REAL-TIME BYZANTINE-RESILIENT POWER GRID INFRASTRUCTURE

by

Sahiti Bommareddy

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

January, 2025

Abstract

The power grid, critical to society and the economy, is increasingly targeted by sophisticated cyber attacks, especially from nation-state actors. These threats, at both system and network levels, aim to compromise key grid components, risking severe disruptions and blackouts. While much of the existing research focuses on isolated security concerns, it neglects complex threats to the broad grid infrastructure, especially in substations. This gap undermines grid resilience and endangers both lives and billions of dollars.

This thesis takes a step towards resilient grid infrastructure by introducing a novel comprehensive threat model and pioneering real-time, Byzantine-resilient solutions for grid infrastructure. We present the first real-time Byzantine-resilient architecture and protocols for the substation, ensuring correct protective operations even in the face of protective relay compromises and network attacks. We evaluate our implementation across a comprehensive range of fault-free and faulty operating conditions in relevant testbeds, demonstrating its ability to meet strict real-time latency requirements even in the worst operating conditions.

We introduce the first end-to-end Byzantine-resilient system framework for the broad grid infrastructure from the control center to the substation and field devices under the comprehensive threat model. We demonstrate the proposed system framework's ability to support real-time grid operations in a Byzantine-resilient manner. To enhance situational awareness, we integrate unsupervised machine learning models for anomaly detection.

The solutions we propose satisfy other critical domain needs, including continuous availability over a long system lifetime and seamless integration with the grid. We implemented all modules and protocols and made them available to the community within the open-source Spire Toolkit. The system has successfully withstood a purple team exercise and has been transitioned to SCADA manufacturers GE and Siemens, as well as two national laboratories PNNL and SANDIA. Finally, we propose a practical incremental deployment strategy for large-scale real-world power grid topologies.

Advisor: Dr. Yair Amir

Readers: Dr. Yair Amir, Dr. Yinzhi Cao and Dr. Amy Babay

Acknowledgments

I am deeply grateful to my advisor, Yair Amir, for his unwavering support, invaluable guidance, and constant encouragement throughout my journey. His mentorship has been more than an academic partnership; it has been truly transformative. Yair has an extraordinary ability to identify long-term, impactful problems, and his resilience and vision were especially evident during challenging times, such as the pandemic. Not only did he help us pursue our research goals with efficiency, but he also continued to teach in person, demonstrating his exceptional commitment to both students and the field. His boundless energy, optimism, and pursuit of excellence have inspired me to push beyond my own boundaries. Yair has taught me to balance the big picture with meticulous attention to detail, to strive for excellence consistently, and to maintain harmony between my professional and personal life. Working closely with him has not only shaped me as a researcher but also as an individual.

I am also deeply grateful to Amy Babay, my unofficial co-advisor, for her collaborative spirit, exceptional guidance and unwavering support throughout my research journey. Working together on various projects, including research papers, has been both enriching and inspiring. Amy's expertise and insight were crucial to my work, and I am especially grateful for her steadfast support as a member of both my GBO and dissertation committees.

I would like to extend my heartfelt thanks to Yinzhi Cao for making security research not only intellectually stimulating but also genuinely exciting. Working with him on one of my qualifying projects has been a truly enriching experience. His insightful advice over the years has been invaluable, and his contributions to both my GBO and dissertation committees have played a pivotal role in shaping my academic journey.

I would like to thank several faculty members at Johns Hopkins for their help and support throughout my journey. Thanks to James (Jim) Bellingham for our insightful discussions, his valuable advice, and for chairing my GBO committee. I am also grateful to Yuri Dvorkin for providing new perspectives on the energy sector and for serving on my GBO committee. Lastly, I'd like to thank Ilya Shpitser for giving me the opportunity to serve as teaching assistant, an experience I truly enjoyed and from which I learned a great deal.

I would also like to thank my fellow lab members at the DSN Lab — Daniel Qian,

Brian Wheatman, and Jerry Chen. A special thanks to Daniel Qian, who worked with me in the early stages of my research. I am also grateful to Brian and Jerry for their friendship, support, and for making the lab such a great environment throughout my journey.

I would also like to extend my gratitude to my collaborators at the RSSL Lab, University of Pittsburgh — Maher Khan, Huzaifah Nadeem, and Benjamin Gilby. Working with all of you has been a rewarding experience, and I truly appreciate the insights and expertise you brought to our joint research efforts.

I would like to express my sincere gratitude to the researchers and industry partners who collaborated with us on our research projects: Paul Skare, Christopher Bonebrake, David Jonathan Sebastian Cardenas, Cliff Eyre, Carl Miller, Beverly Johnson, Adrian Chavez, Kandy Phan, Linton Wells, Imes Chiu, John van de Lindt, Jagannadh Vempati and our other industry partners. Your collective expertise and guidance were instrumental in shaping and guiding my research, deepening my understanding, and enabling the development of practical, effective solutions.

I am also deeply thankful to Ramakrishna Sarma Chivukula, my undergraduate mentor, for introducing me to the world of research and for being a constant guide throughout my journey. His mentorship extended far beyond academics, helping me navigate personal challenges and making a profound impact on my career choices. His wisdom and support have shaped my path in ways that I will always cherish.

To my family, words cannot express my gratitude for your unwavering love, encouragement, and support. I am deeply thankful to my parents, Madhava Reddy and Kokila Devi, for instilling in me strong values, offering wise counsel in times of need, and celebrating my successes along the way. Your steadfast belief in me has been a constant source of strength and inspiration. None of this would have been possible without you. I also extend my thanks to my aunt, Vijaya, my in-laws, Ayyavaru Reddy and Seethamma, my niece Rishika, my nephews, and friends Srihari and Waqar for their love and constant support.

Finally, I want to express my heartfelt gratitude to my husband, Ashok Reddy, and my children, Subhash and Lasyapriya, for their unconditional love, patience, and understanding throughout this journey and in life. Your presence has been my greatest strength and I am forever grateful for your support.

During my time at Hopkins, I received support from the Department of Energy (DOE) Offices of Cybersecurity, Energy Security, and Emergency Response (CESER); Electricity (OE); and Nuclear Energy (NE) under the Grid Modernization Laboratory Consortium (GMLC) Topic 5.1.4 – Cyber-Physical Security, the Department of Defense (DoD) Strategic Environmental Research and Development Program (SERDP) grant RC20-1138, the Installation Technology Transition Program (ITTP) project "Severe Impact Resilience", and from the Department of Computer Science, Johns Hopkins University.

Contents

Abstract **ii**

Acknowledgments **iii**

List of Tables **viii**

List of Figures **ix**

1 Introduction **1**
1.1 Problem Statement 2
1.2 Solution Highlights 5
1.3 Thesis Organization 11

2 Background Work **13**
2.1 Byzantine Fault Tolerant State Machine Replication 13
2.2 Intrusion-Tolerant SCADA for Power Grid Control Center 15
2.3 Spire for the Control Center 16
2.3.1 Introduction 16
2.3.2 Threat Model 17
2.3.3 Algorithms and Architectures 19
2.4 Complementary Intrusion-Handling Approaches 20

3 Model **21**
3.1 System and Network Model 22
3.2 Assumptions 23
3.3 Fault and Threat Model 24
3.4 Service Properties 25

4 Real-Time Byzantine Resilience for the Substation **28**
4.1 Architecture 30
4.2 Arbiter Protocol 31

4.2.1 Protocol Description 32
4.3 Peer Protocol 33
4.3.1 Architecture Abstraction through Threshold Cryptography and Time Discretization 33
4.3.2 State Machine 34
4.4 Diversity and Proactive Recovery 38
4.5 Guarantees and Correctness Proof 39
4.5.1 Arbiter Protocol 39
4.5.2 Peer Protocol 40
4.6 Evaluation 41
4.6.1 Low Performance Servers Testbed Evaluation 44
4.6.2 Modern Industrial NUCs Testbed Evaluations 49
4.6.3 Modern Low Cost Data Center Servers Testbed Evaluations 55
4.7 Real-time Kernels in Support of Low-cost Solutions 55
4.7.1 Evaluation with Real-time Linux kernel 60
4.8 Discussion and Analysis of System Tradeoffs 64
4.9 Deployment and Purple Teaming 65

5 End-to-End Byzantine-Resilient SCADA 69
5.1 High Level Overview 69
5.2 End-to-End Byzantine-Resilient Architecture Design to Support Real-time Grid Operations 73
5.3 Control Center Command Operations and Status Updates 74
5.4 Substation Command Operations 78
5.5 Substation Autonomous Protection Operations 80
5.6 Service Properties 82
5.7 End-to-End Byzantine-Resilient Architecture for Large-Scale Deployments 82

6 Machine Learning based Intrusion Detection for Situational Awareness 85
6.1 Data Collection, Processing and Storage Pipeline 86
6.2 Feature engineering 87
6.3 Machine Learning Models, Training and Prediction 89
6.4 NIDS in action 91

7 Implementation 93
7.1 Supporting SCADA Communication Protocols 93
7.2 Software Relays and Emulated Relays 95
7.3 pvbrowser-based HMIs Implementation 96

7.4 Integrated Scenario and Power Topology Emulator 97

8 Deployment Strategies **99**
8.1 Securing the Network 99
8.2 Incremental Deployment of the Complete Solution 101

9 Conclusion **103**

Bibliography **105**

Vita **113**

List of Tables

4.1 Performance of the Arbiter Protocol and Peer Protocol under different operating conditions with four relay nodes ($f = 1, k = 1$) in Low Performance Servers Testbed 44
4.2 Performance of the Arbiter Protocol and Peer Protocol under different operating conditions with four relay nodes ($f = 1, k = 1$) in Modern Industrial NUCs Testbed 49
4.3 Performance of the Arbiter Protocol and Peer Protocol under different operating conditions with four relay nodes ($f = 1, k = 1$) in Modern Low Cost Data Center Servers Testbed 55
4.4 Performance of the Peer Protocol under different operating conditions with four relay nodes ($f = 1, k = 1$) in Low Performance Servers Testbed with normal and real-time kernels 60

6.1 The table below shows the detection performance of the packet-analysis based models, traffic-pattern based models, and the overall system across a range of attacks in testbed illustrated in Figure 6.1 91

List of Figures

1.1 High level overview of broad grid infrastructure with primary and cold-backup control centers, substations, PLCs and RTUs 4
1.2 High level overview of the proposed real-time Byzantine-resilient solution with SCADA Master Replicas deployed across multiple control center sites, Byzantine-resilient substations, and PLCs and RTUs secured with proxies. 5

2.1 Configuration 6 - Spire single site with six server ($f_c = 1, k_c = 1, N_c = 6$) 18
2.2 Configuration 3+3+3+3 - Spire four sites with twelve server ($f_c = 1, k_c = 4, N_c = 12$) 18

3.1 Grid SCADA Overview 21
3.2 Fundamental concept of false negative elimination with safety threshold (G_{ts}) and epsilon margin (ϵ). The event is detected (red region) if the measured state (G_{ms}) is at least or above $G_{ts} - \epsilon$ 26
3.3 Power grid over current protection with two safety thresholds (G_{ts}) and two corresponding ϵ margins, defining the *Safe Envelope*. The relay will issue trip (red regions) if the grid's measured state G_{ms} falls outside of the safe envelope. 26

4.1 An architecture that tolerates one Byzantine relay node ($f = 1$) and one relay node undergoing proactive recovery ($k = 1$) simultaneously with a total of four relay nodes in an IEC61850 Substation. 30
4.2 Peer Protocol partial state machine 35
4.3 Peer Protocol full state machine 36
4.4 Low Performance Servers Testbed Fault-Free operating condition performance evaluation with million actions 45
4.5 Low Performance Servers Testbed Peer Protocol Fail-Stop and Byzantine fault operating condition performance evaluation with million actions 46

4.6 Low Performance Servers Testbed Fail-Stop fault with proactive recovery operating condition performance evaluation with million actions . 47
4.7 Low Performance Servers Testbed Byzantine fault with proactive recovery operating condition performance evaluation with million actions 48
4.8 Modern Industrial NUCs Testbed Fault-Free operating condition performance evaluation with million actions 50
4.9 Modern Industrial NUCs Testbed Peer Protocol Fail-Stop and Byzantine fault operating condition performance evaluation with million actions 51
4.10 Modern Industrial NUCs Testbed Fail-Stop fault with proactive recovery operating condition performance evaluation with million actions . 52
4.11 Modern Industrial NUCs Testbed Byzantine fault with proactive recovery operating condition performance evaluation with million actions 53
4.12 Modern Low Cost Data Center Servers Testbed Fault-Free operating condition performance evaluation with million actions 56
4.13 Modern Low Cost Data Center Servers Testbed Peer Protocol Fail-Stop and Byzantine fault operating condition performance evaluation with million actions 57
4.14 Modern Low Cost Data Center Servers Testbed Fail-Stop fault with proactive recovery operating condition performance evaluation with million actions 58
4.15 Modern Low Cost Data Center Servers Testbed Byzantine fault with proactive recovery operating condition performance evaluation with million actions 59
4.16 Low Performance Servers Testbed normal and real-time kernels in Fail-stop fault operating condition performance evaluations with million actions 61
4.17 Low Performance Servers Testbed normal and real-time kernels in Fail-stop fault and proactive recovery operating condition performance evaluations with million actions 62
4.18 Low Performance Servers Testbed normal and real-time kernels in Byzantine fault and proactive recovery operating condition performance evaluations with million actions 63

5.1 End-to-end Byzantine-resilient architecture for simple RTUs or PLCs 72
5.2 End-to-end Byzantine-resilient architecture for sophisticated substation with a local Byzantine-resilient system 72
5.3 Control center to substation command operation steps 74
5.4 Substation to control center status update operation steps 77
5.5 Substation command operation steps 79
5.6 Substation autonomous protection operation steps 81

6.1 IDS integration with Spire 87

6.2 ML Data Pipeline 88

7.1 Control Center HMI for Integrated Scenario with DNP3 and Modbus PLCs at top and three IEC61850 substations at bottom 96
7.2 Three Substation HMIs for Integrated Scenario 97

Chapter 1

Introduction

Critical infrastructures, such as energy, water, and essential manufacturing are vital to the functioning of society. Supervisory Control and Data Acquisition (SCADA) systems are used to control and monitor these infrastructures. Originally designed to operate within isolated, air-gapped environments dependent on electro-mechanical devices, traditional SCADA has undergone significant evolution. In response to growing demands for cost efficiency, enhanced performance, and scalability, these systems have integrated modern microprocessor-based devices and standardized Information Technology (IT) platforms, extending their connectivity beyond conventional Operational Technologies (OT).

However, this increased interconnectivity has rendered SCADA systems more vulnerable to cyber threats, a reality underscored by the escalating number of cyberattacks targeting critical infrastructure [1–4]. Of particular concern is the power grid, a cornerstone upon which virtually all sectors rely, making it a prime target for determined nation-state actors. The power grid is a complex distributed system that interlinks thousands of power plants with millions of consumers through an extensive network of substations. This vast infrastructure presents numerous potential entry points for cyber intrusions [5]. In an era characterized by heightened interconnectivity and evolving cyber threats, even localized breaches pose significant risks to the grid's overall resilience.

To ensure the grid's resilience, SCADA systems must guarantee correct operations and continuous availability at expected levels of performance, even in the face of faults and cyber intrusions. Recent cyberattacks, alongside red team exercises, have highlighted the insufficiency of traditional perimeter defenses and established industry best practices in addressing sophisticated cyber threats [4,6]. While existing research and industry efforts focus on limited isolated security concerns, such as SCADA Master intrusions or firewall or single component fault, they fail to consider more complex, systematic threats and end-to-end grid operational resilience needs. As a result critical and vast sections of the grid remain vulnerable to cyber threats.

To the best of our knowledge, no comprehensive threat model currently exists that

fully accounts for threats to the broader grid ecosystem, including control centers, substations, Remote Terminal Units (RTUs), and Programmable Logic Controllers (PLCs). Moreover, a significant gap exists in the resilience of grid substations, and there is a lack of integrated solutions for developing end-to-end, Byzantine-resilient systems that encompass the wide range of grid SCADA operations.

This thesis takes a significant step towards Byzantine-resilient power grid infrastructure by defining a novel, comprehensive threat model with simultaneous Byzantine system faults and network attacks for a broad system model. It presents the first real-time Byzantine-resilient architecture and protocols for the substation. In a 60Hz power grid, such as in the US, these protocols ensure correct and timely protective operation with an extremely demanding quarter-cycle reaction time (4.167 milliseconds) even in the presence of protective relay compromises and network attacks.

Additionally, this thesis presents the first end-to-end Byzantine-resilient system framework for the broad grid infrastructure from the control center to the substation to power grid field devices, ensuring important safety and performance service properties at both the substation level and the broader grid level under the comprehensive threat model.

To further enhance situational awareness, we introduce a machine learning-based Network Intrusion Detection System (NIDS) that is custom-designed to offer real-time detection of anomalies and intrusions within the solution. An implementation of these concepts is made available to the community via the open-source Spire Toolkit [7]. Finally, we propose a two-phase deployment strategy that enables the practical, incremental, and scalable deployment of our solution across existing power grid infrastructures.

1.1 Problem Statement

A typical SCADA system for the power grid comprises several key components that ensure real-time monitoring and control. These include the Human-Machine Interface (HMI), control center SCADA Master, substations, and field control devices such as Programmable Logic Controllers (PLCs) and Remote Terminal Units (RTUs) (Figure 1.1). The HMI provides operators with a visual interface to monitor the grid and issue commands. The SCADA Master serves as the central server, responsible for processing data from these components (substations, PLCs and RTUs), issuing control commands to them, and delivering situational awareness through HMI visualization.

Substations include a key SCADA component, protective relays. The role of the protective relay is to monitor its part of the grid continuously, and upon detecting that the substation state is not as desired (due to a risky events such as short circuit), trip a breaker to disconnect the power circuit in order to protect grid assets. A high voltage (345kV and up) transformer is an example of an asset protected by protective relays in transmission substations. Since such a transformer serves vast spans of the

grid, costs millions of dollars, and takes over a year to procure, damaging it threatens grid stability. Key protection functions within substations include protection schemes such as over current protection, differential protection, distance protection, and transformer protection. During risky events, a rapid response is crucial to minimize damage and prevent further disturbances. Consequently, industry standards from organizations like IEEE and IEC mandate that these critical substation autonomous protective actions must adhere to a strict real-time latency requirement of a quarter-power cycle (4.167 ms in a 60 Hz system) [8].

Substations, PLCs and RTUs play another critical role, they control and monitor physical grid equipment based on commands from the control center. These components enable efficient grid operation by continuously monitoring power flow, voltage levels, and equipment status, ensuring stability, fault detection, and optimized performance across the grid. To provide real-time monitoring and control capabilities, SCADA systems for the power grid must deliver device status updates from field to control center and supervisory commands from control center to field within 100-200ms [9].

To address potential control center SCADA Master failures, current grid SCADA systems incorporate fault tolerance, typically using a primary-backup approach. These SCADA systems typically use a primary-backup approach, with a hot-backup of the central control SCADA Master taking over immediately if the primary master fails. Modern grid SCADA architectures can involve two control centers. A hot-backup SCADA master is used within each control center, and the cold-backup control center can be activated if the primary control center fails (Figure 1.1). However, these architectures only provide sufficient resilience to overcome benign failures, but they are not adequate to cope with the hostile environments that SCADA systems are now being exposed to i.e., they were never designed to withstand malicious attacks at both system and network level.

Compromises of the SCADA Master or protective relays, could have severe, grid-wide consequences. A compromised SCADA Master could issue malicious commands to damage power grid components and manipulate monitoring data, preventing operators from detecting and responding to problems. Similarly, a compromised relay could impact grid protection schemes, and if multiple substations are targeted, overall grid stability could be jeopardized [10]. Network-level cyberattacks can disrupt communication between SCADA components, impairing grid management, as most grid communications have stringent real-time latency requirements [8, 9].

