireless RPR network infrastructure can be adopted into their systems.

- *Intrusion and botnet detection for SCADA networks.* In Chapter 4, we design a P2P-botnet detection method for SCADA systems, leveraging built-in traffic monitoring capabilities of SCADA networking devices. We propose to use unsupervised learning for P2P-botnet identification, which not only identifies known P2P botnets but also captures newly emerged ones. In addition, we propose a deep-learning-based network intrusion detection system for SCADA networks to protect ICSs from both conventional

and SCADA specific network-based attacks. Instead of relying on hand-crafted features for individual network packets or flows, our proposed approach employs a convolutional neural network (CNN) to characterize salient temporal patterns of SCADA traffic and identify time windows where network attacks are present. We incorporate a re-training scheme to handle previously unseen network attack instances, enabling SCADA system operators to extend our neural network models with site-specific network attack traces. Our results using realistic SCADA traffic data sets collected from energy-delivery system test beds show that the proposed deep-learning-based approach is well-suited for network intrusion detection in SCADA systems, achieving high detection accuracy and providing the capability to handle newly emerged threats.

- *Detecting payload attacks on PLC control programs.* To protect critical ICS infrastructure, we propose to model runtime behaviors of legitimate PLC payload program and use runtime behavior monitoring in PLC firmware to detect payload attacks in Chapter 5. By monitoring the I/O access patterns, network access patterns, as well as payload program timing characteristics, our proposed firmware-level detection mechanism can detect abnormal runtime behaviors of malicious PLC payload. Using our proof-of-concept implementation, we evaluate the memory and execution time overhead of implementing our proposed method and find that it is feasible to incorporate our method into existing PLC firmware. In addition, our evaluation results show that a wide variety of payload attacks can be effectively detected by our proposed approach. The proposed firmware-level payload attack detection scheme complements existing bump-in-the-wire solutions (e.g., external temporal-logic-based model checker) in that it can detect payload attacks that violate real-time requirements of ICS operations and does not require any additional apparatus.

## 6.2 Future Work

Built upon this dissertation, multiple research problems can be future explored. This section summarizes a few research directions that can be pursued in the future.

### 6.2.1 Time-varying deterministic network calculus (DNC) for delay performance analysis

So far, two major limitations of state-of-the-art methods of applying deterministic network calculus have been observed. On the one hand, existing methods are designed for tandem or feedforward networks, whereas practical applications may impose non-feedforward traffic patterns. This limitation is addressed by Chapters 2 and 3. On the other hand, the envelope (or bounding) models introduced by existing approaches are time-invariant. Although the functional forms of these models can be flexibly chosen (e.g., piecewise linear arrival and service curves become popular in recently proposed approaches such as [13, 120]). As time-invariant models have limited representative power, network flows with fast-changing traffic profiles cannot be properly modeled. In addition, networking nodes in wireless networks often offer time-varying service curves, which cannot be accurately modeled by the state-of-the-art piecewise-linear rate-latency service curve models. To address this limitation, we expect to introduce time-variant models (e.g., [28]) into our analytical framework, both enhancing the expressiveness of the models and facilitating further utilization of information included in measurements.

### 6.2.2 Extending existing time-invariant network calculus for other cyber-physical systems (CPSs)

Network-calculus-base delay performance analysis has been applied to many CPS applications. As one of our future directions, we would like to explore the application of network-calculus-based analysis to problems in systems other than the smart grid. For instance, as safety-critical applications start to leverage intra-vehicular networks [127], worst-case delay performance of such networks needs to be rigorously analyzed. Existing methods (e.g., [103]) only provide case-specific analyses, whereas algorithms giving sufficiently tight bounds under non-feedforward traffic patterns are yet to be validated. As another example, resource contention among multiple tasks served by a system-on-chip (SoC) or network-on-chip (NoC) system can also be modeled by network calculus [49]. Existing work (e.g., [46, 128]) introduces network-calculus models to analyze different resource contention scenarios, but a general framework that can

be incorporated into the design toolchain of NoC/SoC design has not been established. By introducing proper extensions and adaptations, we expect to devise a principled approach to analyzing miscellaneous systems, including intra-vehicular networks [103], SoC/NoC systems [46], as well as quality-of-service provisioning and load balancing in software-defined networks (SDNs).

### 6.2.3 Network Traffic Analytics for Industrial Control Systems (ICSs)

To further enhance cybersecurity of SCADA systems, algorithms detecting attacks on SCADA protocols such as IEC 61850, Inter-Control Center Communication Protocol (ICCP), and Modbus protocol, still need to be developed. In addition to intrusion detection, the data collection method proposed in Chapter 4 can be exploited to develop other applications to enhance the efficiency, robustness, and resilience of industrial control systems. For instance, automated fault diagnosis [80] has been realized in zero-energy buildings using sensor readings and actuator status data. However, it remains to be examined whether fault/anomaly detection can be realized in critical industrial applications with data from a partial set of sensors and actuators (because there may be ICS filed devices that are not connected to network). We expect to further study network traffic generated by various applications and design data analysis system that can be incorporated into ICS network infrastructure.

# Bibliography


[1] Catalyst Switched Port Analyzer (SPAN) Configuration Example. https://www.cisco.com/c/en/us/support/docs/switches/catalyst-6500-series-switches/10570-41.html. Accessed: October 2018.

[2] US-CERT Alert TA17-163A: CrashOverride Malware. https://www.us-cert.gov/ncas/alerts/TA17-163A. Accessed: March 2019.

[3] Ali Abbasi and Majid Hashemi. Ghost in the PLC Designing an Undetectable Programmable Logic Controller Rootkit via Pin Control Attack. *Black Hat Europe*, pages 1–35, November 2016.

[4] Ali Abbasi and Majid Hashemi. Ghost in the PLC: Designing an Undetectable Programmable Logic Controller Rootkit via Pin Control Attack. In *Black Hat Europe '16*, pages 1–35, November 2016.

[5] Ephrem Ryan Alphonsus and Mohammad Omar Abdullah. A Review on the Applications of Programmable Logic Controllers (PLCs). *Renewable and Sustainable Energy Reviews*, 60(Supplement C):1185–1205, July 2016.

[6] Nicoleta Arghira, Daniela Hossu, Ioana Fagarasan, Sergiu Stelian Iliescu, and Daniel Razvan Costianu. Modern SCADA Philosophy in Power System Operation - A Survey. *University Politehnica of Bucharest Scientific Bulletin, Series C: Electrical Engineering*, 73(2):153–166, 2011.

[7] M. M. Awad, H. H. Halawa, M. Rentschler, R. M. Daoud, and H. H. Amer. Novel System Architecture for Railway Wireless Communications. In *IEEE EUROCON 2017 - 17th International Conference on Smart Technologies*, pages 877–882, July 2017.

[8] S. Azodolmolky, R. Nejabati, M. Pazouki, P. Wieder, R. Yahyapour, and D. Simeonidou. An Analytical Model for Software Defined Networking: A Network Calculus-Based Approach. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 1397–1402, December 2013.

[9] Rafael R. R. Barbosa, Ramin Sadre, and Aiko Pras. *Difficulties in Modeling SCADA Traffic: A Comparative Analysis*, pages 126–135.

[10] Rafael R. R. Barbosa, Ramin Sadre, and Aiko Pras. Difficulties in Modeling SCADA Traffic: A Comparative Analysis. In *Passive and Active Measurement (PAM)*, pages 126–135, March 2012.

[11] S. Bera, S. Misra, and J. J. P. C. Rodrigues. Cloud Computing Applications for Smart Grid: A Survey. *IEEE Trans. Parallel Distrib. Syst.*, 26(5):1477–1494, 2015.

[12] Steffen Bondorf, Paul Nikolaus, and Jens B. Schmitt. Delay Bounds in Feed-Forward Networks - A Fast and Accurate Network Calculus Solution. *arXiv preprint arXiv:1603.02094v1 [cs.NI]*, March 2016.

[13] Steffen Bondorf, Paul Nikolaus, and Jens B. Schmitt. Quality and Cost of Deterministic Network Calculus: Design and Evaluation of an Accurate and Fast Analysis. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(1):16:1–16:34, June 2017.

[14] Steffen Bondorf, Paul Nikolaus, and Jens B. Schmitt. Quality and Cost of Deterministic Network Calculus: Design and Evaluation of an Accurate and Fast Analysis. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(1):16:1–16:34, June 2017.

[15] Steffen Bondorf and Jens B. Schmitt. The DiscoDNC v2: A Comprehensive Tool for Deterministic Network Calculus. In *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS '14)*, pages 44–49, December 2014.

[16] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, 2001.

[17] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag, 2001.

[18] A. Bouillard, L. Jouhet, and E. Thierry. Tight Performance Bounds in the Worst-Case Analysis of Feed-Forward Networks. In *Proceedings of the 2010 IEEE Conference on Computer Communications (INFOCOM 2010)*, pages 1–9, March 2010.

[19] A. Bouillard and G. Stea. Exact Worst-Case Delay in FIFO-Multiplexing Feed-Forward Networks. *IEEE/ACM Transactions on Networking*, 23(5):1387–1400, October 2015.

[20] A. Bouillard and G. Stea. Exact Worst-Case Delay in FIFO-Multiplexing Feed-Forward Networks. *IEEE/ACM Transactions on Networking*, 23(5):1387–1400, October 2015.

[21] Anne Bouillard and Aurore Junier. Worst-Case Delay Bounds with Fixed Priorities Using Network Calculus. In *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS '11)*, pages 381–390, May 2011.

[22] Anne Bouillard and Giovanni Stea. Exact Worst-Case Delay in FIFO-Multiplexing Feed-Forward Networks. *IEEE/ACM Trans. Netw.*, 23(5):1387–1400, October 2015.

[23] Anne Bouillard and Giovanni Stea. Exact Worst-Case Delay in FIFO-Multiplexing Feed-Forward Networks. *IEEE/ACM Trans. Netw.*, 23(5):1387–1400, October 2015.

[24] G. Cena, S. Scanzio, and A. Valenzano. Experimental Evaluation of Seamless Redundancy Applied to Industrial Wi-Fi Networks. *IEEE Transactions on Industrial Informatics*, 13(2):856–865, April 2017.

[25] G. Cena, S. Scanzio, and A. Valenzano. A Prototype Implementation of Wi-Fi Seamless Redundancy with Reactive Duplication Avoidance. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 179–186, September 2018.

[26] Cheng-Shang Chang. *Performance Guarantees in Communication Networks*. Springer-Verlag.