Existing research has primarily focused on developing intrusion-tolerant SCADA Master solutions for control center, overlooking threats to other critical grid components. To the best of our knowledge, no existing model defines the complex threats, at both the system and network levels, across the broad grid infrastructure, spanning control centers, substations, Remote Terminal Units (RTUs), Programmable Logic Controllers (PLCs), and field devices. Furthermore, there is lack of solutions that ensure resilience specifically for substations and those that provide comprehensive,

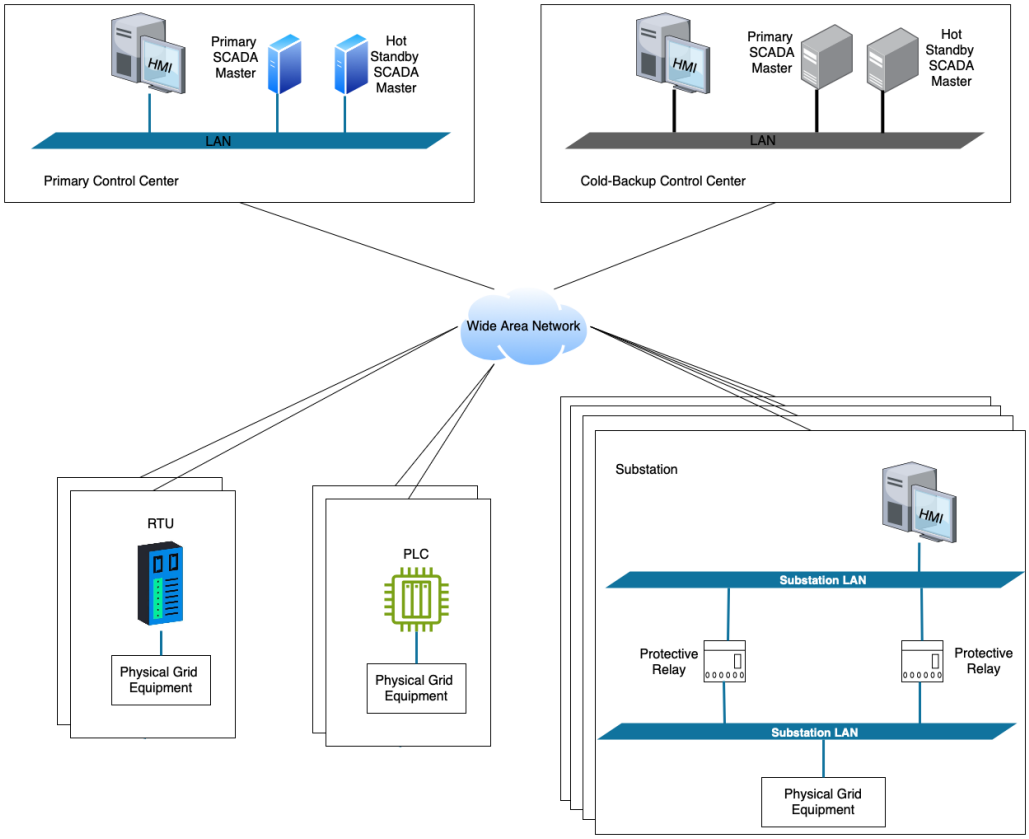


Figure 1.1: High level overview of broad grid infrastructure with primary and cold-backup control centers, substations, PLCs and RTUs

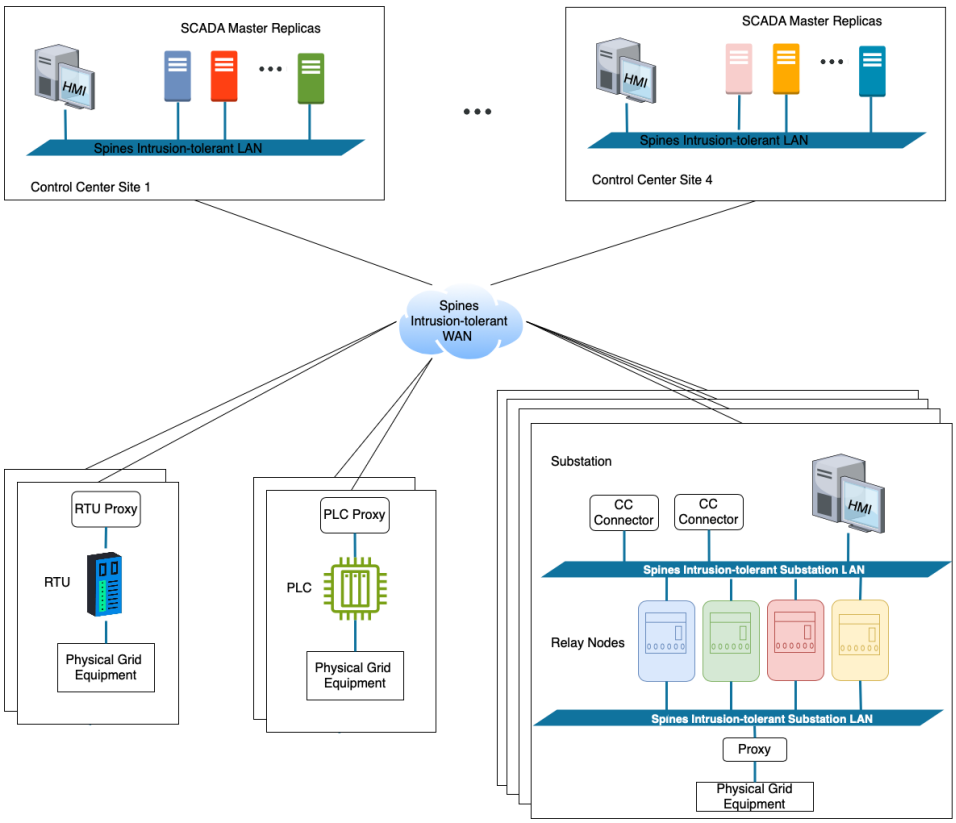


Figure 1.2: High level overview of the proposed real-time Byzantine-resilient solution with SCADA Master Replicas deployed across multiple control center sites, Byzantine-resilient substations, and PLCs and RTUs secured with proxies.

end-to-end resilience for grid operations as a whole, integrating all critical components.

1.2 Solution Highlights

This thesis introduces a novel and comprehensive threat model specifically designed to address the growing vulnerabilities of the broad power grid infrastructure. Our threat model considers Byzantine faults at both the system and network levels. This includes scenarios in which critical components, such as the SCADA Master and relays, are successfully compromised and fully controlled by a sophisticated attacker aiming to disrupt grid operations. It also considers a diverse range of network attacks targeting connectivity between grid components, further undermining grid resilience.

To defend against these multifaceted risks, we propose a set of innovative solutions grounded in the key concepts of Byzantine-resilient architectures, real-time

Byzantine-resilient protocols, and an intrusion-tolerant network. These solutions form the foundation for transforming vulnerable power grid systems into Byzantine-resilient infrastructures capable of withstanding sophisticated cyberattacks.

A key element of our approach is redundancy, particularly for relays and the SCADA Master. As illustrated in Figure 1.2, we deploy multiple protective relays in each substation, along with SCADA Master replicas, to tolerate Byzantine faults. Given that grid devices like relays were not originally designed to support Byzantine-resilient solutions, we augment them with a custom harness, collectively referred to as relay nodes. For other critical components, such as RTUs, PLCs, HMIs, and circuit breakers, we employ a proxy-based approach to provide security by effectively addressing vulnerabilities associated with insecure SCADA protocols (e.g., Modbus and DNP3). This approach restricts these protocols to direct wire connections between the proxy and the component.

Network threats are addressed through the use of modern switches in LANs and an intrusion-tolerant network service [7], which combines resilient networking architecture with an intrusion-tolerant overlay, effectively addressing malicious attacks and compromises in both the underlying infrastructure and the overlay itself. Thus, we secures all network communications. Even with intrusion-tolerant network service in place, hosting SCADA Master replicas at a single control center site remains a potential vulnerability, as sophisticated network attacks could isolate the control center site and disrupt operations. To mitigate this, we distribute the SCADA Master replicas across multiple geographically dispersed control center sites, offering greater resilience and flexibility for operators. The number of sites can be adjusted based on the desired level of resiliency, ensuring operators can tailor the system configuration to meet specific resilience requirements.

While redundancy is crucial, it is not sufficient on its own. If all replicas or nodes are identical, an attacker compromising one replica could easily exploit the others. To prevent this, we diversify the redundant components employed, ensuring that even if one replica is compromised, the attacker cannot leverage the same exploit across the entire system. Additionally, we support proactive recovery, which involves restarting replicas from a clean state, diversifying their attack surfaces, and refreshing cryptographic keys, to reclaim compromised components and making it significantly more difficult for an attacker to compromise and control the system.

At the core of the proposed solutions are real-time Byzantine-resilient protocols, which guarantee correct grid operations, even in the presence of Byzantine faults or malicious attacks. Given the stringent real-time latency requirements inherent to power grid operations, particularly substation protection operations that demand a quarter-cycle latency, designing protocols that can meet these latency constraints is a formidable challenge. To address this, we focus on these rigorous latency requirements, developing and employing protocols that ensure correct grid operation without sacrificing performance, even under challenging operating conditions with successful intrusion and network attacks. In addition to these core protocols, we enhance the

system with an out-of-band, machine learning-based situational awareness module that provides real-time anomaly detection, further bolstering the grid's resilience.

By integrating these techniques, we transform a vulnerable grid (Figure 1.1) into a Byzantine-resilient grid infrastructure (Figure 1.2). As shown in Figure 1.2, the proposed Byzantine-resilient architecture includes geo-distributed SCADA Master replicas, Byzantine-resilient substations with relay nodes, and RTUs/PLCs secured by proxies interconnected by an intrusion-tolerant network. These components work in unison, executing real-time Byzantine-resilient protocols to safeguard the grid against faults, successful intrusions and network attacks.

The primary contributions of this thesis are:

- **Comprehensive Threat Model:** This thesis defines a comprehensive threat model that considers all key components of a SCADA system, control centers, substations, Remote Terminal Units (RTUs), and Programmable Logic Controllers (PLCs). Our model incorporates Byzantine fault at both system and network levels, allowing us to address a broader range of potential vulnerabilities that impact grids.

  Our system has N_s relay nodes in each substation and N_c SCADA Master replicas overall. The system can withstand up to f_s Byzantine faults among the N_s relay nodes in each substation and up to f_c Byzantine SCADA Master faults simultaneously. Furthermore, it also considers network level attacks.

- **Four Service Properties:** The system is designed to enable end-to-end Byzantine-resilient grid operations. Our system guarantees four critical service properties under the comprehensive threat model:

  1. **Substation Safety:** If a valid protective operation is executed by the breaker node, at least one correct relay node issued that action within the last reaction quarter.
  2. **Substation Real-Time Reaction:** If the grid state necessitates a protective action due to an event, that action will be issued for that event's reaction quarter.
  3. **SCADA Safety:** If two correct SCADA Master replicas execute the i^{th} operation, then those operations are identical.
  4. **SCADA Bounded-Latency Reaction:** If a correct system component initiates an operation, the latency for that operation to be executed by a correct SCADA Master replica is upper bounded.

- **Byzantine-Resilient Architecture for Grid Substations:** We present the first real-time Byzantine-resilient architecture for the power grid substations.

Our architecture can tolerate f_s Byzantine relay faults and k_s relays undergoing proactive recovery simultaneously with just $2f_s + k_s + 1$ total relays. This is significantly more efficient than traditional Byzantine Fault Tolerant State Machine Replication-based solution that would need $3f_s + 2k_s + 1$ total relays. As the relays are costly devices, with a substantial number of substations in the grid, this architecture can lead to significant cost savings. This reduction in cost directly contributes to the feasibility of widespread implementation across the grid.

- **Real-Time Byzantine-Resilient Protocols for Grid Substations:** We present the first real-time Byzantine-resilient protocols for the grid substations that simultaneously address relay compromises and network attacks while meeting the strict reaction time requirement. In a 60Hz power grid, these protocols must comply with a quarter-cycle reaction time standard (4.167ms in a 60Hz system). This real-time latency constraint is extremely demanding in the presence of attacks and successful intrusions.

  We deploy and benchmark the architecture and protocols in a range of fault-free and faulty operating conditions across multiple relevant testbeds that represent real substation environments. This setup enables us to evaluate each protocol's ability to meet the stringent quarter-power cycle (4.167 ms) latency requirement, while also analyzing the associated trade-offs. The testbeds are strategically selected to represent different segments of the grid computing landscape, reflecting a variety of potential deployment environments.

  The system successfully withstood a rigorous purple team exercise that included protective relays from Siemens, GE, and Hitachi Energy in 2022. It was transitioned to SCADA manufacturers GE and Siemens and to two national labs (PNNL and SANDIA).

- **End-to-End Byzantine-Resilient SCADA Architecture:** We present the first end-to-end Byzantine-resilient architecture for the broad system model, ensuring seamless Byzantine resilience integration across the entire grid infrastructure. Furthermore, this architecture enables the deployment of solutions in a variety of configurations: individual sub-systems (such as for just a substation or a control center) or combined end-to-end systems (such as for control center and few substations or the entire grid). This flexibility allows power grid operators to deploy tailored solutions that meet both security requirements and operational needs, whether securing isolated subsystems or managing the entire grid infrastructure.

- **Real-Time Grid Operations in End-to-End Byzantine-Resilient System:** We demonstrate grid operations, commands from the control center,

commands from a substation, substation autonomous protection operations and status or monitoring updates within the end-to-end Byzantine-resilient system.

These operations are demonstrated using a power scenario, called integrated scenario. It includes, control center, PLCs, RTUs, and substations. This approach highlights the system's ability to support real-time SCADA operations in an Byzantine-resilient manner.

- **Meeting Domain Requirements for Practical and Effective Solution:** Our system provides the following key features:

  1. **Long system lifetime with continuous availability:** It is achieved by supporting proactive recovery and diversity. Our protocols only guarantee correctness as long as the number of compromised SCADA Master replicas and relay nodes does not exceed the tolerated threshold. However, if all replicas or nodes in the system are identical copies of one another, an attacker who successfully exploits one replica or node can simply reuse the same exploit to compromise all of the other identical ones.

     To prevent an attacker from gaining control of more than the tolerated threshold, the system must ensure that the replicas and nodes present diverse attack surfaces. Diversity can be achieved using approaches such as N-version programming [11,12], operating system diversity [13], or software diversification at compilation or run time [14–16] or by using components from different SCADA manufacturers.

     Even if replicas and nodes are sufficiently diverse, given enough time, a dedicated attacker will eventually be able to craft enough distinct attacks to compromise more than tolerated threshold. Therefore, it is necessary to use proactive recovery to ensure survivability over the lifetime of the system.

     In proactive recovery, each replica is periodically brought down and restarted from a known clean state (removing any compromises) with a new diverse variant of the software (and potentially of the entire operating system) that is with high probability different from all past and future variants. This makes the job of the attacker significantly harder, as they now must simultaneously compromise more than tolerated threshold within a limited time window [17].

  2. **Seamless integration without altering existing grid devices:** Our proposed Byzantine-resilient architecture is decoupled from grid devices, ensuring that existing hardware (relays, HMIs, breakers) can operate without being aware of the underlying Byzantine-resilient mechanisms. This abstraction simplifies deployment and avoids the need for modifications to legacy devices.

3. **Modularity:** The solution is designed as a system of systems, enabling the independent deployment of subsystems that can later be integrated into a larger infrastructure. This modularity supports incremental deployment and scaling, ensuring that the system can evolve alongside the grid's growing needs.
4. **Interoperability:** SCADA systems often consist of devices from multiple manufacturers, each potentially using different communication protocols. Our solution addresses this challenge by using a proxy-based approach to ensure interoperability across heterogeneous devices, without requiring changes to the underlying devices or protocols. By securely bridging disparate communication protocols, we ensure that the system remains resilient, even as industry standards evolve.

- **Machine Learning Modules for Situational Awareness:** To complement our Byzantine-resilient solutions, we developed a machine learning-based Network Intrusion Detection System (NIDS) that enhances real-time situational awareness. Unlike traditional SCADA IDS solutions that rely on rule-based or signature-based techniques, our IDS operates out-of-band without introducing new vulnerabilities or impacting grid operations. It uses unsupervised machine learning models to detect anomalies and intrusions in the network traffic. Our approach enables high detection rates while minimizing false positives, providing operators with timely alerts of potential intrusions without compromising system performance.

- **Open-Source Release through Spire Toolkit:** We implement our solution modules and protocols, including the IDS modules, within the open-source Spire Toolkit [7] because we envision it as a comprehensive bag of tools for grid operators, offering flexibility and scalability for Byzantine-resilient SCADA system deployments. The Spire Toolkit is designed to meet the diverse and evolving security needs of modern power grid infrastructures.

  While the original Spire system focused on Byzantine-resilient control center SCADA Master operation, we have extended it by providing Byzantine-resilient systems for substations and end-to-end system framework, along with NIDS. The Spire Toolkit enables the deployment of SCADA systems in a variety of configurations: individual sub-systems (such as for substations or control centers) or combined end-to-end systems. This flexibility allows power grid operators to deploy tailored solutions that meet both security requirements and operational needs, whether securing isolated subsystems or managing the entire grid infrastructure.

- **Deployment Strategies:** The deployment strategy is designed to transition vulnerable power grids to Byzantine-resilient infrastructures incrementally. The

strategy includes:

Securing the Network: Prioritize securing the network through intrusion-tolerant network services that defend against both external and internal network attacks.

Incremental Deployment of Complete Solution: Gradual deployment of Byzantine-resilient architectures and protocols in legacy systems. This phased approach allows grid operators to modernize their infrastructure by addressing vulnerabilities in a strategic way while reducing risk, economic burden and disruption to grid operations.

1.3 Thesis Organization

The remainder of this thesis is organized as follows:

- Chapter 2 reviews related works on Byzantine Fault Tolerant State Machine Replication protocols, Intrusion-tolerant SCADA systems and other complementary intrusion tolerance approaches.
- Chapter 3 presents the system and threat models along with assumptions and service properties.
- Chapter 4 presents the first Byzantine-resilient architecture for substations, along with two real-time Byzantine-resilient protocols: the Arbiter Protocol and the Peer Protocol. It also includes comprehensive performance evaluations of both protocols under varying operating conditions across multiple testbeds. We discuss the performance of the protocols in these test environments, highlighting system trade-offs and other insights. Additionally, this chapter presents a purple team engagement exercise conducted by a hacker team from Sandia National Labs (SNL) on the system at Pacific Northwest National Laboratory (PNNL).
- Chapter 5 presents the first end-to-end Byzantine-resilient SCADA system architecture. We describe the transaction flow for facilitating end-to-end Byzantine-resilient operations for control center, substation and autonomous protection.
- Chapter 6 presents the machine learning based situational awareness module customized for our system.
- Chapter 7 presents some system implementation details such as supporting different SCADA protocols, emulated relays, software relays and implemented power scenarios.
- Chapter 8 presents a two-step deployment strategy that facilitates the seamless, incremental adoption of our solutions across the power grid infrastructure at scale.

- Chapter 9 concludes the thesis.

Chapter 2

Background Work

2.1 Byzantine Fault Tolerant State Machine Replication

Byzantine Fault Tolerant (BFT) State Machine Replication (SMR) protocols are designed to guarantee safety (correctness and consistency of the system state) and liveness (progress in processing updates) of distributed systems even in the presence of Byzantine faults — malicious behaviors such as arbitrary node failures [18]. Byzantine Reliable Broadcast (BRB) established the concept of reliable broadcast to ensure message delivery integrity even in the presence of Byzantine faults, forming a foundational component for many subsequent BFT protocols [19]. The seminal works of Paxos [20] and Practical Byzantine Fault Tolerance (PBFT) marked significant milestones in this field [17]. Paxos, introduced in the late 1980s, provides a protocol for achieving consensus among distributed nodes under benign fault conditions. PBFT, extends this research by effectively addressing Byzantine faults, enabling a set of nodes to agree on the ordering and execution of commands despite up to f Byzantine faulty nodes in a network of $3f + 1$ replicas. To support long system lifetimes, it is necessary to employ proactive recovery. Proactive recovery allows replicas to be periodically taken down and restored to a known clean state [21]. Ensuring continuous availability with proactive recovery typically requires $3f + 2k + 1$ total replicas to simultaneously tolerate up to f compromised replicas and k recovering replicas simultaneously [22].

The evolution of BFT protocols has focused on optimizing performance metrics such as throughput, latency, and scalability while maintaining stringent fault tolerance guarantees. Asynchronous PBFT relaxed the synchronous network assumption of PBFT, enhancing its applicability to networks where message delays and timing variations are prevalent, thus improving efficiency without compromising security [23, 24]. Further optimizations include Zyzzyva [25], which streamlined the communication overhead in PBFT by reducing redundant message exchanges using

speculation. Protocols like Steward, BLink and Menicus built efficient replicated state machines for wide-area networks. [26–28]. BFT-SMaRt introduced modularity in BFT protocol design, allowing for easier experimentation and integration of new consensus algorithms [29]. Prime addressed challenges related to network asynchrony and scalability, optimizing BFT performance for high-demand applications [30]. Protocols like MinBFT and Cheap BFT reduce the number of replicas needed by making stronger assumptions [31, 32]. Protocols like RAM, EBAWA, and Aliph focus on improving performance in the fault-free case [33–35]. Additionally, works have explored separating agreement (consensus) and execution components of BFT SMR protocols to leverage public cloud infrastructures while preserving privacy and security [36–39].

The rise of blockchain technology driven innovation in BFT protocols, particularly in the context of decentralized networks. Protocols like Hotstuff [40] introduced a leader-based BFT consensus algorithm that significantly reduced latency and resource requirements compared to traditional approaches, influencing subsequent developments in blockchain consensus mechanisms such as Libra's BFT-based protocol [41]. HoneyBadger leveraged batching and concurrent execution to achieve BFT consensus with high throughput, ensuring robustness and scalability in distributed ledger systems [42]. Tendermint implemented a BFT consensus algorithm using delegated Proof of stake, gossip protocol and efficient leader election to providing fast finality and consistency guarantees essential for transactions [43]. It is to be note that this class of BFT SMR protocols are built specifically for blockchains to scale and are not suitable for latency sensitive applications and critical infrastructure domain.

Another line of BFT SMR protocols research is in the context of applications and services running on high performance computing clusters (HPC). It includes fault-tolerant SMR protocols such as HovercRaft [44], DARE [45], and Hermes [46] that can achieve very low latencies (in tens of microseconds) in the fault-free case but can incur latencies in the tens of milliseconds during leader election. To achieve the excellent latency in the fault-free case, they require a special NIC (Network Interface Card), a lossless network, and Remote Direct Memory Access (RDMA) support [47] to eliminate CPU and I/O bottlenecks. These protocols are designed for applications deployed in HPC clusters, relying on high-speed networks (e.g., 100Gig/sec), making them unsuitable for the more conservative power grid SCADA environment with its need for rugged hardware and traditional networks (e.g., 1 Gig/sec).

The evolution of BFT SMR protocols showcases ongoing advancements in addressing fault tolerance, performance optimization, and adaptation to new technological environments. Although our solution utilizes the Prime replication engine to handle Byzantine faults in SCADA Masters, existing BFT SMR protocols fall short for power grid substations. These substations demand latencies of 4.167 milliseconds, while current BFT SMR protocols typically exhibit latencies that are one to two orders of magnitude greater. This gap/vacuum highlights the critical need for novel solutions to meet the stringent latency requirements of power grid systems even under worst operating conditions.