[27] Cheng-Shang Chang. *Performance Guarantees in Communication Networks*. Springer-Verlag, 2000.

[28] Cheng-Shang Chang and R. L. Cruz. A Time Varying Filtering Theory for Constrained Traffic Regulation and Dynamic Service Guarantees. In *IEEE INFOCOM '99 – Conference on Computer Communications Proceedings*, volume 1, pages 63–70 vol.1, March 1999.

[29] X. Cheng, W. Lee, and X. Pan. Modernizing Substation Automation Systems: Adopting IEC Standard 61850 for Modeling and Communication. *IEEE Ind. Appl. Mag.*, 23(1):42–49, January 2017.

[30] X. Cheng, W. J. Lee, and X. Pan. Modernizing Substation Automation Systems: Adopting IEC Standard 61850 for Modeling and Communication. *IEEE Industry Applications Magazine*, 23(1):42–49, January 2017.

[31] X. Cheng, W. J. Lee, and X. Pan. Modernizing Substation Automation Systems: Adopting IEC Standard 61850 for Modeling and Communication. *IEEE Industry Applications Magazine*, 23(1):42–49, January 2017.

[32] Lucian Cojocar, Kaveh Razavi, and Herbert Bos. Off-the-Shelf Embedded Devices as Platforms for Security Research. In *Proceedings of the 10th European Workshop on Systems Security (EuroSec'17)*, pages 1:1–1:6, April 2017.

[33] R. L. Cruz. A Calculus for Network Delay, Part I: Network Elements in Isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

[34] N. Delgado, A. Q. Gates, and S. Roach. A Taxonomy and Catalog of Tuntime Software-Fault Monitoring Tools. *IEEE Transactions on Software Engineering*, 30(12):859–872, December 2004.

[35] R. Deng, G. Xiao, R. Lu, H. Liang, and A. V. Vasilakos. False Data Injection on State Estimation in Power Systems – Attacks, Impacts, and Defense: A Survey. *IEEE Transactions on Industrial Informatics*, 13(2):411–423, April 2017.

[36] Stephen Dunlap, Jonathan Butts, Juan Lopez, Mason Rice, and Barry Mullins. Using Timing-Based Side Channels for Anomaly Detection in Industrial Control Systems. *International Journal of Critical Infrastructure Protection*, 15(Supplement C):12–26, 2016.

[37] Samuel East, Jonathan Butts, Mauricio Papa, and Sujeet Shenoi. A Taxonomy of Attacks on the DNP3 Protocol. In *International Conference on Critical Infrastructure Protection (ICCIP)*, pages 67–81, March 2009.

[38] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. Stuxnet Dossier. *White Paper, Symantec Corp., Security Response*, 5(6), 2011.

[39] Marisol García-Valls, Jorge Domínguez-Poblete, Imad Eddine Touahria, and Chenyang Lu. Integration of Data Distribution Service and Distributed Partitioned Systems. *Journal of Systems Architecture*, 83:23–31, February 2018.

[40] Luis Garcia, Ferdinand Brasser, Mehmet Hazar Cintuglu, Ahmad-Reza Sadeghi, Osama A Mohammed, and Saman A Zonouz. Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit. In *Network and Distributed System Security Symposium (NDSS)*, 2017.

[41] Luis Garcia and Saman A Zonouz. Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit. In *Proceedings of the 2017 Network and Distributed System Security Symposium (NDSS '17)*, 2017.

[42] H. Georg, N. Dorsch, M. Putzke, and C. Wietfeld. Performance Evaluation of Time-Critical Communication Networks for Smart Grids Based on IEC 61850. In *Proceedings of the 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 43–48, April 2013.

[43] D. Germanus, A. Khelil, J. Schwandke, and N. Suri. Coral: Reliable and Low-Latency P2P Convergecast for Critical Sensor Data Collection. In *Proc. of the 2013 IEEE Int. Conf. on Smart Grid Communications (SmartGridComm)*, pages 300–305.

[44] Daniel Germanus, Abdelmajid Khelil, and Neeraj Suri. Increasing the Resilience of Critical SCADA Systems Using Peer-to-Peer Overlays. In *Proc. of the 1st Int. Conf. on Architecting Critical Systems*, pages 161–178.

[45] Asem Ghaleb, Sami Zhioua, and Ahmad Almulhem. On PLC Network Security. *International Journal of Critical Infrastructure Protection*, 22:62–69, September 2018.

[46] F. Giroudot and A. Mifdaoui. Buffer-Aware Worst-Case Timing Analysis of Wormhole NoCs Using Network Calculus. In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 37–48, April 2018.

[47] N. Govil, A. Agrawal, and N. O. Tippenhauer. On Ladder Logic Bombs in Industrial Control Systems. *arXiv:1702.05241 [cs.CR]*, February 2017.

[48] M. Hendawy, M. ElMansoury, K. N. Tawfik, M. M. ElShenawy, A. H. Nagui, A. T. ElSayed, H. H. Halawa, R. M. Daoud, H. H. Amer, M. Rentschler, and H. M. ElSayed. Application of Parallel Redundancy in a Wi-Fi-Based WNCS using OPNET. In *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–6, May 2014.

[49] T. Henriksson, P. van der Wolf, A. Jantsch, and A. Bruce. Network Calculus Applied to Verification of Memory Access Performance in SoCs. In *2007 IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia*, pages 21–26, October 2007.

[50] Hanan Hindy, David Brosset, Ethan Bayne, Amar Seeam, Christos Tachtatzis, Robert Atkinson, and Xavier Bellekens. A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets. *arXiv preprint arXiv:1806.03517*, 2018.

[51] Q. Huang, S. Jing, J. Li, D. Cai, J. Wu, and W. Zhen. Smart Substation: State of the Art and Future Development. *IEEE Transactions on Power Delivery*, 32(2):1098–1105, April 2017.

[52] Programmable Controllers - Part 3: Programming Languages. International standard, International Electrotechnical Commission (IEC), February 2013.

[53] Industrial Communication Networks - High Availability Automation Networks - Part 3: Parallel Redundancy Protocol (PRP) and High-Availability Seamless Redundancy (HSR). Iec standard, International Electrotechnical Commission (IEC), March 2016.

[54] IEEE Standard for Electric Power Systems Communications–Distributed Network Protocol (DNP3). Ieee standard, October 2012.

[55] IEEE Standard Communication Delivery Time Performance Requirements for Electric Power Substation Automation. Ieee standard, February 2005.

[56] D. M. E. Ingram, P. Schaub, D. A. Campbell, and R. R. Taylor. Performance Analysis of PTP Components for IEC 61850 Process Bus Applications. *IEEE Transactions on Instrumentation and Measurement*, 62(4):710–719, April 2013.

[57] D. M. E. Ingram, P. Schaub, R. R. Taylor, and D. A. Campbell. Performance Analysis of IEC 61850 Sampled Value Process Bus Networks. *IEEE Transactions on Industrial Informatics*, 9(3):1445–1454, August 2013.

[58] D. M. E. Ingram, P. Schaub, R. R. Taylor, and D. A. Campbell. Performance Analysis of IEC 61850 Sampled Value Process Bus Networks. *IEEE Transactions on Industrial Informatics*, 9(3):1445–1454, August 2013.

[59] Communication Networks and Systems for Power Utility Automation - Part 90-1: Use of IEC 61850 for the Communication between Substations. Iec tr 61850-90-1:2010, March 2010.

[60] Communication Networks and Systems for Power Utility Automation - Part 8-1: Specific Communication Service mapping (SCSM) - Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3. Iec 61850-8-1:2011, June 2011.

[61] Communication Networks and Systems for Power Utility Automation - Part 9-2: Specific Communication Service Mapping (SCSM) - Sampled Values over ISO/IEC 8802-3. Iec 61850-9-2:2011, September 2011.

[62] Communication Networks and Systems for Power Utility Automation - Part 5: Communication Requirements for Functions and Device Models. Iec 61850-5:2013, January 2013.

[63] Communication Networks and Systems for Power Utility Automation - Part 90-4: Network Engineering Guidelines. Iec tr 61850-90-4:2013, August 2013.

[64] Communication Networks and Systems for Power Utility Automation - Part 90-12: Wide Area Network Engineering Guidelines. Iec tr 61850-90-12:2015, July 2015.

[65] Communication Networks and Systems for Power Utility Automation - Part 90-2: Using IEC 61850 for Communication between Substations and Control Centres. Iec tr 61850-90-2:2016, February 2016.

[66] Helge Janicke, Andrew Nicholson, Stuart Webber, and Antonio Cau. Runtime-Monitoring for Industrial Control Systems. *Electronics*, 4(4):995–1017, December 2015.

[67] B. Kilberg, C. B. Schindler, A. Sundararajan, A. Yang, and K. S. J. Pister. Experimental Evaluation of Low-Latency Diversity Modes in IEEE 802.15.4 Networks. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 211–218, September 2018.

[68] Sergio Kimura, Andre Rotta, Ricardo Abboud, Rogério Moraes, Eduardo Zanirato, and Juliano Bahia. Applying IEC 61850 to Real Life: Modernization Project for 30 Electrical Substations. In *Proceedings of the 10th Annual Western Power Delivery Automation Conference*, April 2008.

[69] G. Kirubavathi and R. Anitha. Botnet Detection via Mining of Traffic Flow Characteristics. *Computers & Electrical Engineering*, 50:91–101, 2016.

[70] S. Kugele, D. Hettler, and J. Peter. Data-Centric Communication and Containerization for Future Automotive Software Architectures. In *2018 IEEE International Conference on Software Architecture (ICSA)*, pages 65–74, April 2018.

[71] D. Kushner. The Real Story of Stuxnet. *IEEE Spectrum*, 50(3):48–53, March 2013.

[72] Murat Kuzlu, Manisa Pipattanasomporn, and Saifur Rahman. Communication Network Requirements for Major Smart Grid Applications in HAN, NAN and WAN. *Computer Networks*, 67:74–88, July 2014.

[73] R. Langner. Stuxnet: Dissecting a Cyberwarfare Weapon. *IEEE Security & Privacy*, 9(3):49–51, May 2011.

[74] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient Backprop. In *Neural Networks: Tricks of the Trade, Second Edition*, pages 9–48. Springer Berlin Heidelberg, 2012.

[75] R Lee. CRASHOVERRIDE: Analysis of the Threat to Electric Grid Operations. *Dragos Inc.*, March 2017.