2.2 Intrusion-Tolerant SCADA for Power Grid Control Center

Previous research on intrusion-tolerant power grid infrastructure has predominantly focused on addressing Byzantine faults in SCADA Master component of control center using Byzantine fault tolerant state machine replication (BFT SMR) based approaches. Early works Zhao et al. [48] and Kirsch et al. [49] used PBFT and Prime replication protocols to overcome SCADA Master compromises.

Zhao et al. [48] used PBFT [17] with four replicas setup in a local-area network to overcome one compromise and demonstrated that the setup can sustain the subsecond sampling rate required by SCADA operations. Meanwhile, Kirsch et al. [49] use Prime to add intrusion tolerance to a Siemens SCADA product in a prototype implementation. Their work also identified and addressed several challenges in integrating modern intrusion-tolerant protocols with conventional SCADA architectures, such as employing a logical timeout protocol for the SCADA master replicas to agree on a logical time at which to poll field devices for the latest status.

However, these early works are limited to single control center, failing to account for network attacks that could sever connections between the control center and the rest of the network. Moreover, they do not consider the resilience of substations that need to support local autonomous protection operations. Furthermore, the lack of a comprehensive end-to-end threat model in existing research leaves significant gaps in understanding and addressing the full spectrum of vulnerabilities inherent in SCADA systems.

Building on these foundational works, subsequent research has sought to extend SCADA Master replication across multiple geographic sites. Notable works include [50] and Spire [51]. The work in [50] replicates a SCADA Master and Distribution Management System across three geographic sites using the MinBFT [31]. Each replica is placed in its own site, resulting in an extended threat model that supports a threshold of system-level compromises or benign site failures. However, the architecture does not support sophisticated network attacks considered by Spire and raises concerns about scalability and performance due to the use of a separate site for each replica.

Spire, to the best of our knowledge, is the only intrusion-tolerant SCADA system that integrates both SCADA Master compromises at the system level and network attacks into its comprehensive threat model [51]. This dual focus significantly enhances its resilience against sophisticated cyber threats. The architecture of Spire features a SCADA master developed from scratch, utilizing Remote Terminal Unit (RTU) and Programmable Logic Controller (PLC) proxies to maintain an event-driven SCADA system. Despite these innovations, Spire does not adequately address the resiliency requirements of substations, which are critical for grid resiliency. Additionally, it

lacks a comprehensive framework for enabling end-to-end resiliency, leaving significant sections of grid vulnerable. We are motivated by Spire's foundational principles and will explore its architecture and design decisions in greater detail in Chapter 2.3

In parallel, SMaRt-SCADA [52] was developed as an intrusion-tolerant SCADA prototype by integrating Eclipse NeoSCADA [53] with the BFT-SMaRt intrusion-tolerant replication engine [29], both are credible open-source projects. The authors of SMaRt-SCADA identify several challenges with making a traditional SCADA master support intrusion-tolerant replication: ensuring determinism is hard in systems built around non-deterministic features such as timeouts and multiple entry points for messages. To address these challenges, SMaRt-SCADA employs a proxy architecture that serializes all incoming and outgoing messages for each system component, including the SCADA Master, Human-Machine Interface (HMI), and RTU frontend. It also utilizes a logical timeout protocol akin to that of Kirsch et al. [49] to synchronize timeouts among the replicas. However, similar to its predecessors, SMaRt-SCADA is confined to a single control center and neither considers the implications of network attacks nor adequately address the resiliency requirements of substations and lacks a comprehensive framework for enabling end-to-end resiliency.

In summary, while significant advancements have been made in developing intrusion-tolerant systems for SCADA Masters through Byzantine fault tolerance and multi-site replication, critical gaps remain. Notably, existing research does not adequately address vulnerabilities in substations, which must operate autonomously to provide critical protection and seamlessly integrate with other grid systems. Additionally, existing threat models often overlook comprehensive resiliency across the entire spectrum of SCADA operations, underscoring the need for continued research to bridge these gaps and enhance the overall resilience of power grid infrastructures.

2.3 Spire for the Control Center

2.3.1 Introduction

Spire is the first intrusion-tolerant SCADA system that simultaneously addresses SCADA Master system compromises and network attacks. By employing Prime replication engine, along with proactive recovery and diversity, Spire ensures operational safety as long as no more than a predefined threshold of SCADA Master replicas are compromised.

To address sophisticated network attacks capable of targeting and isolating a control center, Spire integrates an advanced intrusion-tolerant networking approach with a novel architecture that distributes SCADA master replicas across at least three geographically diverse sites. This architecture not only offers enhanced intrusion tolerance but also allows for one of these sites to be a commodity data center, significantly reducing the financial burden on organizations that might otherwise need to

invest in an additional high-cost third control center.

2.3.2 Threat Model

Spire's threat model includes system-level compromises as well as network-level threats. At the system level, it considers compromised (Byzantine) SCADA master replicas that are completely under the control of the attacker and may exhibit arbitrary behavior. At the network level, it considers network link failures, misconfigurations, and malicious network attacks, including (but not limited to) routing attacks (e.g. BGP hijacking [54]) and sophisticated denial-of-service attacks (e.g. Coremelt [55] and Crossfire [56]) that can isolate a targeted site. Spire uses Spines's intrusion-tolerant networking foundation to addresses this broad network threat model and allows us to reduce it to a narrower model that assumes only a single site can be disconnected. Hence, in Spire they develop an architecture that simultaneously tolerates the compromise of up to f_c SCADA master replicas, the disconnection of one system site (possibly a control center), and the unavailability of one replica due to proactive recovery.

The threat model does not include rogue operators or compromised HMI, PLCs or RTUs. Spire uses proxies for all system endpoints, including HMIs, RTUs and PLCs making them considerably more difficult to compromise.

As long as no more than f_c SCADA master replicas are simultaneously compromised, Spire guarantees consistency of the system state. Specifically, it guarantees safety: if two correct replicas execute the i^{th} update, then those updates are identical. Moreover, proactive recovery forces an attacker to compromise more than f_c replicas within a confined time window to undermine system guarantees (rather than over the entire lifetime of the system) .

In terms of performance, Spire guarantees bounded delay: the latency for an update introduced by an authorized system component to be executed by a correct SCADA master replica is upper bounded. At any time, we assume that at most one site may be disconnected from the rest of the network. To provide bounded delay, Spire requires that all of the correct SCADA master replicas, with the exception of those in the disconnected site, are able to communicate with one another. Moreover, at least one correct SCADA master replica in a control center must be able to communicate with field substations.

Due to the network stability requirements of Prime replication engine, communication must meet the bounded-variance property of [30], which requires that for each pair of correct servers, the network latency does not vary by more than a factor K_{Lat}.

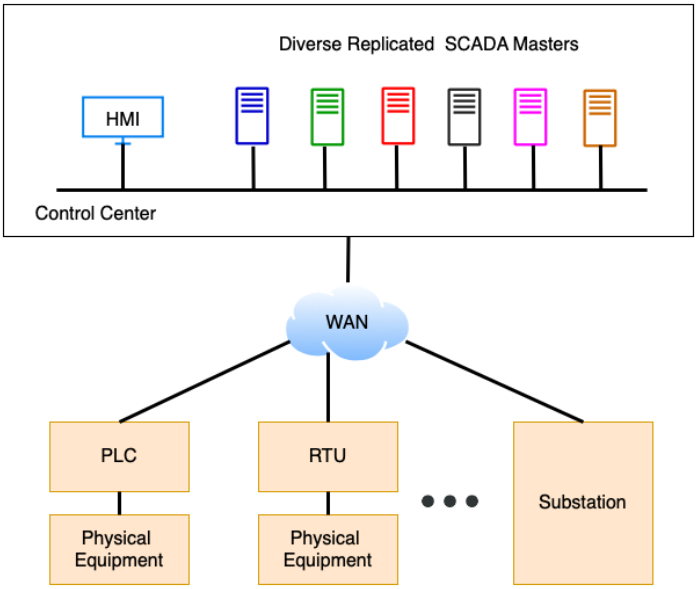


Figure 2.1: Configuration 6 - Spire single site with six server ($f_c = 1, k_c = 1, N_c = 6$)

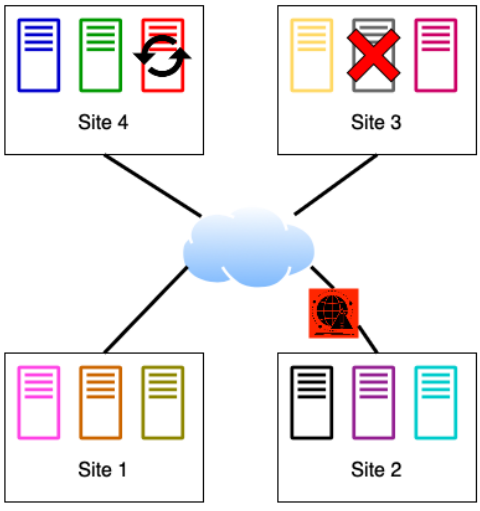


Figure 2.2: Configuration 3+3+3+3 - Spire four sites with twelve server ($f_c = 1, k_c = 4, N_c = 12$)

2.3.3 Algorithms and Architectures

Initial efforts to create intrusion-tolerant SCADA used replication within a single control center, using $3f_c + 1$ replicas (4 for $f_c = 1$) to tolerate f_c intrusions or $3f_c + 2k_c + 1$ replicas (6 for $f_c = 1, k_c = 1$) to simultaneously tolerate f_c intrusions and k_c proactive recoveries. Spire extends existing industry standard architectures to support intrusion tolerance as shown in Figure 2.1, by deploying six replicas in a control center. However, this configuration (6), cannot tolerate a control center going down or becoming disconnected due to a network attack.

To address these limitations, the next logical step is to distribute replicas across two sites. BFT SMR protocols need $2f_c + k_c + 1$ out of $3f_c + 2k_c + 1$ to be up, correct and connected in order to make progress. Targeting the larger of the two sites will disconnect a majority of the system replicas, leaving the rest unable to make progress without them. Thus, we need more than two sites for robust intrusion-tolerance.

To support the full threat model, including simultaneous site disconnection, an intrusion and a proactive recovery, k_c is set to the number of replicas in the largest site plus the maximum number of simultaneous proactive recoveries (in this case, one). This ensures that there are $2f_c + k_c + 1$ correct and connected replicas at all times, allowing system to be not just alive but provide bounded delay. A viable solution is the "6+6+6" configuration, involving three sites with six SCADA Master replicas each, totaling 18 replicas. This setup maintains performance even with simultaneous site failure, intrusion, and proactive recovery. Alternatively, the "3+3+3+3" configuration, shown in Figure 2.2, uses 12 replicas across four sites, requiring an additional site but providing same resilience. Detailed discussions on configurations, trade-offs, and performance are available in [51].

As discussed in Chapter 1, current industry-standard SCADA architectures typically use two control center sites. However, advanced intrusion-tolerant architectures require at least three geographically dispersed sites to ensure robust security and operational resilience. Establishing and maintaining additional control centers with comprehensive capabilities for interfacing with Remote Terminal Units (RTUs), Programmable Logic Controllers (PLCs), and managing power grid infrastructure entail substantial financial investment. To address this, Spire introduces an innovative solution by deploying two dedicated control centers and leveraging commodity data centers for the remaining sites.

This strategic use of commodity data centers significantly lowers both the initial capital expenditure and ongoing operational costs. Furthermore, it reduces the administrative burden associated with site management and maintenance. By integrating these cost-effective data centers into the Spire architecture, we achieve a balance between advanced security requirements and practical economic considerations.

2.4 Complementary Intrusion-Handling Approaches

In securing SCADA systems, several complementary approaches have emerged, each addressing different aspects of system vulnerabilities. These techniques include firewalls, resilient network switches, trusted hardware, and digital twins, among others.

Intrusion-tolerant firewalls and network switches are commonly used to reduce the attack surface and prevent unauthorized access to SCADA systems. Technologies like CRUTIAL Information Switches and SieveQ firewalls incorporate features like proactive recovery, replication, and filtering, which are designed to block malicious traffic before it can reach critical system components [57, 58]. These solutions excel at defending against external threats and network-based attacks, such as Distributed Denial of Service (DDoS) and routing attacks. However, they fall short in addressing threats that occur once an adversary gains access to the internal network or exploits vulnerabilities within SCADA-specific protocols. If the firewall or network switch is breached, they can no longer prevent the attacker from infiltrating the system or compromising its operational integrity.

Trusted hardware solutions, such as Hardware Security Modules (HSMs) and Trusted Platform Modules (TPMs), offer physical security for critical system components by safeguarding sensitive data and cryptographic keys from unauthorized access [59, 60]. These technologies are particularly effective in defending against external tampering and attacks that target system configurations or data integrity. However, they cannot mitigate the risks associated with insider attacks that can be launched from compromised components or advanced network attacks.

In recent years, the use of digital twin, virtual replica of physical SCADA system component, has gained traction as an innovative method for improving system resilience. A digital twin continuously simulates the operations of the physical SCADA component in real-time, enabling system operators to monitor for anomalies and deviations from expected behaviors. By detecting unusual patterns, such as sudden changes in system states or performance, digital twins can help identify potential attacks or failures early on [61]. However, while digital twins provide valuable insights into system performance and can detect cyber-physical attacks, they are not immune to the same vulnerabilities that affect the physical SCADA systems they represent and might have more potential attack vectors. Also, once compromised, even detection capabilities could be bypassed.

However, while each approach offers valuable defense, none of them can fully address the complex, multi-faceted threat models faced by modern SCADA infrastructures, especially when considering evolving sophisticated cyber-attacks at both system and network levels considered in our comprehensive threat model.

Chapter 3

Model

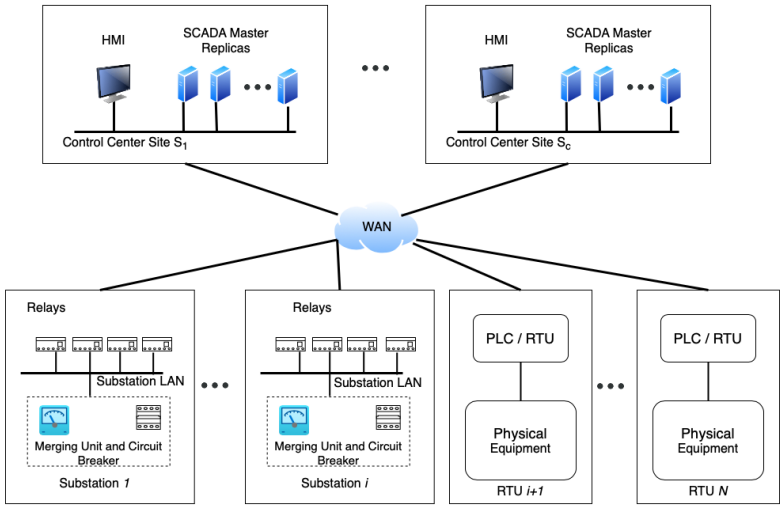


Figure 3.1: Grid SCADA Overview

Grid Supervisory Control and Data Acquisition (SCADA) systems are designed for monitoring and controlling the power grid. Substations play a key role by providing real-time protection and control of grid assets, with relays or intelligent electronic devices (IEDs) at their core. These devices continuously assess the grid's state using data from Merging Units (MUs) and protect assets during faults by triggering circuit breakers. Any malfunction in the relays can result in significant damage to grid assets such as transformers and threaten overall grid stability.

The control center, on the other hand, is responsible for overarching grid management, utilizing real-time monitoring and control capabilities. It consolidates data from various substations and field devices controlled and monitored through programmable logic controllers (PLCs) and remote terminal units (RTUs) to provide a central point for grid operation and control. The key component in a control cen-

ter is the SCADA Master, as any fault or compromise in it can lead to widespread consequences across the entire grid.

3.1 System and Network Model

Our comprehensive system model encompasses the power grid SCADA infrastructure, spanning from substations and various field devices to control centers. The field devices include programmable logic controllers (PLCs), remote terminal units (RTUs), and intelligent electronic devices (IEDs) that control and monitor grid assets such as circuit breakers, transformers, and transmission lines.

System Model: Each substation in our model is equipped with multiple protective relays. The number of relays in each substation is denoted by N_s. In context of substations, we specifically focus on the relay-based real-time autonomous protection operations, as they are the most critical and demanding operations within substations. To enable functionalities needed for our solution, each relay is equipped with a harness. A relay and its harness are referred to as a relay node.

The system also includes multiple control center sites. The number of control center sites is denoted by S_c. These sites can be dedicated control centers or a combination of control centers and data centers, as discussed in [51]. Distributed across these sites, there are N_c SCADA Master replicas, with each site hosting a portion of these replicas. These SCADA Master replicas enable real-time monitoring and control of any number of substations, PLCs and RTUs.

Review of Substation Protective Operations: During grid faults, current levels can surge, risking damage to grid assets within just a few power cycles. A rapid response is essential to mitigate this risk and ensure grid stability. Consequently, industry standards from organizations such as IEEE and IEC dictate that relays must detect a fault in a quarter of a power cycle, followed by an additional quarter of a power cycle for transfer time. Transfer time, as defined by IEC 61850, refers to the duration required for complete message transmission between sender and receiver, including necessary processing at both ends [8]. This defined transfer time establishes the latency budget for substation solutions, making it one of the most demanding operational requirements, typically around 4.167 ms in a 60 Hz system. We will refer to these time intervals as the *detection quarter* (the time allocated for fault detection) and the *reaction quarter* (the time allocated for message transfer).

Network Model: A local area network (LAN) connects the SCADA components within a substation or a control center. The substations, PLCs and RTUs are interconnected through a wide area network (WAN), which also connects control centers and data centers, enabling communication and data exchange across the entire SCADA system.

3.2 Assumptions

Grid Clock Synchronization Assumptions: Power grid standards mandate clock synchronization for distributed systems within and between substations to ensure correct operation of power devices [62]. This requirement is essential for various devices, including IEDs, Merging Units (MUs) at the bay and process levels, and Phasor Measurement Units (PMUs) used in Wide Area Monitoring, Protection, and Control (WAMPAC). These devices require synchronization accuracy of one microsecond or less [8, 63].

In practice, achieving this level of synchronization is feasible through standard time synchronization protocols such as IEEE 1588 Precision Time Protocol (PTP) [64]. We assume that the power devices deployed in the system adhere to these synchronization standards.

However, the work presented in this thesis does not require such stringent synchronization accuracy. Instead, we require that relay nodes in a substation are synchronized to within one millisecond of each other. This requirement is substantially less stringent, by at least three orders of magnitude compared to the grid standards.

Cryptographic Assumptions: We assume the attacker cannot break or subvert the cryptographic mechanisms used by our protocols i.e., attackers cannot break encryption, forge digital signatures, or find collisions in secure hash functions.

We also assume that each relay node and SCADA Master replica is equipped with a trusted hardware component, such as a Trusted Platform Module (TPM). This trusted hardware component has an associated public and private key. Additionally, we assume that accessing the private key requires physical access to the device and we presume that attackers lack physical access to the devices.

Network Assumptions: We assume that Local Area Networks (LANs) facilitate uninterrupted communication among correct SCADA components in the same LAN, even in the presence of faulty components that launch attacks such as resource exhaustion attacks. This resilience is made possible by the capabilities of modern switches.

For the Wide Area Network (WAN), we assume uninterrupted communication among correct sites. This assumption is supported by an intrusion-tolerant network service, building on the foundations established by Obenshain et al. [65] and Babay et al. [66]. This service combines resilient networking architecture with an intrusion-tolerant overlay, effectively addressing malicious attacks and compromises in both the underlying infrastructure and the overlay itself. It connects overlay nodes through multiple ISPs and is deployed on standard IP networks, allowing it to withstand the complete failure of any underlying network. By not relying on conventional Internet routing, it can quickly navigate around and mitigate malicious attacks and disruptions in the routing infrastructure.

We further assume that the WAN communication meets the specific timeliness requirement associated with Spire, namely the bounded-variance property of Prime [30],

which requires that for each pair of correct SCADA Master replicas, the network latency does not vary by more than a factor K_{Lat} [51].

While the deployment of this intrusion-tolerant network service is integral to our design and implementation, we present its existence as an assumption for the purposes of our analysis of our higher level protocols. The overall system fully meets this assumption in practice using the Spines intrusion-tolerant network service [67].

3.3 Fault and Threat Model

We adopt a Byzantine fault model at both the system and network levels. The Byzantine fault model represents arbitrary component failures, including scenarios where a sophisticated attacker compromises and gains full control of the component.

System-level Byzantine Faults: The key SCADA components - relay nodes and SCADA Master replicas are either correct or faulty. Correct components execute protocol specifications faithfully. In contrast, the faulty components may exhibit arbitrary behavior ranging from malfunctioning to deliberately causing harm. We assume a strong attacker model, where faulty components can coordinate but they cannot break cryptographic assumptions. Furthermore, we assume that attackers lack physical access to the machines.

Our system has N_s relay nodes in each substation and N_c SCADA Master replicas overall. The system can withstand up to f_s Byzantine faults among the N_s relay nodes in each substation and up to f_c Byzantine SCADA Master faults simultaneously.

Network level attacks: Our network threat model considers a diverse range of attacks targeting connectivity between SCADA components across local area networks (LANs) and the wide area network (WAN). As outlined in the assumptions section above, to navigate this complex threat landscape, we rely on modern switches within LANs to ensure efficient traffic management and maintain availability. For the WAN, we deploy an intrusion-tolerant network service that safeguards against extensive network disruptions, counters malicious routing efforts, and significantly increases the difficulty of executing successful denial-of-service (DoS) attacks.