[76] Robert M. Lee, Michael J. Assante, and Tim Conway. Analysis of the Cyber Attack on the Ukrainian Power Grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, March 2016.

[77] Robert M. Lee, Michael J. Assante, and Tim Conway. Analysis of the Cyber Attack on the Ukrainian Power Grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, March 2016.

[78] H. Lei, C. Singh, and A. Sprintson. Reliability Modeling and Analysis of IEC 61850 Based Substation Protection Systems. *IEEE Transactions on Smart Grid*, 5(5):2194–2202, September 2014.

[79] Xiaomin Li, Di Li, Jiafu Wan, Athanasios V. Vasilakos, Chin-Feng Lai, and Shiyong Wang. A Review of Industrial Wireless Networks in the Context of Industry 4.0. *Wireless Networks*, 23(1):23–41, January 2017.

[80] C. Liu, S. Ghosal, Z. Jiang, and S. Sarkar. An Unsupervised Spatiotemporal Graphical Modeling Approach to Anomaly Detection in Distributed CPS. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pages 1–10, April 2016.

[81] Y. Liu, H. Gao, W. Gao, and F. Peng. Development of a Substation-Area Backup Protective Relay for Smart Substation. *IEEE Transactions on Smart Grid*, 8(6):2544–2553, November 2017.

[82] Sixing Lu, Minjun Seo, and R. Lysecky. Timing-Based Anomaly Detection in Embedded Systems. In *The 20th Asia and South Pacific Design Automation Conference*, pages 809–814, January 2015.

[83] R. E. Mackiewicz. Overview of IEC 61850 and Benefits. In *2006 IEEE PES Power Systems Conference and Exposition*, pages 623–630, October 2006.

[84] J. O. Malchow, D. Marzin, J. Klick, R. Kovacs, and V. Roth. PLC Guard: A Practical Defense against Attacks on Cyber-Physical Systems. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 326–334, September 2015.

[85] S. Mathew, S. Upadhyaya, M. Sudit, and A. Stotz. Situation Awareness of Multistage Cyber Attacks by Semantic Event Fusion. In *2010 Military Communications Conference (MILCOM)*, pages 1286–1291, October 2010.

[86] Peter Maynard, Kieran McLaughlin, and Berthold Haberler. Towards Understanding Man-in-the-Middle Attacks on IEC 60870-5-104 SCADA Networks. In *International Symposium for ICS & SCADA Cyber Security Research (ICS-CSR)*, September 2014.

[87] Peter Maynard, Kieran McLaughlin, and Sakir Sezer. Using Application Layer Metrics to Detect Advanced SCADA Attacks. In *International Conference on Information Systems Security and Privacy (ICISSP)*, pages 418–425, January 2018.

[88] Stephen McLaughlin and Patrick McDaniel. SABOT: Specification-Based Payload Generation for Programmable Logic Controllers. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*, pages 439–449, 2012.

[89] Stephen E McLaughlin, Saman A Zonouz, Devin J Pohly, and Patrick D McDaniel. A Trusted Safety Verifier for Process Controller Code. In *Proceedings of the 2014 Network and Distributed System Security Symposium (NDSS '14)*, 2014.

[90] Bill Miller and Dale Rowe. A Survey of SCADA and Critical Infrastructure Incidents. In *Proceedings of the 1st Annual Conference on Research in Information Technology (RIIT '12)*, pages 51–56, October 2012.

[91] Bill Miller and Dale Rowe. A Survey of SCADA and Critical Infrastructure Incidents. In *Proc. of the 1st Annu. Conf. on Research in Information Technology (RIIT '12)*, pages 51–56, 2012.

[92] Bill Miller and Dale Rowe. A Survey SCADA and Critical Infrastructure Incidents. In *Proceedings of the 1st Annual Conference on Research in Information Technology (RIIT)*, pages 51–56, October 2012.

[93] M. Mohiuddin, M. Popovic, A. Giannakopoulos, and J. Le Boudec. Experimental Validation of the Usability of Wi-Fi over Redundant Paths for Streaming Phasor Data. In *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 533–538, November 2016.

[94] Yacin Nadji, Manos Antonakakis, Roberto Perdisci, David Dagon, and Wenke Lee. Beheading Hydras: Performing Effective Botnet Takedowns. In *Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security (CCS '13)*, pages 121–132, 2013.

[95] P. Narang, S. Ray, C. Hota, and V. Venkatakrishnan. PeerShark: Detecting Peer-to-Peer Botnets by Tracking Conversations. In *Proc. of the 2014 IEEE Security and Privacy Workshops*, pages 108–115, 2014.

[96] Pratik Narang, Chittaranjan Hota, and Husrev Taha Sencar. Noise-Resistant Mechanisms for the Detection of Stealthy Peer-to-Peer Botnets. *Computer Communications*, 2016. in press.

[97] Julio Navarro, Aline Deruyver, and Pierre Parrend. A Systematic Survey on Multi-Step Attack Detection. *Computers & Security*, 76:214–249, July 2018.

[98] D. Papp, Z. Ma, and L. Buttyan. Embedded Systems Security: Threats, Vulnerabilities, and Attack Taxonomy. In *Proc. of the 2015 13th Annu. Conf. on Privacy, Security and Trust*, pages 145–152, 2015.

[99] P. Park, S. Coleri Ergen, C. Fischione, C. Lu, and K. H. Johansson. Wireless Network Design for Control Systems: A Survey. *IEEE Communications Surveys & Tutorials*, 20(2):978–1013, Second Quarter 2018.

[100] R. Lopez Perez, F. Adamsky, R. Soua, and T. Engel. Machine Learning for Reliable Network Attack Detection in SCADA Systems. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 633–638, August 2018.

[101] Frank Petruzella. *Programmable Logic Controllers*. McGraw-Hill Education, New York, NY, USA, 5 edition, 2017.

[102] M. Pignati, M. Popovic, S. Barreto, R. Cherkaoui, G. Dario Flores, J. Le Boudec, M. Mohiuddin, M. Paolone, P. Romano, S. Sarri, T. Tesfay, D. Tomozei, and L. Zanni. Real-Time State Estimation of the EPFL-Campus Medium-Voltage Grid by Using PMUs. In *2015 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5, February 2015.

[103] R. Queck. Analysis of Ethernet AVB for Automotive Networks Using Network Calculus. In *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, pages 61–67, July 2012.

[104] M. Rentschler, H. H. Halawa, R. M. Daoud, H. H. Amer, A. T. ElSayed, A. H. Nagui, M. M. ElShenawy, K. N. Tawfik, M. ElMansoury, M. Hendawy, and H. M. ElSayed. Simulative Comparison of Parallel Redundant Wireless Systems with OMNet++. In *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pages 1135–1140, June 2014.

[105] M. Rentschler and P. Laukemann. Performance Analysis of Parallel Redundant WLAN. In *2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, pages 1–8, September 2012.

[106] M. Rentschler and P. Laukemann. Towards a Reliable Parallel Redundant WLAN Black Channel. In *2012 9th IEEE International Workshop on Factory Communication Systems*, pages 255–264, May 2012.

[107] Luigi Rizzo. netmap: A Novel Framework for Fast Packet I/O. In *2012 USENIX Annual Technical Conference (USENIX ATC '12)*, pages 101–112, June 2012.

[108] Olivier Rossi and Philippe Schnoebelen. Formal Modeling of Timed Function Blocks for the Automatic Verification of Ladder Diagram Programs. In *Proceedings of the 4th International Conference on Automation of Mixed Processes - Hybrid Dynamic Systems (ADPM'2000)*, pages 177–182, 2000.

[109] C. Rossow, D. Andriesse, T. Werner, B. Stone-Gross, D. Plohmann, C. J. Dietrich, and H. Bos. SoK: P2PWNED - Modeling and Evaluating the Resilience of Peer-to-Peer Botnets. In *Proc. of the 2013 IEEE Symp. on Security and Privacy (SP)*, pages 97–111, 2013.

[110] Agustín Rullán. Programmable Logic Controllers versus Personal Computers for Process Control. *Computers & Industrial Engineering*, 33(1):421–424, October 1997.

[111] A. Sajid, H. Abbas, and K. Saleem. Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges. *IEEE Access*, 4:1375–1384, 2016.

[112] J. B. Schmitt, F. A. Zdarsky, and M. Fidler. Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch... In *Proceedings of the 27th IEEE Conference on Computer Communications (INFOCOM 2008)*, pages 1669–1677, April 2008.

[113] J. B. Schmitt, F. A. Zdarsky, and M. Fidler. Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch... In *Proc. IEEE INFOCOM 2008*, pages 1669–1677, April 2008.

[114] J. B. Schmitt, F. A. Zdarsky, and L. Thiele. A Comprehensive Worst-Case Calculus for Wireless Sensor Networks with In-Network Processing. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, pages 193–202, December 2007.

[115] J. B. Schmitt, F. A. Zdarsky, and L. Thiele. A Comprehensive Worst-Case Calculus for Wireless Sensor Networks with In-Network Processing. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, pages 193–202, Dec 2007.

[116] T. S. Sidhu and Y. Yin. Modelling and Simulation for Performance Evaluation of IEC61850-Based Substation Communication Systems. *IEEE Transactions on Power Delivery*, 22(3):1482–1489, July 2007.

[117] Sérgio S.C. Silva, Rodrigo M.P. Silva, Raquel C.G. Pinto, and Ronaldo M. Salles. Botnets: A Survey. *Computer Networks*, 57(2):378–403, February 2013.

[118] T. Skeie, S. Johannessen, and C. Brunner. Ethernet in Substation Automation. *IEEE Control Systems*, 22(3):43–51, June 2002.

[119] V. Skendzic and R. Moore. Extending the Substation LAN Beyond Substation Boundaries: Current Capabilities and Potential New Protection Applications of Wide-Area Ethernet. In *2006 IEEE PES Power Systems Conference and Exposition*, pages 641–649, October 2006.

[120] A. Soni, X. Li, J. Scharbarg, and C. Fraboul. Optimizing Network Calculus for Switched Ethernet Network with Deficit Round Robin. In *2018 IEEE Real-Time Systems Symposium*, pages 300–311, December 2018.

[121] A. Soni, X. Li, J. Scharbarg, and C. Fraboul. Optimizing Network Calculus for Switched Ethernet Network with Deficit Round Robin. In *2018 IEEE Real-Time Systems Symposium (RTSS)*, pages 300–311, December 2018.

[122] Ralf Spenneberg, Maik Brüggemann, and Hendrik Schwartke. PLC-Blaster: A Worm Living Solely in the PLC. In *Black Hat Asia '16*, 2016.