Nevertheless, SCADA systems remain attractive targets for nation-state adversaries, who may allocate significant resources to disrupt these critical infrastructures. With enough resources, it is possible to execute sophisticated denial-of-service attacks that can target a specific site and isolate it from the rest of the network, such as demonstrated by Coremelt [55] and Crossfire [56] attacks. Nonetheless, we believe that achieving a complete simultaneous failure of multiple ISP backbones—necessary to disconnect several sites—would be nearly impossible within the confines of our intrusion-tolerant network. Consequently, we assume that up to one site that hosts SCADA Master replicas may be isolated from the rest of the network.

Model Scope: Our threat model does not account for compromised Human-Machine Interfaces (HMIs), rogue operators, or faulty circuit breakers. These vulner-

abilities necessitate separate mitigation strategies. However, the solution developed in this thesis offers protection for all system components, including HMIs and circuit breakers, through a proxy-based approach.

In this work, proxies are directly connected to the components using a single physical wire and communicate over the intrusion-tolerant network with the rest of the system. Importantly, these proxies are backward compatible with existing components that utilize standard SCADA communication protocols. This compatibility allows us to isolate the use of these insecure standards to a single wire connection behind the secure proxies.

Furthermore, we assume that attackers lack physical access to system components, and cannot manipulate Sampled Values (IEC61850-9-2) in substations.

3.4 Service Properties

A clear understanding of relay operational logic is crucial for grasping the definitions and service properties discussed in this section. Protective relays operate on a logic that emphasizes eliminating false negatives (instances where a relay fails to detect a fault requiring a trip operation). Such failures can lead to significant damage of grid assets (e.g. transformers) and pose serious safety hazards. As a result, relays maintain strict thresholds to ensure all genuine threats are identified and addressed promptly.

However, this emphasis on eliminating false negatives means that a small level of false positives (where the relay mistakenly identifies normal conditions as faults) may need to be tolerated. This trade-off is necessary to ensure that no false negatives occur. Additionally, false positives can stem from the conversion of analog measurements into digital signals, influenced by factors like time discretization, sampling intervals, noise, and fluctuations. By allowing for a small level of false positives, relays can establish a reliable fault detection threshold that responds to borderline conditions, potentially indicating an impending fault.

This balance between sensitivity and specificity can lead to variations in whether, and exactly when, each relay trips in borderline conditions. Different relays (e.g., with different hardware or software versions, or from different manufacturers) may have slightly different configurations or thresholds, resulting in different relay behaviors in borderline conditions.

Figure 3.2, illustrates the fundamental logic of false negative elimination through thresholds and epsilon margins. In Figure 3.2, the safety threshold (G_{ts} line) represents the threshold at which a protective action must be triggered. To eliminate false negatives, we define a small margin epsilon (ϵ), below the safety threshold, as an acceptable margin for triggering the protective action. This ϵ margin effectively "tightens" the safety threshold ($G_{ts} - \epsilon$). Note that protective actions may be trig-

gered slightly before the safety threshold is crossed. This principle of using an epsilon margin applies broadly to any system where analog data is sampled and converted into digital signals for processing and decision-making. Whether it's in power systems, healthcare monitoring, or industrial applications, this logic ensures that all critical events are detected.

In the context of power systems, protection schemes must ensure that the grid operating conditions remain within a safe operational range, across multiple dimensions such as current, voltage and frequency. Each dimension typically has two safety thresholds, one on each side, as illustrated in Figure 3.3. For example, the U.S. electric grid transmits and distributes electricity at a nominal frequency of 60 Hz. If the grid frequency deviates significantly from this value, such as dropping below 59.5 Hz or exceeding 60.5 Hz in the Eastern Interconnection, protective actions must be triggered to prevent asset damage and grid instability [68–70]. The fundamental logic of false negative elimination with epsilon margin applies to the safety thresholds on both sides. By applying an epsilon margin on both sides, the system triggers protective actions when it measures frequency values slightly below or above the defined critical limits. We define the operational range between the two safety thresholds that is further restricted by the two ϵ margins on both sides as the *Safe Envelope*, as depicted in Figure 3.3.

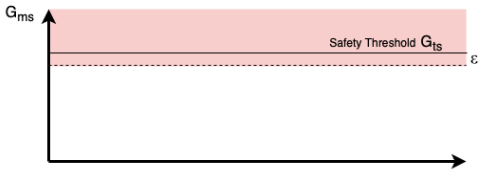


Figure 3.2: Fundamental concept of false negative elimination with safety threshold (G_{ts}) and epsilon margin (ϵ). The event is detected (red region) if the measured state (G_{ms}) is at least or above $G_{ts} - \epsilon$

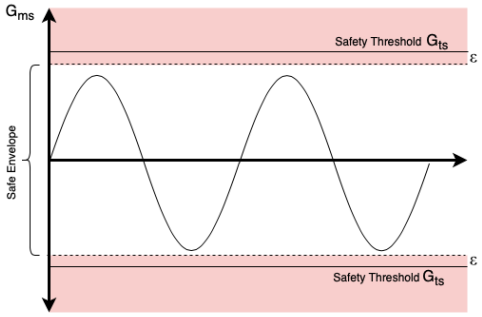


Figure 3.3: Power grid over current protection with two safety thresholds (G_{ts}) and two corresponding ϵ margins, defining the *Safe Envelope*. The relay will issue trip (red regions) if the grid's measured state G_{ms} falls outside of the safe envelope.

Definition 1. A *correct relay* performs as follows:

- If the grid state necessitates a protective action to protect grid assets, i.e., the grid state breaches the safety threshold, a correct relay must issue the required protective action within a quarter of a cycle (the detection quarter).

- If a correct relay issues a protective action, then within the last quarter of a cycle (the detection quarter), that relay measured the grid state breaching the safe envelope.

Definition 2. A *correct relay node* consists of a correct relay and a correct harness. A correct harness executes protocol specifications faithfully and in a timely manner, ensuring adherence to the reaction quarter requirement.

Definition 3. A *correct SCADA Master replica* executes protocol specifications faithfully and in a timely manner.

The system is designed to enable end-to-end Byzantine-resilient grid operations. Formally, the following guarantees apply, based on the assumptions outlined above:

- **Substation Safety:** If a valid protective operation is executed by the breaker node, at least one correct relay node issued that action within the last reaction quarter.
- **Substation Real-Time Reaction:** If the grid state necessitates a protective action due to an event, that action will be issued for that event's reaction quarter[1].
- **SCADA Safety:** If two correct SCADA Master replicas execute the i^{th} operation, then those operations are identical.
- **SCADA Bounded-Latency Reaction:** If a correct system component initiates an operation, the latency for that operation to be executed by a correct SCADA Master replica is upper bounded.

Additionally, any practical solution must meet the following domain requirements:

Economic Cost: The proposed solution must be economically feasible. Given the vastness of the grid, which includes numerous substations each with multiple relay-based protection schemes, the cost of large-scale deployment is a major consideration. The design should focus on minimizing costs while preserving system resilience, as cost efficiency will directly affect the feasibility of widespread deployment.

Seamless Grid Integration: The proposed solution must seamlessly integrate into existing grid infrastructure, ensuring compatibility with various grid topologies, SCADA protocols, and legacy systems. This will help avoid expensive upgrades or replacements and facilitate a more straightforward transition.

Long System Lifetime with Continuous Availability: The proposed solution must guarantee uninterrupted operation over an extended period. It should also account for routine equipment maintenance and incorporate robust fault recovery mechanisms to ensure continuous availability.

[1]The guarantee that the action will be issued for the reaction quarter is augmented by an empirical evaluation demonstrating that the issuance and processing of that action will complete by the real-time deadline in relevant environments (see Chapters 4.5 and 4.6 for more details)

Chapter 4

Real-Time Byzantine Resilience for the Substation

Protective relays are crucial for ensuring the stability and safety of the power grid. They continuously monitor the grid through key electrical parameters such as current, voltage and phase angle. The measurements are multicasted by Merging Units (MUs) over the process bus in Sampled Value (SV) messages. Protective relays process the measurements to detect situations that put the grid and its customers at risk (i.e., when measured grid state breaches safe envelope as explained in Chapter 3). Relays, upon detecting such a risky situation, issue IEC61850 GOOSE messages (Generic Object Oriented Substation Event) to trip the circuit breaker, cut the power flow, and protect the grid assets and connected devices. Once the issue that caused the trip is resolved, an operator usually instructs the relay to issue a GOOSE message to close the circuit breaker.

All protection schemes are time-critical, i.e., the reaction time needed is very exact to effectively protect the grid asset. During grid faults, current levels can surge, risking damage to grid assets within just a few power cycles. A rapid response is essential to mitigate this risk and ensure grid stability. Consequently, industry standards from organizations such as IEEE and IEC dictate that relays must detect a fault in a quarter of a power cycle, followed by an additional quarter of a power cycle for transfer time. Transfer time, as defined by IEC 61850, refers to the duration required for complete message transmission between sender and receiver, including necessary processing at both ends [8]. This defined transfer time establishes the latency budget for substation solutions, making it one of the most demanding operational requirements, typically around 4.167 ms in a 60 Hz system. As defined in Chapter 3, we refer to these time intervals as the *detection quarter* (the time allocated for fault detection) and the *reaction quarter* (the time allocated for message transfer).

Due to the critical role of the protective relays and the very tight reaction time requirements, operators may deploy multiple relays for the same protection function, each with unilateral power to trip or close a breaker. However, existing state-of-the-

art protective relay technology is susceptible to intrusions. A protective relay with unilateral power under the control of an attacker presents two problems: First, a relay that does not trip when it should, can cause irreparable damage to the grid and its connected customers. Second, a relay that does unnecessarily trip causes a significant disruption to many customers. Either case is costly and highly undesirable.

Constructing a Byzantine resilient protective relay for the substation is challenging due to several rigid factors:

- The standard [8] requires reaction time to be within a quarter of a power cycle. Specifically, in a 60Hz system (e.g., in North America), a power cycle amounts to 16.67 milliseconds (ms), and a quarter cycle amounts to 4.167ms. This exact real-time constraint on the responsiveness of the protective scheme is extremely demanding in the face of a successful attack.

- A Byzantine resilient protective relay function has to be continuously available in the field over a long lifetime (years). Therefore, proactive recovery is necessary to periodically, automatically and seamlessly regain control of compromised relays. This prevents the attacker from gaining long term access to relays and limits the attacker's ability to compromise a critical number of relays.

- The protective relay is a relatively expensive device, and there are many of them across grid substations. Hence, to reduce cost there is a high incentive to limit the additional relays needed in a Byzantine resilient scheme.

- Any practical Byzantine resilient scheme has to seamlessly integrate into existing substations without significant changes to other substation components or overall substation architecture, in order to facilitate actual deployment.

The first intuition to address Byzantine faults in a substation would be to tune and adapt classic state machine replication-based protocols (BFT SMR). However, in our experience, it is incredibly challenging to meet all the substation requirements with state machine replication-based protocols as the environment is fundamentally different and operates at much finer time scales. For example, achieving the quarter-power cycle requirement in all operating conditions, including during successful intrusions and simultaneous network attacks, is highly challenging.

Our key insight is that the circuit breaker state in a relay-based protection function only depends on the most recent relay action (trip or close). This means that the total ordering provided by BFT SMR protocols is not strictly necessary to construct a Byzantine resilient system. Moreover, with BFT SMR, the minimum number of relays needed to tolerate f_s Byzantine relays with k_s relays undergoing proactive recovery simultaneously would be $3f_s + 2k_s + 1$ [22]. This means a BFT SMR-based solution needs at least six relays to practically tolerate a single intrusion. Since a protective relay is a fairly expensive device (tens of thousands of dollars), BFT SMR-based schemes will be costly in practice, reducing the feasibility of broad adoption.

We look at the environment in the grid substation and the specific need of a Byzantine resilient protective relay. We design an architecture and protocols to ensure protection functions work correctly even in the presence of successful intrusions and network attacks.

This chapter presents the first real-time Byzantine resilient architecture and protocols for the substation that simultaneously address system compromises and network attacks while meeting the strict timeliness requirement (4.167ms). The architecture can tolerate f_s Byzantine relays and k_s relays undergoing proactive recovery simultaneously with just $2f_s + k_s + 1$ total relays. This is in contrast to a traditional BFT SMR-based solution that would need $3f_s + 2k_s + 1$ total relays. The architecture has long system lifetime and seamlessly integrates into existing substations without the need to modify existing components. Parts of this chapter have been previously published in [71, 72].

4.1 Architecture

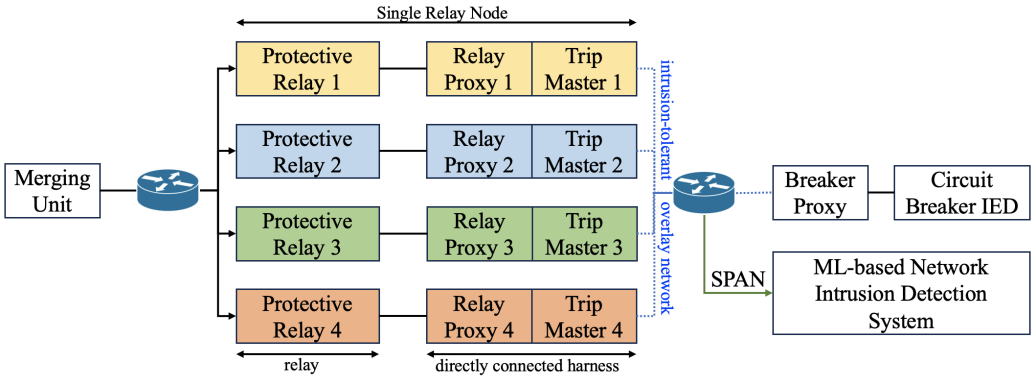


Figure 4.1: An architecture that tolerates one Byzantine relay node ($f = 1$) and one relay node undergoing proactive recovery ($k = 1$) simultaneously with a total of four relay nodes in an IEC61850 Substation.

A practical Byzantine resilient solution must be transparent to both the protective relays and the circuit breakers. Furthermore, such a solution must ensure that all other existing substation components are entirely unaware of the new Byzantine resilience capabilities and continue to function unchanged. Therefore, we design our solution such that each protective relay works as if they were directly connected to the breaker and using the same (unprotected) protocols to issue their trip and close action commands. Similarly, the circuit breaker operates unchanged. To facilitate this mode of operation, our harness running Byzantine resilient protocols is directly attached to the protective relays on one side and the circuit breaker on the other side, as shown in Figure 4.1. Our architecture in Figure 4.1 shows that each relay and its

directly connected harness (Relay Proxy and Trip Master) form a single logical entity called a relay node. Similarly, the circuit breaker and its directly attached harness form a single logical entity called a breaker node. The proxies limit the use of insecure communication protocols, namely the GOOSE messages in IEC61850, so that these messages are only used on a direct wire between the protective relay and its proxy and the circuit breaker and its proxy (i.e, they never flow on any of the networks). The modular design of the architecture allows the solution to be adapted to other SCADA protocols by employing suitable proxies.

All network communication in the system, between relay nodes and other relay nodes or the breaker node, is conducted over an intrusion-tolerant overlay network, Spines [66]. This intrusion-tolerant network design addresses network-level threats in the substation's Local Area Network. This design is complemented by a machine-learning-based Network Intrusion Detection System to detect suspicious and anomalous traffic, providing situational awareness [73]. This further precludes long running denial of service resource consumption attacks.

The Byzantine resilient architecture can support two protocols: Arbiter Protocol and Peer Protocol. The overview and details of protocols are provided in Chapters 4.2 and 4.3.

4.2 Arbiter Protocol

To prevent f_s Byzantine relays from controlling the circuit breaker, we need the circuit breaker to act only if $f_s + 1$ relays issue the same action at about the same time. Thus, the total number of relays required to tolerate f_s Byzantine relays will minimally be $2f_s+1$. However, we need to support the diversity of attack surface and proactive recovery [22] . Therefore, to support k_s simultaneous relays going through proactive recovery, a total of $2f_s + k_s + 1$ relays are needed [57].

A simplistic solution could be to just have each relay node sign an action (i.e., trip or close) and send the action to the breaker node. Suppose the breaker node receives verified messages from $f_s + 1$ distinct relays with the same action and acts on it. In that case, an adversary that recorded these messages could replay them to change the breaker state in the future. Hence, we need to verify the freshness of relay messages. One way to ensure the freshness of the messages is to have the breaker node issue a nonce that is valid for a short time period. The relay nodes would then include this nonce along with the action in their signed message, and the breaker node would verify the freshness of the nonce when counting the $f_s + 1$ messages. In such a solution, the breaker node could issue nonces periodically or on-demand by relay nodes. Both of these approaches incur additional costs compared to simply generating a signed message with no nonce. While the former needs the breaker node to generate more messages periodically, the latter would add latency, requiring an additional round before generating a signed message. Either case would render the

breaker node logic more complex. Furthermore, scaling would pose a challenge since these relay nodes can operate more than one circuit breaker. It would result in either additional messages or added latency, each of which is undesirable.

Fortunately, we can eliminate the need for an explicit nonce generation. In our Arbiter Protocol , we use the current time as part of the signed message sent by the relay node. The breaker node will verify freshness by ensuring the message is within a short period from its current clock time. The substation standards require time synchronization between substation components to be in the order of sub-microsecond. While we do not need such fine-grain synchronization, we still need to ensure the responsiveness to within the quarter cycle (4.167ms). Therefore, we chose our freshness period to be one millisecond, which is three orders of magnitude coarser than the substation requirements (hence adds a very weak assumption) but still allows our solution to meet the quarter cycle guarantee without the overhead associated with the explicit nonce techniques.

4.2.1 Protocol Description

Let the breaker be closed. Upon detection of a risky situation, a correct relay issues a trip message. The connected Trip Master (in the relay node) receives this trip action, generates a signed trip message with its local time, and sends it to the breaker node. The breaker node verifies signature and freshness (i.e., the time in the message is within one millisecond of its current time). Upon receipt of $f + 1$ distinct fresh relay node trip messages, the breaker node trips the circuit breaker. After the circuit breaker trips, the breaker node multicasts the trip acknowledgment to the relay nodes.

Relay nodes continue to generate fresh signed trip messages (with updated time) every one millisecond as long as their relay remains in a trip state and they have not received the trip acknowledgment message.

When the risky situation is resolved, a correct relay issues a close message. The corresponding relay node generates a signed close message with its local time and sends it to the breaker node. Similar to the tripping logic above, upon receiving fresh close messages from $f_s + 1$ distinct relay nodes, the breaker node closes the circuit breaker. After the circuit breaker closes, the breaker node multicasts the close acknowledgment to the relay nodes.

Relay nodes continue to generate fresh signed close messages (with updated time) every one millisecond as long as their relay remains in a close state and they have not received the close acknowledgment message.

As described above, the breaker node in the Arbiter Protocol is complex with sophisticated logic. This increases attack surface across the substation and presents integration challenges as the system scales. In Section 4.3 we present the Peer Protocol addressing these important concerns.

4.3 Peer Protocol

In order to facilitate seamless integration while reducing the attack surface of the breaker node (which is a single point of failure), the breaker node should not have a role beyond authenticating the action message and executing the action. This can be achieved by moving the logic of the breaker node in the Arbiter Protocol into a distributed Byzantine resilient protocol in the Trip Master. This protocol coordinates the relay nodes and uses threshold signatures to ensure the support of $f_s + 1$ relay nodes. With this support, relay nodes can generate a single threshold-signed action message to the breaker node. We present the details of this Peer Protocol below.

4.3.1 Architecture Abstraction through Threshold Cryptography and Time Discretization

The relay nodes make use of a threshold signature scheme. At a high level, an $(f_s + 1,\ N_s)$ threshold signature scheme creates a public key and a signing key, just like a typical signature scheme. However, N_s parties split the signing key, and $f_s + 1$ of these parties need to collaborate to create a signature. Most threshold signature schemes (including the one used in this protocol) use a two-step process to create these signatures. First, each party can create partial signatures on the same message (generating shares). Second, if any distinct set of at least $f_s + 1$ of these valid shares exists on the same message, we can form the threshold signature by combining them. The main challenge with this process is that shares should be generated on the exact same message. One way to do this would be to have relay nodes agree on a message. However, this additional coordination would require at least an extra round of communication, which is prohibitively expensive given the stringent latency requirement.

To reduce latency, our solution is to have the relays independently generate shares on the same message without coordination. This means that only one round of communication is needed to collect the shares and generate a threshold-signed message for the breaker node. However, this message still needs to carry a nonce (like time in the Arbiter Protocol above) that can prove freshness. The raw local time can not be used as a nonce to create the same message on different relay nodes as the local times of the relay nodes would vary and messages would not be identical. Instead, we can discretize time to closest millisecond and use the discretized time as the nonce, henceforth called the Discretized Time Stamp (DTS).

Given our assumption that the nodes are synchronized to within 1ms of each other, two messages created at the same time on different nodes will always have the same or consecutive DTS. To see why these can be consecutive, we define a *DTS interval* as the set of times that discretize to the same DTS. The difference in time synchronization can cause one node's local time to be in one DTS interval, and the

other node's local time to be in the next DTS interval. In fact, any arbitrarily small synchronization difference can cause this, if the local times are close enough to the endpoints of the intervals.

Now, let us describe how the DTSs are used in the protocol. If a relay node decides on an action, it will generate a share with the action and the current DTS. Relay nodes exchange these shares, and if there are valid shares on the same message (i.e., the same action and same DTS) from $f_s + 1$ relay nodes, they will generate a threshold signature message and send it to the circuit breaker node. Upon receiving the threshold signature message, the circuit breaker then verifies the signature and checks freshness by comparing the DTS on the message to its local DTS.

However, when the DTS of two different relay nodes are different, the nodes would not be able to combine their shares into a threshold signature. To address this, the relay node generates two shares: one for the current DTS and one for the next DTS. The relay node then sends both of these shares in the same message. Therefore, if two correct relay nodes simultaneously decide on the same action, there will be at least one share with the same DTS because of the clock synchronization assumption. Therefore, only one round of communication between the relay nodes is needed to threshold sign the action.