[123] Oregano Systems. syn1588® PCIe NIC - Technical Information, 2017.

[124] Z. Tian, Y. Cui, L. An, S. Su, X. Yin, L. Yin, and X. Cui. A Real-Time Correlation of Host-Level Events in Cyber Range Service for Smart Campus. *IEEE Access*, 6:35355–35364, June 2018.

[125] F. Tramarin, S. Vitturi, M. Luvisotto, and A. Zanella. On the Use of IEEE 802.11n for Industrial Communications. *IEEE Transactions on Industrial Informatics*, 12(5):1877–1886, October 2016.

[126] Nam Nhat Tran, Ruhul Sarker, and Jiankun Hu. An Approach for Host-Based Intrusion Detection System Design Using Convolutional Neural Network. In *International Conference on Mobile Networks and Management (MONAMI)*, pages 116–126, December 2017.

[127] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin. Intra-Vehicle Networks: A Review. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):534–545, April 2015.

[128] P. van der Wolf and J. Geuzebroek. SoC Infrastructures for Predictable System Integration. In *2011 Design, Automation Test in Europe*, pages 1–6, March 2011.

[129] A. Volkova, M. Niedermeier, R. Basmadjian, and H. de Meer. Security Challenges in Control Network Protocols: A Survey. *IEEE Communications Surveys & Tutorials*, pages 1–1, September 2018.

[130] X. Wang, C. Konstantinou, M. Maniatakos, R. Karri, S. Lee, P. Robison, P. Stergiou, and S. Kim. Malicious Firmware Detection with Hardware Performance Counters. *IEEE Transactions on Multi-Scale Computing Systems*, 2(3):160–173, July 2016.

[131] Tobias Wüchner, Martín Ochoa, Mojdeh Golagha, Gaurav Srivastava, Thomas Schreck, and Alexander Pretschner. MalFlow: Identification of C&C Servers Through Host-Based Data Flow Profiling. In *Proc. of the 31st Annu. ACM Symp. on Applied Computing*, pages 2087–2094, 2016.

[132] Q. Yan, Y. Zheng, T. Jiang, W. Lou, and Y. T. Hou. PeerClean: Unveiling Peer-to-Peer Botnets through Dynamic Group Behavior Analysis. In *Proc. of the 2015 IEEE Conf. on Computer Communications (INFOCOM)*, pages 316–324, 2015.

[133] Y. Yan, Y. Qian, H. Sharif, and D. Tipper. A Survey on Cyber Security for Smart Grid Communications. *Commun. Surveys Tuts.*, 14(4):998–1010, 4th Quarter.

[134] H. Yang, L. Cheng, and M. C. Chuah. Detecting Payload Attacks on Programmable Logic Controllers (PLCs). In *2018 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, May 2018.

[135] Huan Yang, Liang Cheng, and Xiaoguang Ma. Analyzing Worst-Case Delay Performance of IEC 61850-9-2 Process Bus Networks Using Measurements and Network Calculus. In *Proceedings of the Eighth International Conference on Future Energy Systems (e-Energy '17)*, pages 12–22. ACM, May 2017.

[136] Yanfang Ye, Tao Li, Donald Adjeroh, and S. Sitharama Iyengar. A Survey on Malware Detection Using Data Mining Techniques. *ACM Comput. Surv.*, 50(3):41:1–41:40, October 2017.

[137] M. R. D. Zadeh, T. S. Sidhu, and A. Klimek. Implementation and Testing of Directional Comparison Bus Protection Based on IEC 61850 Process Bus. *IEEE Transactions on Power Delivery*, 26(3):1530–1537, July 2011.

[138] L. Zhang, H. Gao, and O. Kaynak. Network-Induced Constraints in Networked Control Systems — A Survey. *IEEE Transactions on Industrial Informatics*, 9(1):403–416, February 2013.

[139] X. Zhang, Q. Han, and X. Yu. Survey on Recent Advances in Networked Control Systems. *IEEE Transactions on Industrial Informatics*, 12(5):1740–1752, October 2016.

[140] B. Zhu, A. Joseph, and S. Sastry. A Taxonomy of Cyber Attacks on SCADA Systems. In *Proc. of the 2011 IEEE Int. Conf. on Internet of Things, and Cyber, Physical and Social Computing (iThings/CPSCom)*, pages 380–388.

[141] B. Zhu, A. Joseph, and S. Sastry. A Taxonomy of Cyber Attacks on SCADA Systems. In *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, pages 380–388, October 2011.

[142] B. Zhu, S. Sastry, and A. Joseph. A Taxonomy of Cyber Attacks on SCADA Systems. In *2011 IEEE International Conference on Internet of Things and 4th IEEE International Conference on Cyber, Physical and Social Computing (iThings/CPSCom)*, pages 380–388, October 2011.

[143] S. Zonouz, J. Rrushi, and S. McLaughlin. Detecting Industrial Control Malware Using Automated PLC Code Analytics. *IEEE Security & Privacy*, 12(6):40–47, November 2014.

# VITA

## Education

**Lehigh University** — **Bethlehem, Pennsylvania, USA**
*Ph.D. Candidate in Computer Engineering* — *June 2013 ∼ Present*

**Illinois Institute of Technology** — **Chicago, Illinois, USA**
*M.S. in Electrical Engineering* — *August 2009 ∼ December 2011*

**Beijing University of Posts and Telecommunications** — **Beijing, P.R. China**
*B.E. in Telecommunication Engineering* — *September 2004 ∼ June 2008*

## Research Experience

**Constructing Meta-Models Using Machine Learning**
*Research Assistant, Sponsored by Ingersoll-Rand plc.* — *March 2019 ∼ Present*

**Delay Performance Analysis of Network Infrastructure of Cyber-Physical Systems**
*Research Assistant, NSF Project* — *January 2019 ∼ Present*

**Botnet and Intrusion Detection for Energy Delivery Systems**
*Research Assistant, U.S. Department of Energy (DoE) Project* — *June 2016 ∼ December 2018*

**Enhancing the Resilience of Field Devices in Industrial Control Systems**
*Research Assistant, DoE Project* — *October 2017 ∼ September 2018*

**Performance Analysis of Power Substation Automation Systems**
*Research Assistant, Sponsored by PITA and ABB Inc.* — *August 2014 ∼ July 2015*

**Data Mining and Disaggregation for Consumer Energy Usage Data**
*Research Assistant, Sponsored by PITA and PPL Utilities Corp.* — *June 2013 ∼ July 2014*

## Teaching Experience

**CSE-216 Software Engineering** — **Lehigh University**
*Teaching Assistant* — *January 2016 ∼ May 2016*

**ENGR-005 Introduction to Engineering Practice** — **Lehigh University**
*Teaching Assistant* — *August 2015 ∼ December 2015*

## Publications


[1] **Huan Yang**, Liang Cheng, and Mooi Choo Chuah. "Deep-Learning-Based Network Intrusion Detection for SCADA Systems". In: *International Workshop on Cyber-Physical Systems Security (CPS-Sec), part of IEEE CNS 2019*. June 2019, pp. 1–7. to appear.

[2] **Huan Yang** and Liang Cheng. "Bounding Network-Induced Delays of Wireless PRP Infrastructure for Industrial Control Systems". In: *2019 IEEE International Conference on Communications (ICC)*. May 2019, pp. 1–7. to appear.

[3] **Huan Yang**, Liang Cheng, and Xiaoguang Ma. "Poster Abstract: Combining Measurements and Network Calculus in Worst-Case Delay Analyses for Networked Cyber-Physical Systems". In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Apr. 2019, pp. 1–2. to appear.

[4] **Huan Yang**, Liang Cheng, and Xiaoguang Ma. "Work-in-Progress Abstract: Bounding Network-Induced Delays for Time-Critical Services in Avionic Systems Using Measurements and Network Calculus". In: *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems (with CPS-IoT Week 2019) (ICCPS '19), WiP and Poster Session*. Apr. 2019, pp. 1–2. DOI: 10.1145/3302509.3313330. to appear.

[5] Yanhong Yang, **Huan Yang**, Liang Cheng, and Xiaotong Zhang. "Optimal Time and Channel Assignment for Data Collection in Wireless Sensor Networks". In: *International Journal of Sensor Networks* 28.3 (Nov. 2018), pp. 165–178. DOI: 10.1504/IJSNET.2018.096261.

[6] **Huan Yang**, Liang Cheng, and Mooi Choo Chuah. "Detecting Payload Attacks on Programmable Logic Controllers (PLCs)". In: *2018 IEEE Conference on Communications and Network Security (CNS)*. May 2018, pp. 1–9. DOI: 10.1109/CNS.2018.8433146.

[7] **Huan Yang**, Liang Cheng, and Xiaoguang Ma. "Analyzing Worst-Case Delay Performance of IEC 61850-9-2 Process Bus Networks Using Measurements and Network Calculus". In: *Proceedings of the Eighth International Conference on Future Energy Systems (e-Energy '17)*. ACM, May 2017, pp. 12–22. DOI: 10.1145/3077839.3077856.

[8] **Huan Yang**, Liang Cheng, and Mooi Choo Chuah. "Modeling DNP3 Traffic Characteristics of Field Devices in SCADA Systems of the Smart Grid". In: *2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), part of CPS Week*. IEEE, Apr. 2017, pp. 1–6. DOI: 10.1109/MSCPES.2017.8064535.

[9] **Huan Yang**, Liang Cheng, and Mooi Choo Chuah. "Detecting Peer-to-Peer Botnets in SCADA Systems". In: *2016 IEEE Global Communications Conference Workshops (Cyber-Physical Smart Grid Security and Resilience Workshop)*. Dec. 2016, pp. 1–6. DOI: 10.1109/GLOCOMW.2016.7848877.

[10] **Huan Yang**, Liang Cheng, and Mooi Choo Chuah. "Evaluation of Utility-Privacy Trade-Offs of Data Manipulation Techniques for Smart Metering". In: *International Workshop on Cyber-Physical Systems Security (CPS-Sec), part of IEEE CNS 2016*. Oct. 2016, pp. 396–400. DOI: 10.1109/CNS.2016.7860526.

[11] Yanhong Yang, **Huan Yang**, Liang Cheng, and Xiaotong Zhang. "Abstract: Optimal Time and Channel Assignment for Data Collection in Wireless Sensor Networks". In: *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, Poster Session*. Oct. 2015, pp. 456–457. DOI: 10.1109/MASS.2015.90.