4.3.2 State Machine

The above description of the use of Threshold Cryptography and Discretized Time Stamps covers how relay nodes act when they need to generate a threshold-signed message (i.e., a trip or a close action). To completely specify the behavior of the Peer Protocol , we construct a state machine that handles the messages from the local relay, the other relay nodes, and the breaker node. This forms a single logical real-time Byzantine-resilient relay node out of multiple intrusion-prone real-time relay nodes.

For the ease of understanding, we describe the state machine in two steps: a partial state machine in Fig. 4.2 and a complete state machine in Fig. 4.3. The partial state machine presents the coordination algorithm during fault-free operation while assuming all correct relays issue an action at the same time. The complete state machine adds support for non-identical relays, benign and Byzantine faults, and proactive recovery.

The system state for each relay node is defined based on the relay state (denoted by r) and the breaker state (denoted by b). When the breaker node starts, it acquires the circuit breaker state and notifies the relay nodes of that state. The relay node updates its local relay state when the local relay (the one directly connected to that node) issues an action. In every relay node, the relay state and the breaker state each carry two essential pieces of information: the state (trip or close) and the DTS of the event occurrence (denoted by $r.t$ and $b.t$). For example, in Fig. 4.2, the *Closed*

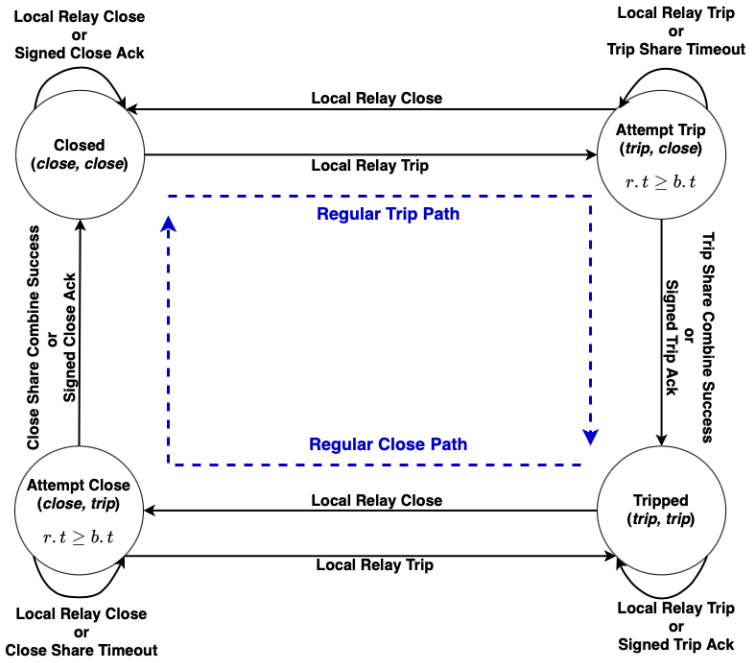


Figure 4.2: Peer Protocol partial state machine

state in the top left corner has a relay state close (i.e., the local relay issued a close action), and the breaker state is close. Another example in Fig. 4.2, in *Attempt Trip* state, the local relay issued a trip and the breaker is closed. Note, when a relay node starts or recovers after proactive recovery, it acquires b and r, then uses their state and DTS data to transition into appropriate state in the state machine and resumes operation. However, to keep the state machine visualization simple, in Fig. 4.2 and Fig. 4.3 we do not show recovery and transitions to appropriate state.

Let us assume that the relay node is in the *Closed* state (top left corner in Fig. 4.2). If a correct local relay detects a risky situation, it will issue a local relay trip message ($r.t \geq b.t$). The relay node attempts to trip the breaker by generating and sending a trip share message to the other relay nodes with current and next DTS shares. It then transitions to the *Attempt Trip* state. While in that state, it receives shares from other relay nodes and keeps generating trip shares every DTS interval. Upon successfully combining shares and generating a valid threshold-signed trip message, the relay node sends that message to the breaker node and transitions to the *Tripped* state.

In the *Tripped* state, the relay node periodically sends the threshold-signed trip message to the breaker node until it receives a valid trip acknowledgment signed by the breaker node. A relay node will remain in the *Attempt Trip* state until either it can generate a valid threshold-signed trip message (by combining other relays shares) or until it receives a valid trip acknowledgment message from the circuit breaker node.

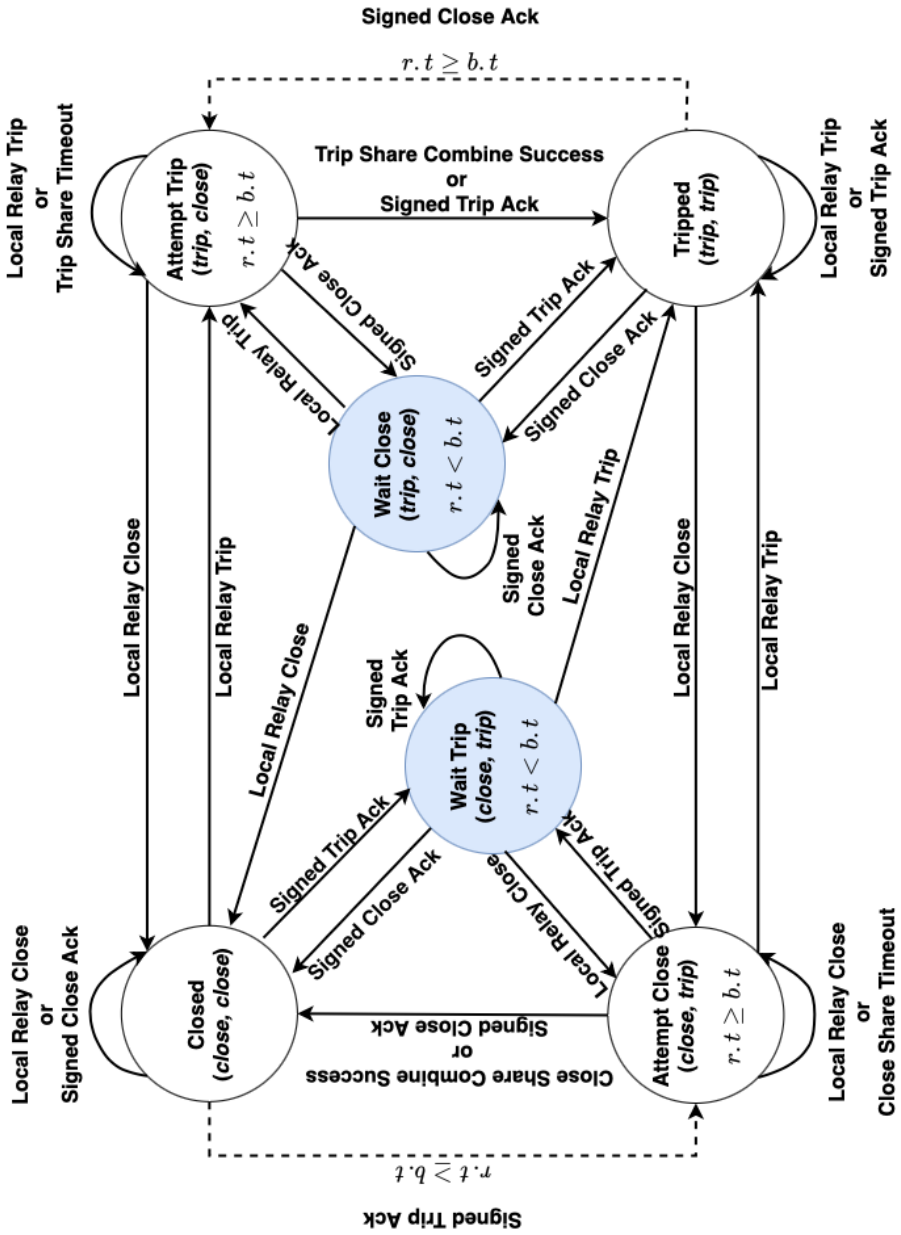


Figure 4.3: Peer Protocol full state machine

The Regular Trip Path in Fig. 4.2 marks this path where relay nodes coordinate to trip a closed breaker.

A similar flow occurs when closing the breaker after the risky situation is resolved. A relay node in the *Tripped* state receives a close message ($r.t \geq b.t$) from its local relay. The relay node then attempts to close the breaker by generating and sending a close share message to the other relay nodes, with current and next DTS shares. It then transitions to *Attempt Close* state. While in that state, it receives shares from other relay nodes and keeps generating close shares every DTS interval. Upon successfully combining shares, generating a valid threshold-signed close message, the relay node sends that message to the breaker node and transitions to the *Closed* state.

In the *Closed* state, the relay node periodically sends the threshold-signed close message to the breaker node until it receives a valid close acknowledgment signed by the breaker node. A relay node will remain in the *Attempt Close* state until either it can generate a valid threshold-signed close message (by combining other relays shares) or until it receives a valid close acknowledgment message from the breaker node. The Regular Close Path in Fig. 4.2 marks this path where relay nodes coordinate to close a tripped breaker.

Unfortunately, relays may not be identical, i.e., correct relays may not trip or close simultaneously (at the exact same time) due to relay diversity (different manufacturers or versions). The partial state machine (Fig. 4.2) does not support a correct relay node that might trip or close relatively slower than other correct relay nodes. For example, let us assume all correct relay nodes are in *Closed* State and two relay nodes are fast and use the Regular Trip Path to transition from the *Closed* to the *Tripped* state. Another correct but slightly slower relay node remains in the *Closed* state (has not yet received a trip message from its local relay) but then receives a valid signed trip acknowledgement. It cannot follow the Regular Trip Path. Hence, to support non-identical relays, we add two wait states (*Wait Trip* and *Wait Close*), the shaded states in the complete state machine (Fig. 4.3). The relay nodes wait in these states for their local relay to catch up to their faster peers. In our example, the relay node can transition from the *Closed* state to the *Wait Trip* state. If the local relay is correct, it will eventually issue a local relay trip message and the relay node will transition from the *Wait Trip* state to the *Tripped* state. Similarly, a correct but slower relay node in the *Tripped* state can move to the *Wait Close* state on receiving a valid close acknowledgment before receiving its local relay close message. When its local relay issues a close message, the relay node transitions from the *Wait Close* state to the *Closed* state.

Furthermore, as discussed in the threat model above, in addition to a relay being delayed, it may fail and be unresponsive. For example, consider the scenario where Relay Node 1 is in the *Closed* state, and its local relay stops issuing any messages. There is a risky situation in which two other relay nodes (say Relay Node 2 and Relay Node 3) decide to trip and transition along the Regular Trip Path. Relay Node 1 will receive a valid trip acknowledgment message from the breaker node and will

transition to the *Wait Trip* state. When the risky situation is resolved, the other tripped relay nodes close the breaker, transitioning along the Regular Close Path. Upon receiving a valid close acknowledgment message, Relay Node 1 transitions from the *Wait Trip* to the *Closed* state. Until the local relay node resumes operation, the relay node toggles between *Closed* and *Wait Trip* states or *Tripped* and *Wait Close* states. Thus, the state machine addresses benign local relay faults.

Finally, a relay node under the control of an intruder can behave arbitrarily, i.e., exhibit Byzantine faults. For example, it can trip unnecessarily or not trip when needed. The Byzantine relay node can generate its own shares but cannot generate the threshold-signed trip message by itself ($f_s + 1$ valid shares are needed to generate the threshold-signed message). Hence, it cannot unilaterally trip the closed breaker or close the tripped breaker. The formal guarantees are elaborated in the proof sketch subsection.

4.4 Diversity and Proactive Recovery

In addition to the exact real-time latency requirement, a solution has to support a long system lifetime with continuous availability. The Byzantine resilient protocol can guarantee correctness as long as the compromised relay nodes do not exceed the design threshold of f_s. To enable the system to continue to work correctly as long as no more than a certain fraction of the relay nodes are compromised, we adapt from existing works the techniques of diversity and proactive recovery [17, 74]. Proactive Recovery entails forcefully restarting the relay node from a clean state, diversifying its attack surface, and refreshing cryptographic keys, before resuming operation.

A diverse attack surface in our setup requires diverse relay nodes, meaning both the protective relays and the harness need to be diverse. To accomplish that, we diversify the harness using standard approaches such as automatic software diversity at either compilation or run time [14–16]. Other useful techniques in that space include operating system diversity [13] and (the more expensive) N-version programming [11, 12]. A good approach to achieve attack surface diversity for the protective relays may be to procure them from different manufacturers (e.g. GE, Siemens, Hitachi Energy). Alternatively, the aforementioned diversity techniques could be employed for protective relays from the same manufacturer.

To maintain availability in the presence of both f_s intrusions and k_s relay nodes undergoing proactive recovery simultaneously, a BFT SMR-based solution requires $3f_s + 2k_s + 1$ total nodes, out of which $2f_s + k_s + 1$ nodes are needed to make progress. In contrast, both Peer Protocol and our Arbiter Protocol support proactive recovery with only $2f_s + k_s + 1$ total nodes, out of which only $f_s + 1$ nodes are needed to make progress i.e., issue valid action command [22], [57].

Since we don't require update history like in BFT SMR, proactive recovery is expedited. Typically, a relay node undergoing proactive recovery can start participating

in the protocol within seconds.

4.5 Guarantees and Correctness Proof

Our system has a total of N_s relay nodes. It can withstand up to f_s Byzantine faulty relay nodes and k_s relay nodes undergoing proactive recovery simultaneously.

Recalling definitions of *correct relay* and *correct relay node* and service properties from Chapter 3:

Definition 1. A *correct relay* performs as follows:

- If the grid state necessitates a protective action to protect grid assets, i.e., the grid state breaches the safety threshold, a correct relay must issue the required protective action within a quarter of a cycle (the detection quarter).

- If a correct relay issues a protective action, then within the last quarter of a cycle (the detection quarter), that relay measured the grid state breaching the safe envelope.

Definition 2. A *correct relay node* consists of a correct relay and a correct harness. A correct harness executes protocol specifications faithfully and in a timely manner, ensuring adherence to the reaction quarter requirement.

The system guarantees the following:

- **Substation Safety:** If a valid protective operation is executed by the breaker node, then at least one correct relay node issued that action within the last reaction quarter.
- **Substation Real-time Reaction:** If the grid state necessitates a protective action due to an event, that action will be issued for that event's reaction quarter.

We can fully prove Safety, and for real-time reaction, we can fully prove the issuance of required protective action by relay nodes for that event. We demonstrate by rigorous evaluation through benchmarks in Chapter 4.6 the extent to which the real-time latency requirement is met under different operating conditions in different relevant environments.

4.5.1 Arbiter Protocol

Proof for Substation Safety: In the Arbiter protocol, the breaker node issues a protective action to the circuit breaker only if it receives valid, fresh protective action messages from $f_s + 1$ distinct relay nodes, each containing the same action and a time within one millisecond of the breaker node's local time. By the assumption of our

threat model, at most f_s relay nodes can be Byzantine faulty. Therefore, among the $f_s + 1$ valid and fresh messages, at least one must come from a correct relay node.

The time in the correct relay node's message is within one millisecond of the breaker node's local time. By the clock synchronization assumption, the correct relay node issued that message within one millisecond of breaker node's local time. Hence, the correct relay node issued the protective action within the last reaction quarter.

Proof for Substation Real-time Reaction: In the Arbiter Protocol, issuing the required protective action for the event's reaction quarter means $f_s + 1$ distinct relay nodes issue valid messages with that protective action for that event's reaction quarter.

We assume that the maximum number of relay nodes that may either be Byzantine faulty or undergoing proactive recovery simultaneously is $f_s + k_s$. Therefore, the minimum number of remaining correct relay nodes is: total relay nodes $(2f_s + k_s + 1)$ - maximum possible Byzantine relay nodes (f_s) - maximum possible relay nodes under proactive recovery (k_s) = minimum remaining correct relay nodes $(f_s + 1)$.

The correct relays of these $f_s + 1$ correct relay nodes will issue the required protective action within the detection quarter for an event. Upon receiving this correct relay message, the correct harnesses execute the protocol faithfully and in a timely manner. Hence, they will generate and issue a valid, fresh required protective action message for the event's reaction quarter.

Meeting the Real-time Reaction Deadline: The above proof proves that the protection action will be issued for that event. As the system is not synchronous or weakly synchronous, meeting a real-time deadline cannot be proven analytically. Instead, we demonstrate by rigorous evaluation through benchmarks in Chapter 4.6 that the Arbiter Protocol meets the real-time latency requirement (i.e., the breaker node receives and accepts the valid protective action messages from $f_s + 1$ distinct relay nodes within the reaction quarter) under all operation conditions in all relevant environments.

4.5.2 Peer Protocol

Proof for Substation Safety: In the Peer protocol, the breaker node issues a protective action to the circuit breaker only if it receives a valid threshold signed protective action message with a timestamp within one millisecond of the breaker node's timestamp. In our threshold-signature scheme, the threshold is $f_s + 1$, i.e., we need valid shares with same action and timestamp from $f_s + 1$ distinct relay nodes to generate a valid threshold signed protective action message. By the assumption of our threat model, at most f_s relay nodes can be Byzantine faulty. Therefore, there is a valid share from at least one correct relay node in the $f_s + 1$ shares that were combined to generate the valid threshold-signed protective action message.

The timestamp in the correct relay node's share is within one millisecond of the breaker node's local timestamp. By the clock synchronization assumption, the correct relay node issued that share within one millisecond of breaker node's local timestamp. Hence, the correct relay node issued the protective action within the last reaction quarter.

Proof for Substation Real-time Reaction: In the Peer protocol, issuing the required protective action for the event's reaction quarter means that a correct relay node issues a valid threshold-signed message with that protective action for that event's reaction quarter.

In our threshold-signature scheme, the threshold is f_s+1, i.e., we need valid shares with the same action and timestamp from f_s+1 distinct relay nodes to generate a valid threshold-signed protective action message. We assume that the maximum number of relay nodes that may either be Byzantine faulty or undergoing proactive recovery simultaneously is $f_s + k_s$. Therefore, the minimum number of remaining correct relay nodes are: total relay nodes $(2f_s + k_s + 1)$ - maximum possible Byzantine relay nodes (f_s) - maximum possible relay nodes under proactive recovery (k_s) = minimum remaining correct relay nodes $(f_s + 1)$, meeting the required threshold for generating valid threshold-signed message.

The correct relays of these f_s+1 correct relay nodes will issue the required protective action within the detection quarter for an event. Upon receiving this correct relay message, the correct harness executes the protocol faithfully and in a timely manner. Hence, they will generate shares, combine them and issue a valid threshold-signed message with the required protective action for the event's reaction quarter.

Meeting the Real-time Reaction Deadline: The above proof proves that the protection action will be issued for that event. As the system is not synchronous or weakly synchronous, meeting a real-time deadline cannot be proven analytically. Instead, we demonstrate by rigorous evaluation through benchmarks in Chapter 4.6 the extent to which the Peer Protocol meets the real-time latency requirement (i.e., the breaker node receives and accepts the valid threshold-signed protective action message within the reaction quarter) under different operating conditions in different relevant environments.

4.6 Evaluation

In this section, we present detailed performance benchmarks for system with the Arbiter Protocol (Chapter 4.2) and Peer Protocol (Chapter 4.3) demonstrating empirically the extent that they meet the strict real-time requirement. These benchmarks are conducted in multiple testbeds over a range of fault-free and faulty operating conditions and demonstrate the advantages and the tradeoffs of the protocols.

Performance Benchmarks Testbeds: We evaluate the system across four

testbeds, each chosen to represent different segments of the grid computing landscape. These testbeds reflect not only the current state of technology but also future trends in grid deployment environments. Below, we discuss the rationale behind each testbed selection and its relevance.

Low Performance Servers Testbed: This testbed is representative of the types of machines commonly found in current power grids, where hardware often lags behind the latest advancements. It is critical for assessing system performance in environments with resource-constrained legacy machines, which will remain in operation for the near term. By evaluating performance on these machines, we ensure compatibility with existing grid compute and demonstrate the system's ability to operate efficiently on lower-performance hardware, a scenario typical in real-world deployments.

The machines in this testbed are four core Intel(R) Xeon(R) E3-1271 CPU @ 3.60GHz with 16GB RAM running CentOS 8, and with 1Gbps network connections. This is the weakest of the three testbeds.

Modern Industrial NUCs Testbed: It is representative of newer, grid-ready machines based on Industrial NUCs. These devices are rugged, highly energy-efficient, compact, which can be used for immediate new grid deployment. By evaluating performance on these machines, we ensure compatibility with current-generation hardware that is poised to be deployed into grid now or within the next few years.

The machines in this testbed are six core Intel i7 NUCs @ 4.0GHz with 16GB RAM running CentOS 8, and with 1Gbps network connections.

Modern Low Cost Data Center Servers Testbed: Featuring the most powerful and scalable systems among the testbeds, this testbed includes contemporary low-cost data center servers capable of handling large-scale, high-performance computing tasks. While these systems cannot be deployed immediately, they represent the future direction of grid-based computing. This testbed allows us to explore the system's performance in a scenario where more powerful machines are available to meet the increasing computational demands of future grid infrastructures.

The machines in this testbed are four core Intel(R) Xeon(R) E-2274G CPU @ 4.00GHz with 16GB RAM running Alma Linux 9 , and with 1Gbps network connections. This is the strongest of the three testbeds.

Real-time Kernel Testbed: Given that low-performance servers are expected to dominate both current and near-future grid infrastructures, and considering their widespread use in power grid deployments, we explore an option that could enhance performance on the testbed: the use of a real-time kernel.

By evaluating the system's performance across multiple testbeds, from Low Performance Servers to high-performance data center hardware (Modern Low-Cost Data Center Servers), we assess its effectiveness across a spectrum of computing environments. This approach ensures our findings are relevant not only to current state-of-the-art systems but also to emerging technologies, particularly in the context of power grids. The performance benchmarks provide valuable insights into how the system can operate efficiently across different generations of hardware, anticipating

both current capabilities and future trends. As grid infrastructures evolve and more advanced hardware becomes available, our results will remain applicable, guiding the development and deployment of systems in an increasingly complex grid technological landscape.

Performance Benchmarks Setup: In all test beds we emulate relays with a process that sends GOOSE (action) messages according to the IEC61850 standard. This relay emulator runs on each relay node, alongside the Relay Proxy and Trip Master processes, and publishes GOOSE messages based on Sampled Values. To emulate these Sampled Values, another machine sends a standard IP multicast messages to all the relay nodes, which then triggers a trip or close GOOSE action on the respective relay emulators.

The system consists of four relay nodes and therefore can simultaneously tolerate up to one faulty relay node (fail-stop or Byzantine fault) and up to one additional relay node undergoing proactive recovery. The end-to-end latency is measured, starting when a Sampled Value is multicast and ending when the circuit breaker changes its state. The Sampled Values are multicast from the same machine running the circuit breaker node so that the clock readings are done using the same local clock. Each benchmark includes one million end-to-end measurements of the trip and the close actions, conducted over a period of about 24 hours.

Performance Benchmarks Scenarios: We evaluate the performance of the system in all five possible operational conditions, presented below:

Normal or Fault-Free: all four relay nodes are working correctly.

Fail-Stop Fault: one of the relay nodes is unavailable due to a fail-stop fault. Note that this situation is identical to having one of the relay nodes undergoing proactive recovery.

Fail-Stop Fault with Proactive Recovery: one of the relay nodes is unavailable due to a fail-stop fault while simultaneously, an additional relay node is undergoing proactive recovery. Effectively, only two correct relay nodes are available.

Byzantine Fault: one of the relay nodes is under the control of a sophisticated attacker. We programmed such a node to perform two simultaneous attacks for each action. First, the Byzantine relay node sends a corrupt share that will result in an unsuccessful combination, costing the system precious time and computational resources of the relay nodes. Second, the Byzantine relay node performs a short intermittent denial of service attack on the other relay nodes to consume their network and computational resources further. Note that the chapter 3 discusses why sustained resource consumption attacks are excluded.

Byzantine Fault with Proactive Recovery: one of the relay nodes exhibits the Byzantine Fault condition described above, and simultaneously, an additional relay node is undergoing proactive recovery. Effectively, only two correct relay nodes are available, while another compromised node attacks the system.

Note that in reality, our relay nodes complete proactive recovery and resume correct operation in a short period of time. However, we conduct the benchmarks

involving proactive recovery without rejuvenating the relay node. In essence, these benchmarks are in the worst case scenario, where only two correct relay nodes are available, over all one million actions throughout the 24 hours. We refer to this situation as non-optionality.

Under non-optionality, meeting the latency requirement is particularly tough. With only two nodes, a random delay on either (e.g., from network delays, kernel scheduling, or even effects of a Byzantine node's actions) would be reflected in the end-to-end latency. Compare this situation to one in which three nodes are available. In such a case, delays would have to occur on two of the three nodes independently and at the same time in order to be reflected in the final latency. If only one node is delayed, the other two nodes would still be able to complete the action in a timely manner.

4.6.1 Low Performance Servers Testbed Evaluation

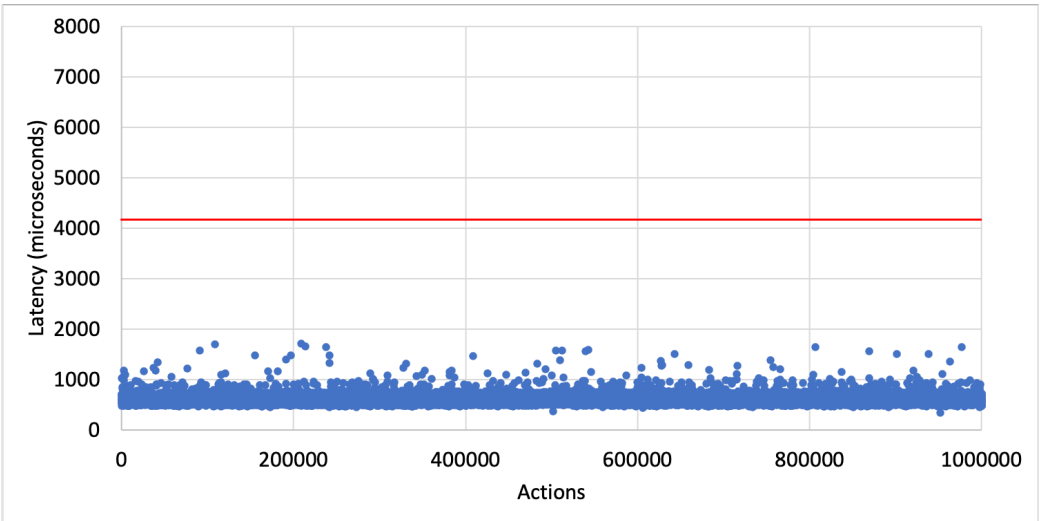
Table 4.1: Performance of the Arbiter Protocol and Peer Protocol under different operating conditions with four relay nodes ($f = 1, k = 1$) in Low Performance Servers Testbed

	Arbiter Protocol (microseconds)			Peer Protocol (microseconds)		
Operating Condition	**Minimum**	**Average**	**Maximum**	**Minimum**	**Average**	**Maximum**
Fault-Free (Normal)	**351**	**613**	**1718**	**1723**	**2187**	**3323**
Fail-Stop Fault or Proactive Recovery	**351**	**628**	**2441**	**1871**	**2260**	**3326**
Fail-Stop Fault and Proactive Recovery	**450**	**623**	**2675**	**1912**	**2328**	**7617** (8*)
Byzantine Fault	**402**	**639**	**2770**	**1737**	**2227**	**3785**
Byzantine Fault and Proactive Recovery	**421**	**637**	**2597**	**1867**	**2313**	**7699** (6*)

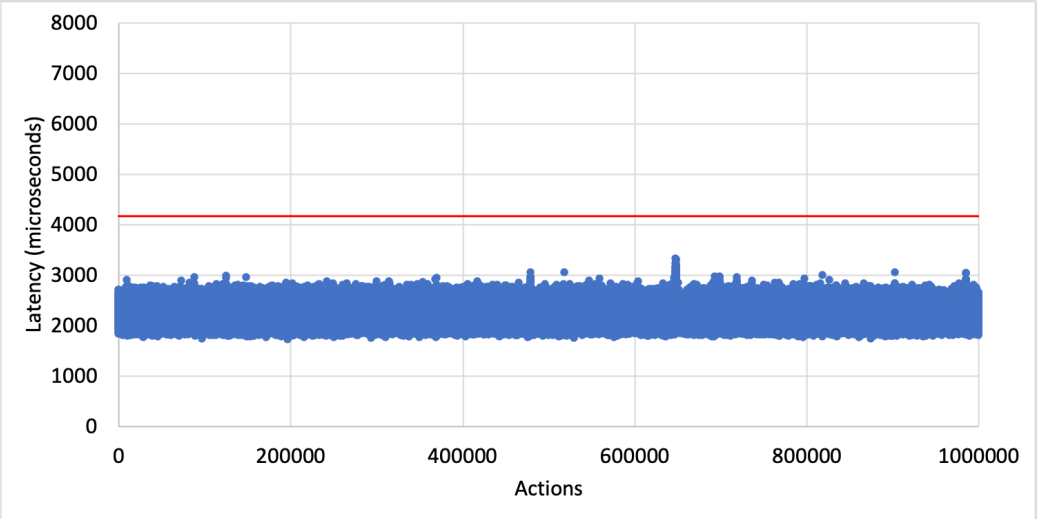
* The count of actions that crossed 4.167 milliseconds (out of 1 million total actions)

Table 4.1 reports the latency statistics: minimum, maximum, and average—across various operating conditions for both protocols. Additionally, detailed scatter plots are provided for notable benchmarks, illustrating one million actions (Figs. 4.4a - 4.7b). These plots allow for a visual comparison between protocols and reveal the impacts of faults and proactive recovery.

An immediate observation, as marked in the table 4.1, is both the Arbiter Protocol and the Peer Protocol meet the real time latency requirement during the Fault-Free operating condition (Fig. 4.4a and Fig. 4.4b respectively). Comparing average latencies between the protocols, the Peer Protocol averages approximately 1500 microseconds longer. This is the price associated with threshold cryptography we pay for its advantages: the reduced attack surface for the circuit breaker and seamless integration with the substation.

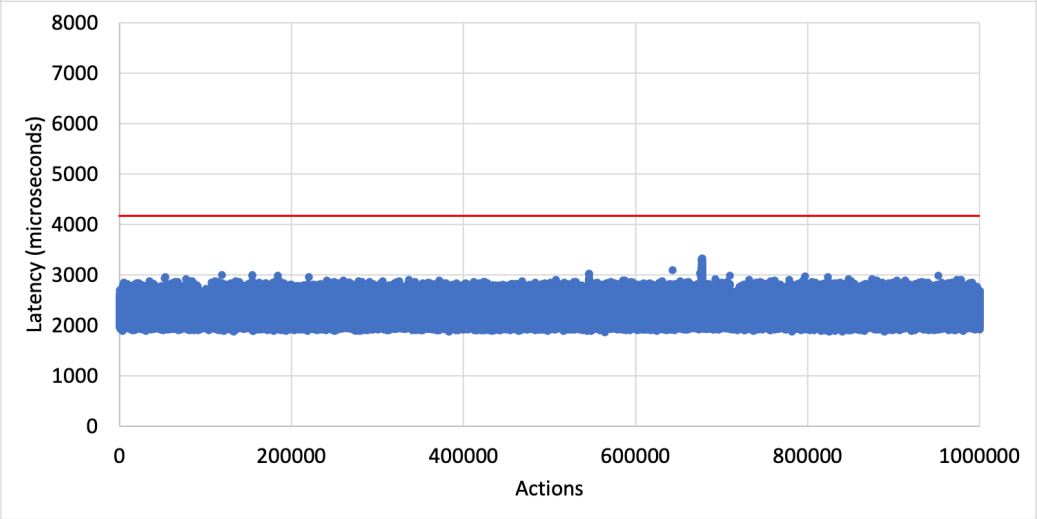


(a) Low Performance Servers Testbed Arbiter Protocol - Fault-Free

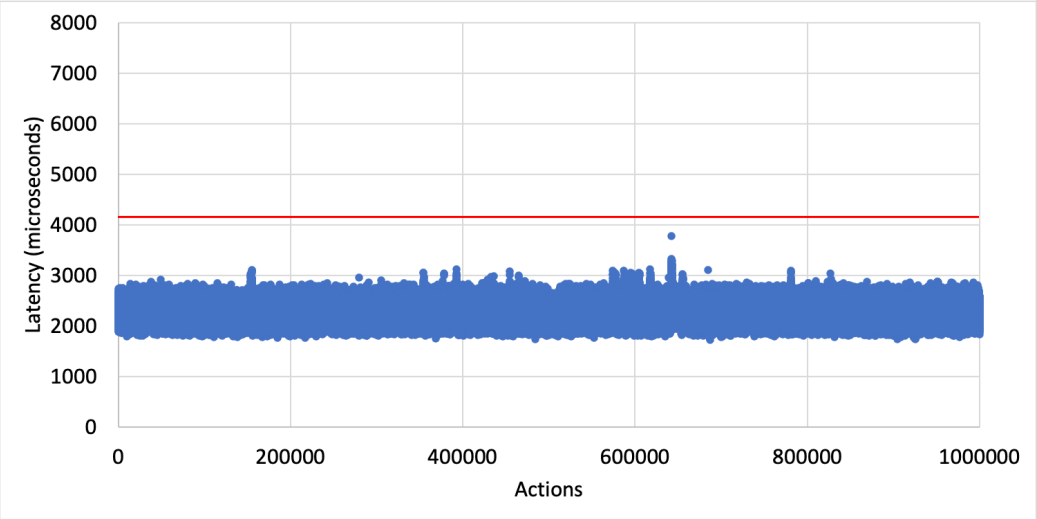


(b) Low Performance Servers Testbed Peer Protocol - Fault-Free

Figure 4.4: Low Performance Servers Testbed Fault-Free operating condition performance evaluation with million actions

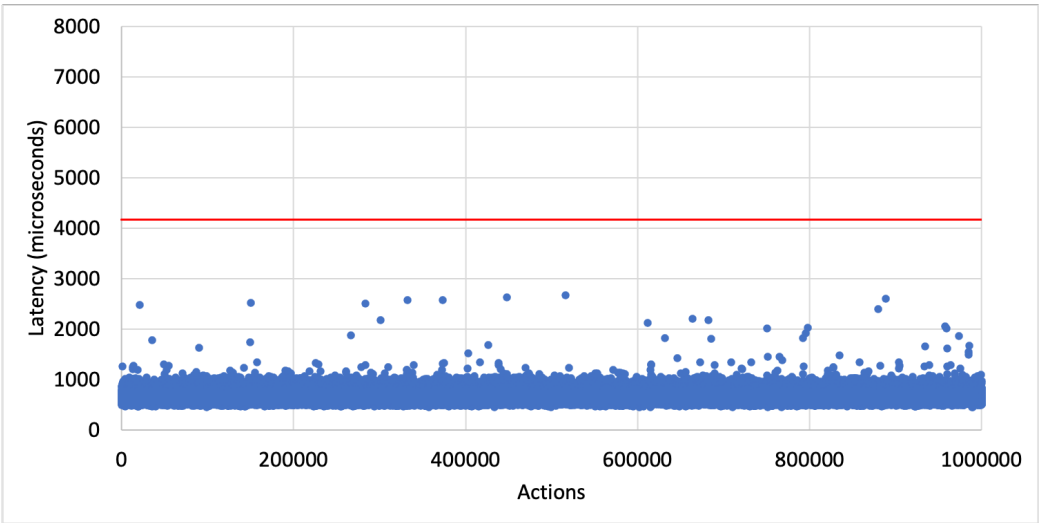


(a) Low Performance Servers Testbed Peer Protocol - Fail-Stop Fault

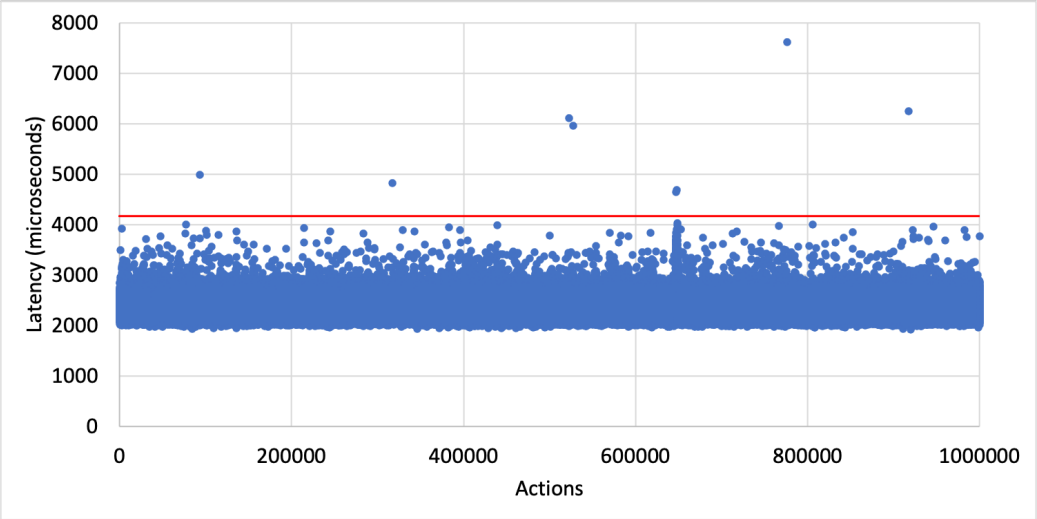


(b) Low Performance Servers Testbed Peer Protocol - Byzantine Fault

Figure 4.5: Low Performance Servers Testbed Peer Protocol Fail-Stop and Byzantine fault operating condition performance evaluation with million actions

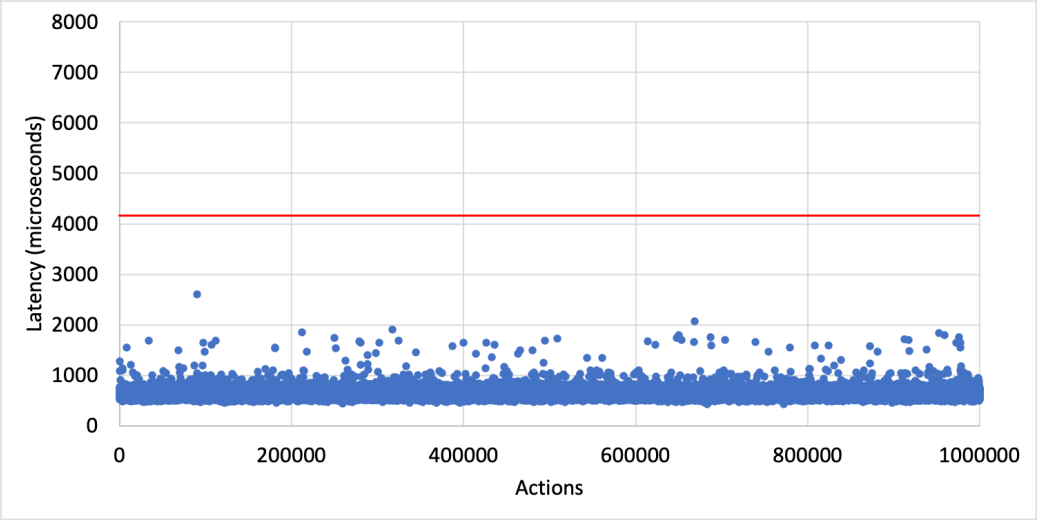


(a) Low Performance Servers Testbed Arbiter Protocol - Fail-Stop Fault and Proactive Recovery

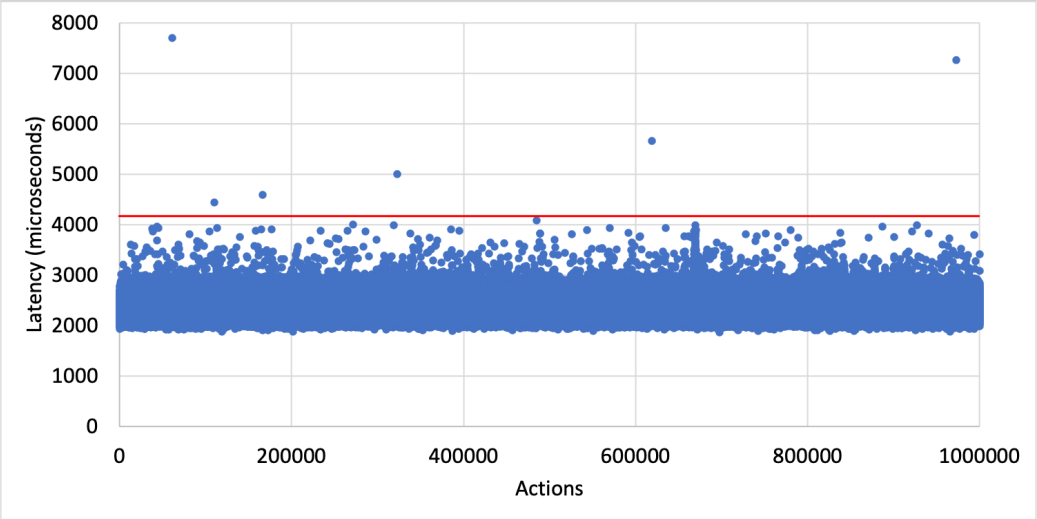


(b) Low Performance Servers Testbed Peer Protocol - Fail-Stop Fault and Proactive Recovery

Figure 4.6: Low Performance Servers Testbed Fail-Stop fault with proactive recovery operating condition performance evaluation with million actions



(a) Low Performance Servers Testbed Arbiter Protocol - Byzantine Fault and Proactive Recovery



(b) Low Performance Servers Testbed Peer Protocol - Byzantine Fault and Proactive Recovery

Figure 4.7: Low Performance Servers Testbed Byzantine fault with proactive recovery operating condition performance evaluation with million actions

Under Fail-Stop Fault (or Proactive Recovery) and Byzantine fault conditions, Table 4.1 illustrates the effects of losing a correct relay node or the presence of a Byzantine relay node. Specifically, these conditions lead to increased latency and a slight rise in variance. Despite these impacts, both protocols continue to satisfy the real-time latency requirements in these scenarios.

The most notable observation in this testbed concerns the impact of non-optionality operating conditions. This effect is evident when comparing the Fail-Stop (alone) Fault condition to the Fail-Stop Fault with Proactive Recovery condition, and similarly, the Byzantine Fault (alone) condition to the Byzantine Fault with Proactive Recovery condition. While non-optionality does not have a significant impact on the average case as seen in Table 4.1, it has a significant impact on the worst cases, as seen by the outliers (including those that do not cross the latency requirement) (Figs. 4.6a, 4.6b, 4.7a, 4.7b).

In the non-optionality conditions, the Arbiter Protocol meets the real time latency requirements in all cases (Figs. 4.6a, 4.7a), while the Peer Protocol meets it with a few exceptions: 8 actions out of one million in the Fail-Stop Fault with Proactive Recovery condition (Fig. 4.6b) and 6 actions out of one million in the Byzantine Fault with Proactive Recovery condition (Fig. 4.7b) violate the requirement.

It should be noted that non-optionality is typically short-lived in practice, as nodes undergoing proactive recovery rejoin the protocol within seconds. Despite occasional lapses, the Peer Protocol maintains better than five nines dependability (99.999%) even under the worst case conditions. However, the rare exceptions observed under non-optionality conditions still highlight the inherent risks on this testbed.

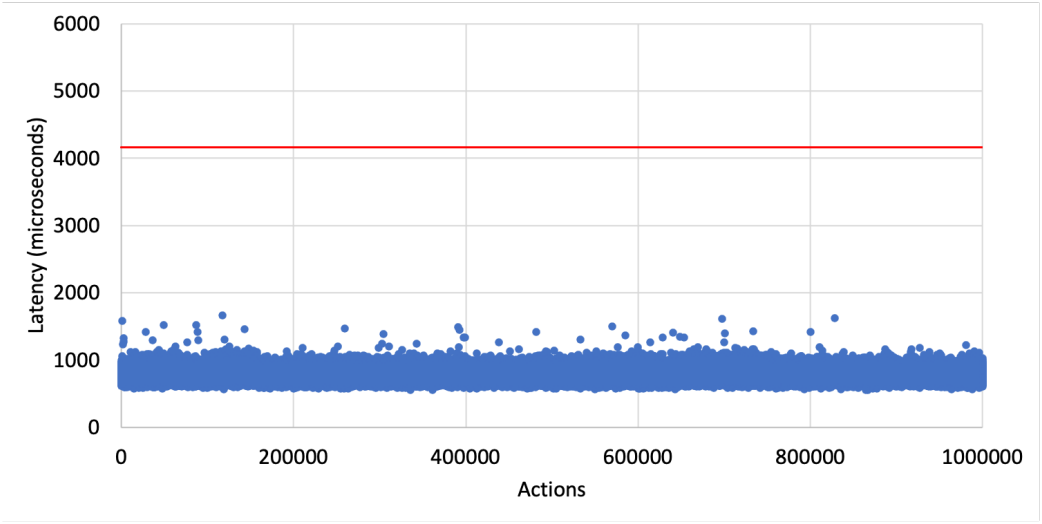
4.6.2 Modern Industrial NUCs Testbed Evaluations

Table 4.2: Performance of the Arbiter Protocol and Peer Protocol under different operating conditions with four relay nodes ($f = 1, k = 1$) in Modern Industrial NUCs Testbed

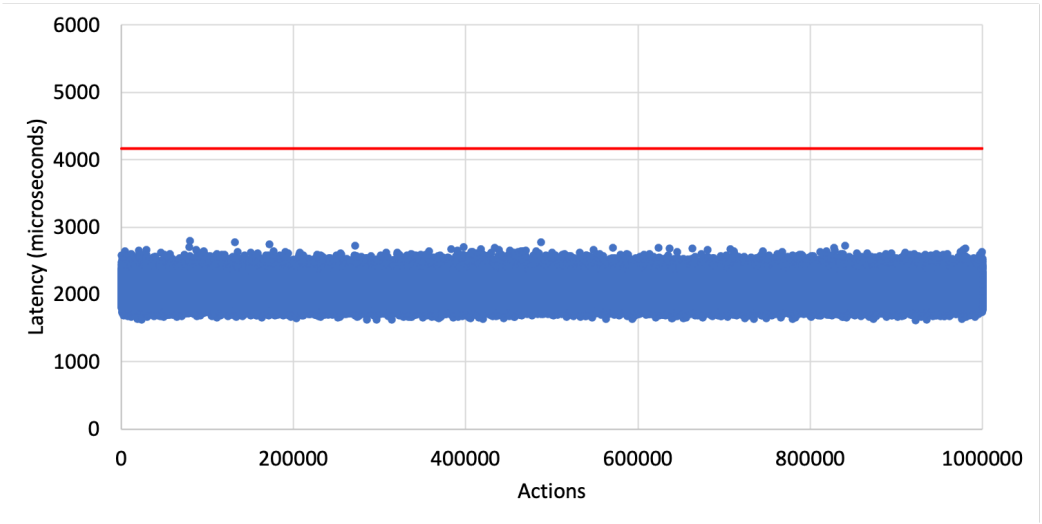
	Arbiter Protocol (microseconds)			Peer Protocol (microseconds)		
Operating Condition	**Minimum**	**Average**	**Maximum**	**Minimum**	**Average**	**Maximum**
Fault-Free (Normal)	**556**	**794**	**1665**	**1604**	**2048**	**2789**
Fail-Stop Fault or Proactive Recovery	**584**	**844**	**2104**	**1604**	**2132**	**3135**
Fail-Stop Fault and Proactive Recovery	**593**	**858**	**2887**	**1595**	**2167**	**5025** (4*)
Byzantine Fault	**564**	**887**	**2691**	**1716**	**2268**	**3253**
Byzantine Fault and Proactive Recovery	**577**	**904**	**2879**	**1733**	**2261**	**5028** (2*)

* The count of actions that crossed 4.167 milliseconds (out of 1 million total actions)

Table 4.2 reports the minimum, maximum and average of the latency distribution under each possible operating condition for both the protocols. For several of the

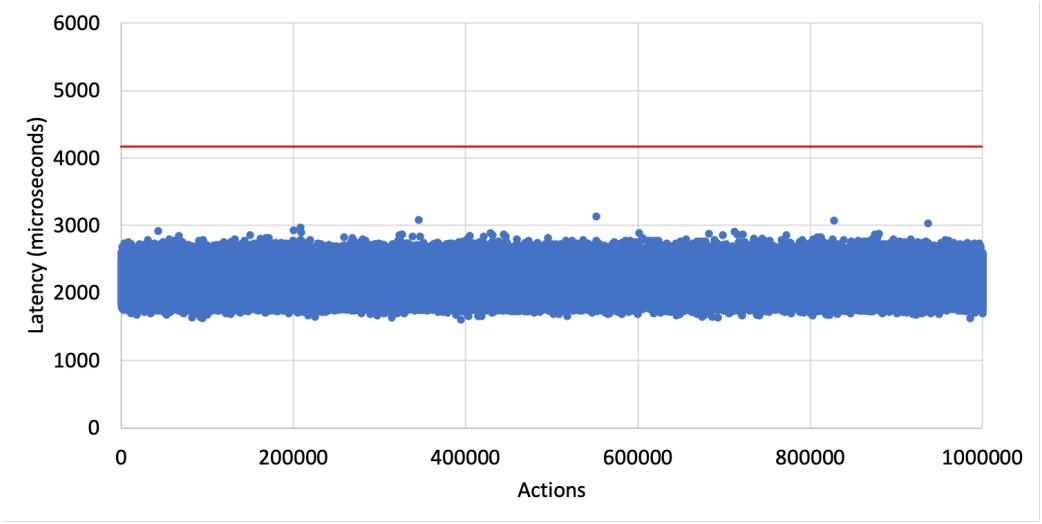


(a) Modern Industrial NUCs Testbed Arbiter protocol - Fault-Free

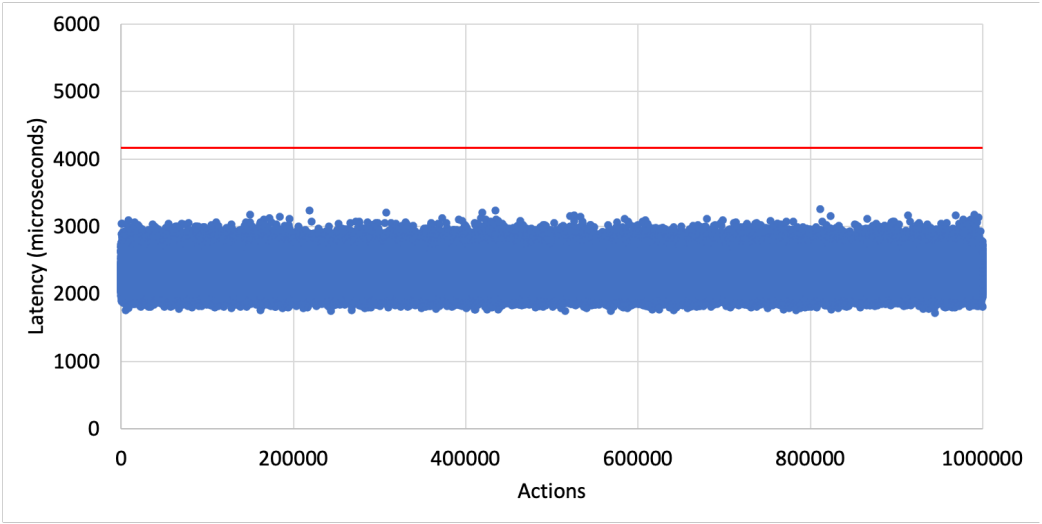


(b) Modern Industrial NUCs Testbed Peer Protocol - Fault-Free

Figure 4.8: Modern Industrial NUCs Testbed Fault-Free operating condition performance evaluation with million actions

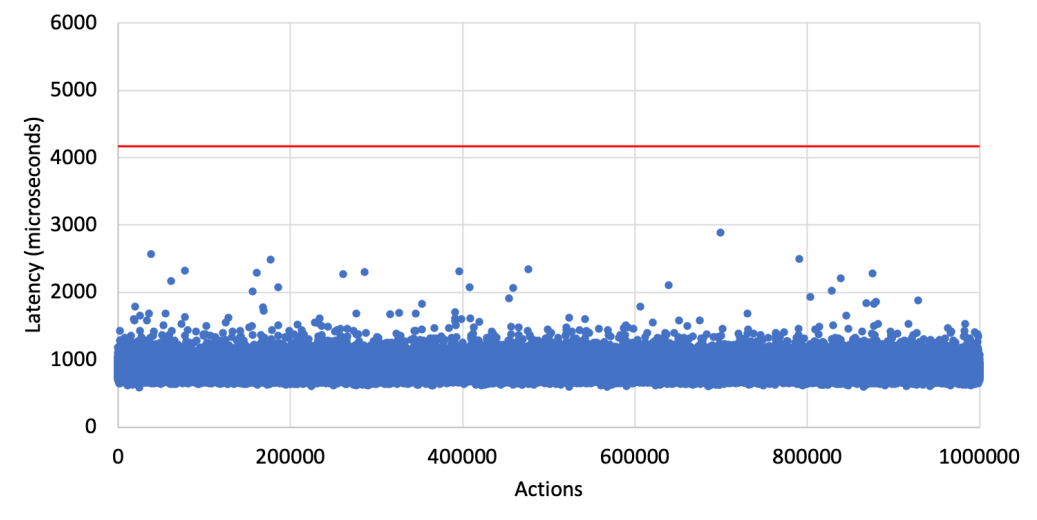


(a) Modern Industrial NUCs Testbed Peer Protocol - Fail-Stop Fault

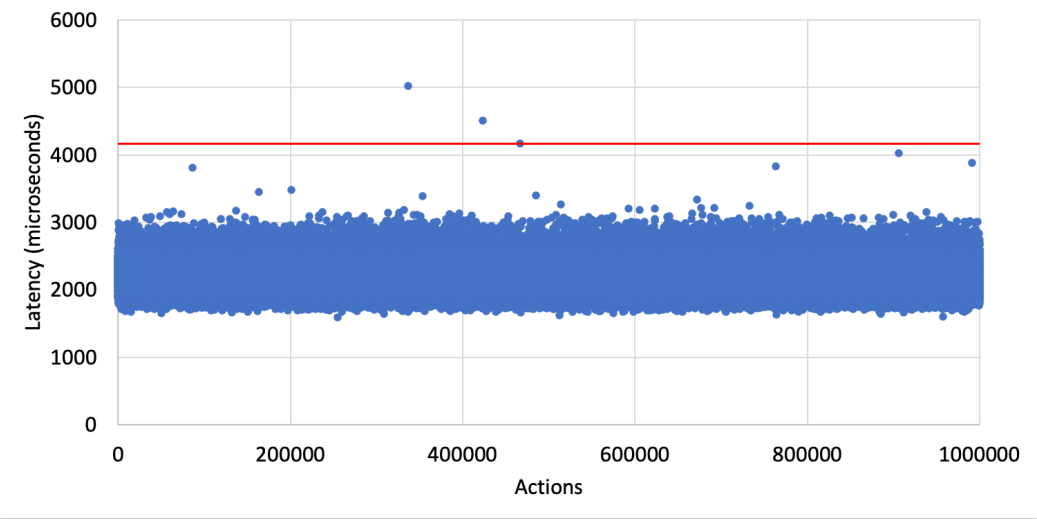


(b) Modern Industrial NUCs Testbed Peer Protocol - Byzantine Fault

Figure 4.9: Modern Industrial NUCs Testbed Peer Protocol Fail-Stop and Byzantine fault operating condition performance evaluation with million actions

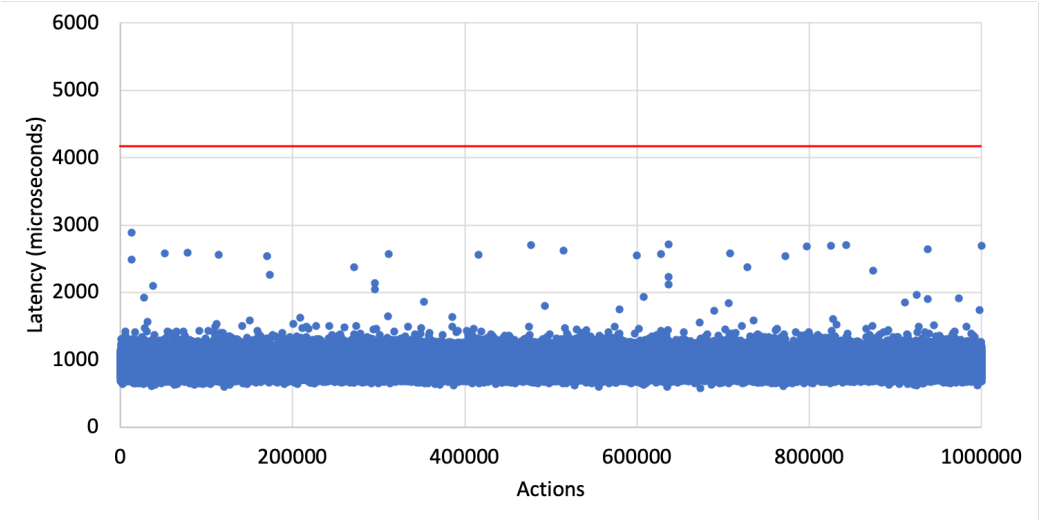


(a) Modern Industrial NUCs Testbed Arbiter Protocol - Fail-Stop Fault and Proactive Recovery

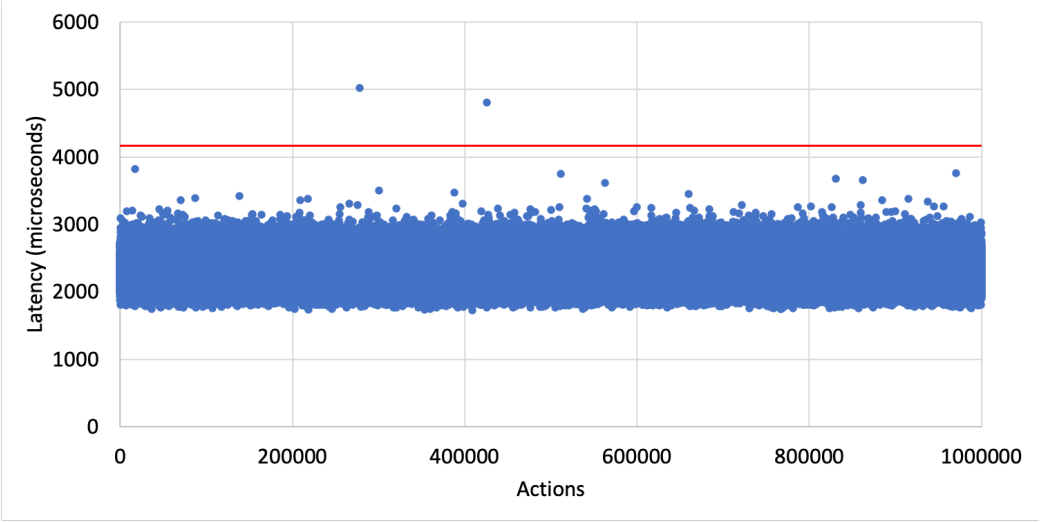


(b) Modern Industrial NUCs Testbed Peer Protocol - Fail-Stop Fault and Proactive Recovery

Figure 4.10: Modern Industrial NUCs Testbed Fail-Stop fault with proactive recovery operating condition performance evaluation with million actions



(a) Modern Industrial NUCs Testbed Arbiter Protocol - Byzantine Fault and Proactive Recovery



(b) Modern Industrial NUCs Testbed Peer Protocol - Byzantine Fault and Proactive Recovery

Figure 4.11: Modern Industrial NUCs Testbed Byzantine fault with proactive recovery operating condition performance evaluation with million actions

notable benchmarks, we also present detailed scatter plots for the one million actions (Figs. 4.8a - 4.11b). By comparing these plots, we can visualize the differences between the protocols and the effects of Byzantine faults and proactive recovery.

An immediate observation, as marked in the table, is both the Arbiter Protocol and the Peer Protocol meet the real time latency requirement during the Fault-Free operating condition (Fig. 4.8a and Fig. 4.8b respectively). When comparing the average latencies for the two protocols in the table, the Peer Protocol takes on average about 1300 microseconds longer. As previously noted (in Low Performance Servers Testbed evaluations), this is the price we pay for its advantages: the reduced attack surface for the circuit breaker and seamless integration with the substation.

In Fail-Stop Fault or Proactive Recovery operation conditions, the table shows that losing a single correct relay node shifts the distribution of latencies higher, with an average increase of about 50 microseconds for Arbiter Protocol and about 80 microseconds for Peer Protocol . However, both the protocols meet the real time latency requirement.

Another observation comes from comparing the Fail-Stop condition to the Byzantine Fault condition, as it quantifies the effect of an active intruder: the average latency increases by 50 microseconds for the Arbiter Protocol and 140 microseconds in case of the Peer Protocol , in addition to slight increase in variance. The impact of two factors—the cost of losing a single correct relay node and the effects of a Byzantine node—can be viewed as components that sum to the total difference in latency between the normal case and Byzantine fault (about 100 microsecond for the Arbiter Protocol and 220 microseconds for the Peer Protocol).

An interesting observation is the effect of non-optionality on both protocols. This effect is observed by comparing the Fail-Stop Fault condition to the Fail-Stop Fault with Proactive Recovery condition, and similarly, the Byzantine Fault condition to the Byzantine Fault with Proactive Recovery condition. While non-optionality does not have a significant impact on the average case as seen in Table 4.2, it has a significant impact on the worst cases, as seen by the outliers (including those that do not cross the latency requirement) (Figs. 4.10a, 4.10b, 4.11a, 4.11b).

In the non-optionality conditions, the Arbiter Protocol meets the real time latency requirements in all cases (Figs. 4.10a, 4.11a), while the Peer Protocol meets it with a few exceptions: 4 actions out of one million in the Fail-Stop Fault with Proactive Recovery condition (Fig. 4.10b) and 2 actions out of one million in the Byzantine Fault with Proactive Recovery condition (Fig. 4.11b) violate the requirement. That the Fail-Stop Fault with Proactive Recovery condition has more exceptions is simply due to random chance, as the non-optionality is the critical factor for the worst case latencies, while the Byzantine behavior mainly impacts the average latency.

We reiterate that non-optionality in practice is typically transient, as a node undergoing proactive recovery reintegrates into the protocol within a few seconds. Despite its limitations, the Peer Protocol demonstrates an impressive dependability of over 99.999% (five nines) even under these challenging conditions.

A critical observation is that by utilizing more powerful machines, we can reduce the outliers. However, even with fewer exceptions observed (compared to the low-performance server testbed), the remaining outliers still underscore the residual risks associated with exceptions.

4.6.3 Modern Low Cost Data Center Servers Testbed Evaluations

Table 4.3: Performance of the Arbiter Protocol and Peer Protocol under different operating conditions with four relay nodes ($f = 1, k = 1$) in Modern Low Cost Data Center Servers Testbed

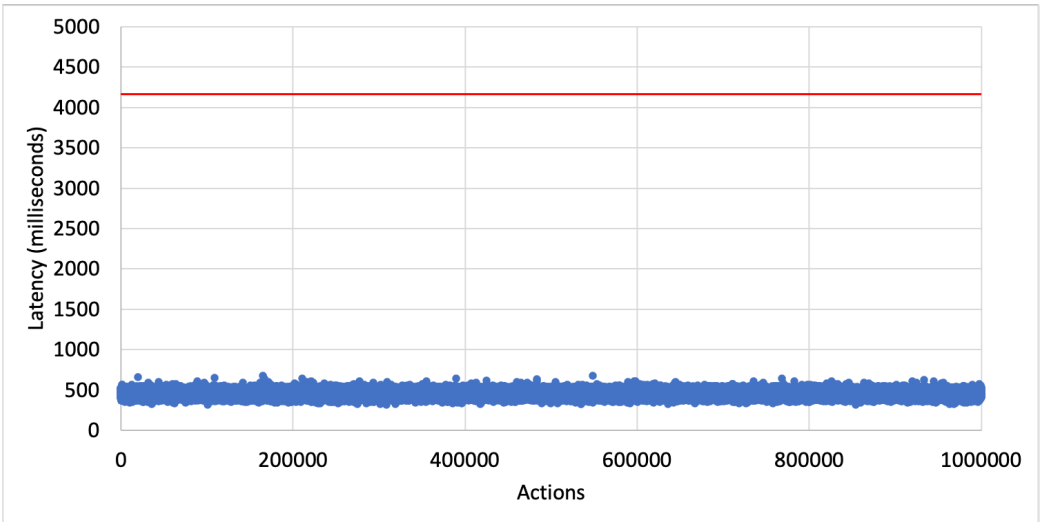
	Arbiter Protocol (microseconds)			Peer Protocol (microseconds)		
Operating Condition	**Minimum**	**Average**	**Maximum**	**Minimum**	**Average**	**Maximum**
Fault-Free (Normal)	**320**	**462**	**682**	**1124**	**1395**	**1790**
Fail-Stop Fault or Proactive Recovery	**324**	**461**	**714**	**1125**	**1398**	**1857**
Fail-Stop Fault and Proactive Recovery	**323**	**478**	**1277**	**1140**	**1422**	**2926**
Byzantine Fault	**298**	**461**	**678**	**1133**	**1403**	**1833**
Byzantine Fault and Proactive Recovery	**317**	**463**	**669**	**1143**	**1436**	**1978**

Both the Arbiter and Peer protocols successfully met the quarter-power cycle requirement under all operating conditions. This achievement can be attributed to the servers' enhanced processing power, which leads to lower latency. As a result, average latency remains consistently similar across all operating conditions, even under the most challenging non-optionality scenarios. Although maximum latency does show a slight increase in non-optionality conditions, it still remains significantly below the quarter-power cycle requirement.

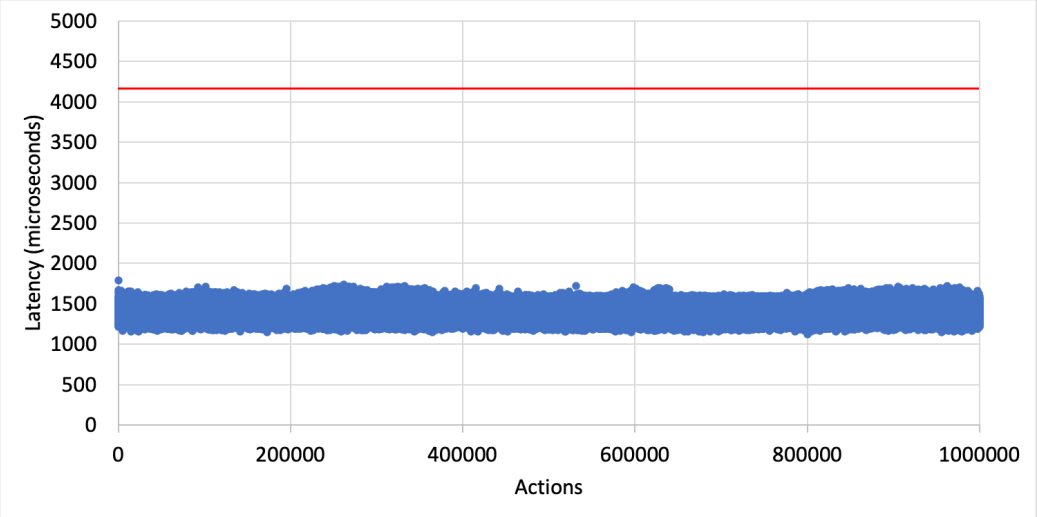
However, given the physical operating conditions of the power domain, where substations can be unmanned and exposed to environmental factors, it might not be feasible to deploy the modern data center compute servers in field. Hence, it becomes critical to explore other alternatives to meet the performance requirements. Therefore, we investigate the real-time kernel option in our least powerful testbed i.e., Low Performance Servers Testbed, with observations detailed in section 4.7.

4.7 Real-time Kernels in Support of Low-cost Solutions

The performance evaluation results from Low Performance Servers Testbed (Table 4.1) and Modern Industrial NUCs Testbed (Table 4.2) demonstrate that meeting latency requirements under non-optionality is particularly tough. With only two

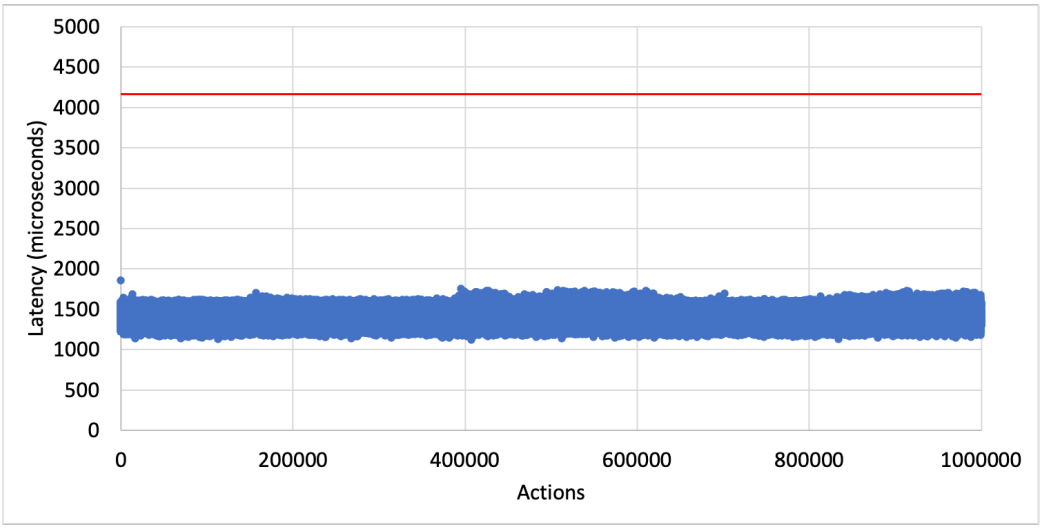


(a) Modern Low Cost Data Center Servers Testbed Arbiter Protocol - Fault-Free

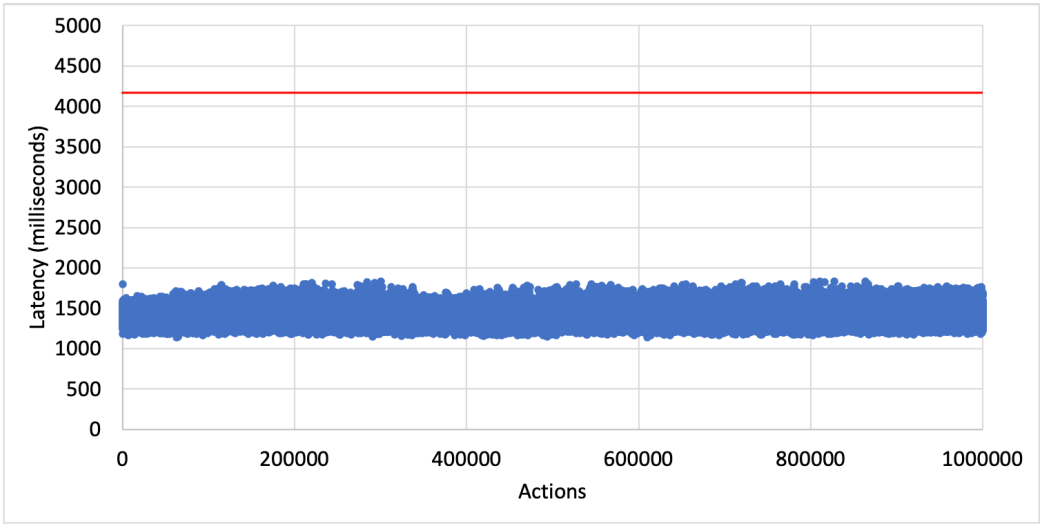


(b) Modern Low Cost Data Center Servers Testbed Peer Protocol - Fault-Free

Figure 4.12: Modern Low Cost Data Center Servers Testbed Fault-Free operating condition performance evaluation with million actions

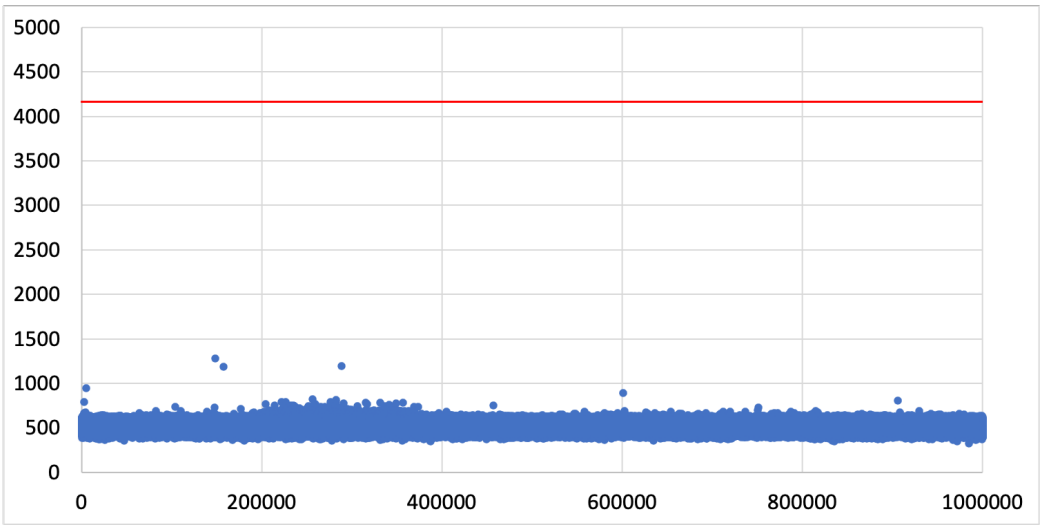


(a) Modern Low Cost Data Center Servers Testbed Peer Protocol - Fail-Stop Fault

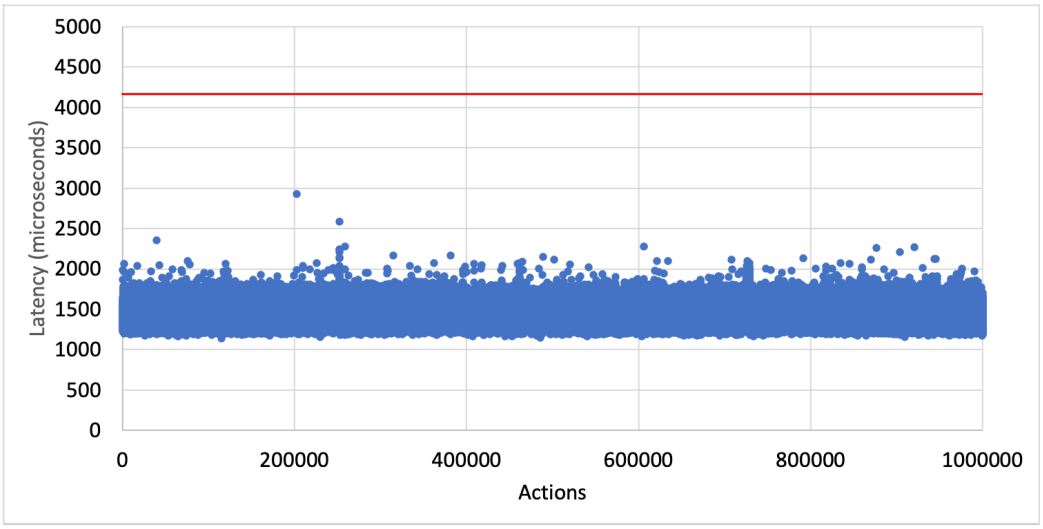


(b) Modern Low Cost Data Center Servers Testbed Peer Protocol - Byzantine Fault

Figure 4.13: Modern Low Cost Data Center Servers Testbed Peer Protocol Fail-Stop and Byzantine fault operating condition performance evaluation with million actions

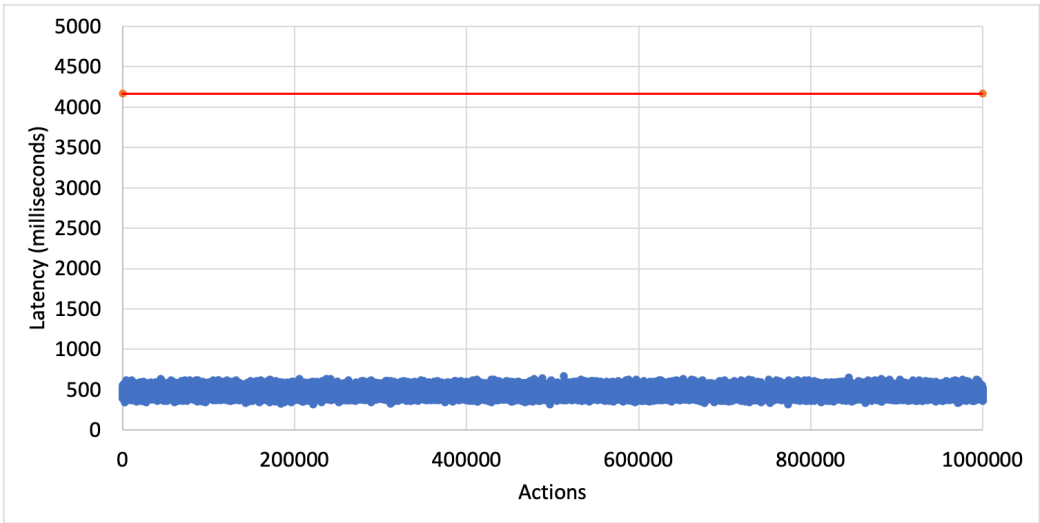


(a) Modern Low Cost Data Center Servers Testbed Arbiter Protocol - Fail-Stop Fault and Proactive Recovery

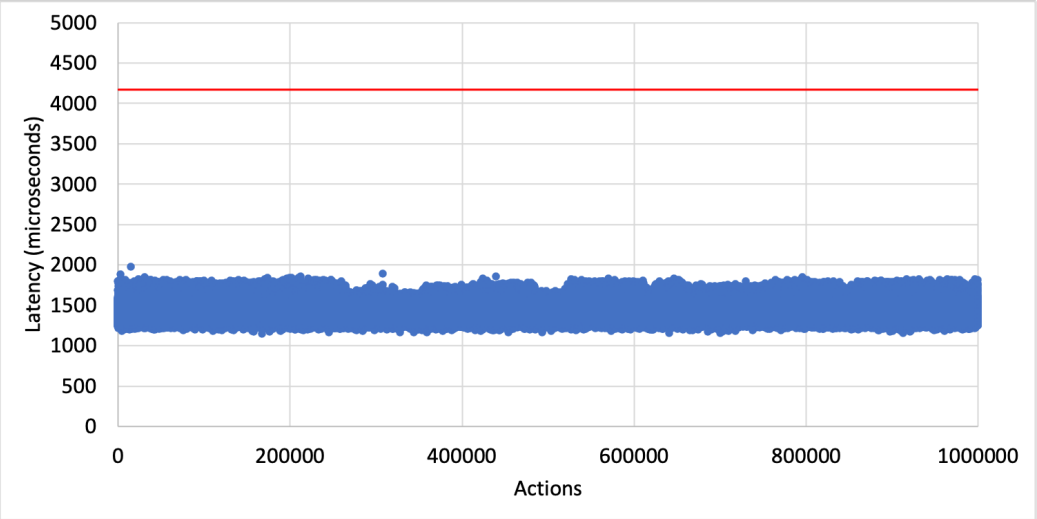


(b) Modern Low Cost Data Center Servers Testbed Peer Protocol - Fail-Stop Fault and Proactive Recovery

Figure 4.14: Modern Low Cost Data Center Servers Testbed Fail-Stop fault with proactive recovery operating condition performance evaluation with million actions



(a) Modern Low Cost Data Center Servers Testbed Arbiter Protocol - Byzantine Fault and Proactive Recovery



(b) Modern Low Cost Data Center Servers Testbed Peer Protocol - Byzantine Fault and Proactive Recovery

Figure 4.15: Modern Low Cost Data Center Servers Testbed Byzantine fault with proactive recovery operating condition performance evaluation with million actions

nodes available, a random delay on either node (e.g., from network delays, kernel scheduling, or even effects of a Byzantine node's actions) would be reflected in the end-to-end latency. An option to solve this issue would be to invest and gain performance as demonstrated in Modern Low Cost Data Center Servers Testbed 4.3. Alternatively, we can also consider exploring specialized real-time hardware and/or software. However, maintaining a COTS hardware and open-source software environment simplifies system management and maintenance [75, 76]. Therefore, we explore a different trade-off, evaluating whether the Linux real-time kernel option can achieve the needed performance in the weakest testbed i.e., Low Performance Servers Testbed.

4.7.1 Evaluation with Real-time Linux kernel

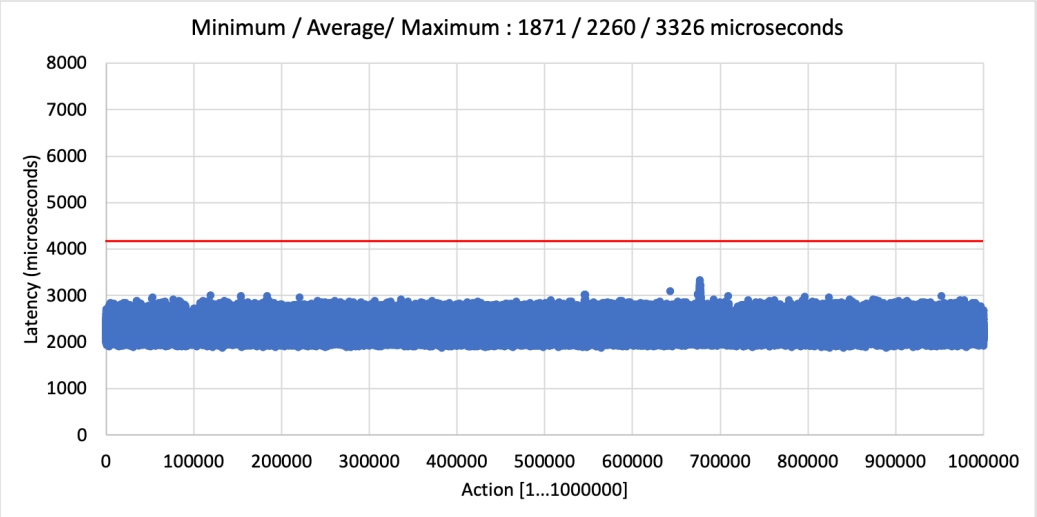
Table 4.4: Performance of the Peer Protocol under different operating conditions with four relay nodes ($f = 1, k = 1$) in Low Performance Servers Testbed with normal and real-time kernels

	Normal Kernel (microseconds)			Real-Time Kernel (microseconds)		
Operating Condition	**Minimum**	**Average**	**Maximum**	**Minimum**	**Average**	**Maximum**
Fault-Free (Normal)	**1723**	**2187**	**3323**	**1637**	**1950**	**3596**
Fail-Stop Fault or Proactive Recovery	**1871**	**2260**	**3326**	**1608**	**1976**	**3726**
Fail-Stop Fault and Proactive Recovery	**1912**	**2328**	**7617** (8*)	**1750**	**2015**	**3996**
Byzantine Fault	**1737**	**2227**	**3785**	**1665**	**1984**	**4002**
Byzantine Fault and Proactive Recovery	**1867**	**2313**	**7699** (6*)	**1767**	**2019**	**4101**

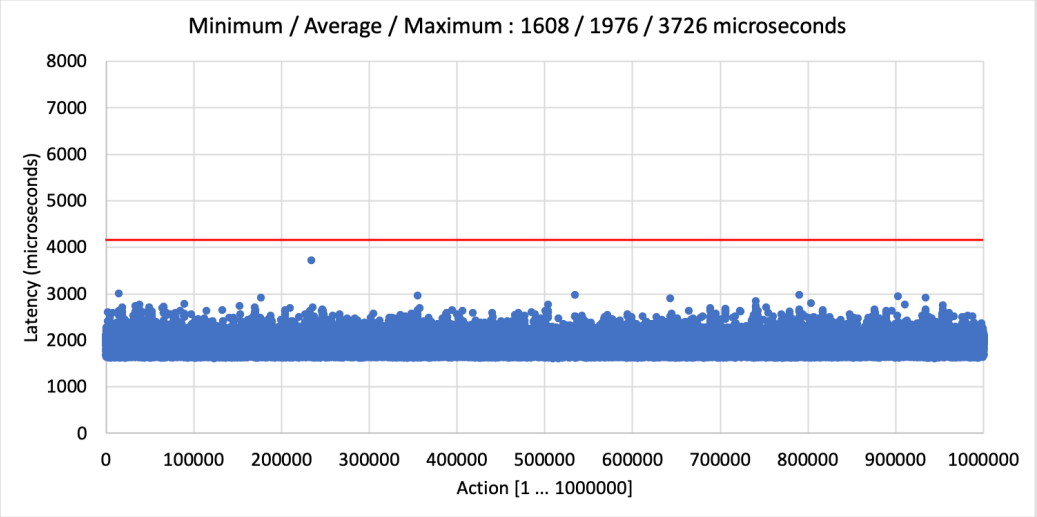
* The count of actions that crossed 4.167 milliseconds (out of 1 million total actions). Note: The normal kernal stats are same as in Table 4.1

Table 4.2 reports the minimum, average and maximum latency in microseconds for all five operating conditions supported by the threat model for both the normal and real-time kernels. Figures 4.17a, 4.17b, 4.18a and 4.18b focus on the most demanding scenarios: Fail-Stop fault with simultaneous proactive recovery and Byzantine fault with simultaneous proactive recovery.

The normal linux kernel is optimized for throughput and fair scheduling of tasks, while the real-time kernel is optimized to maintain low latency, consistent response time and determinism. These characteristics are particularly important in the conditions with non-optionality where any random delay impacts the system performance. Due to its features, in real-time kernel benchmarks, when we use high priority and FIFO scheduling policy, all actions meet the 4.167ms requirement, and average latency is reduced by about 300 microseconds across all operating conditions (Table 4.4). These observations can also be immediately noted by comparing benchmark plots of normal kernel to those of real-time kernel in the three operating conditions shown: Fail-Stop fault (Figure 4.16a vs Figure 4.16b), Fail-Stop fault with simultaneous proactive recovery (Figure 4.17a vs Figure 4.17b), and Byzantine fault with simultaneous proactive recovery (Figure 4.18a vs Figure 4.18b).

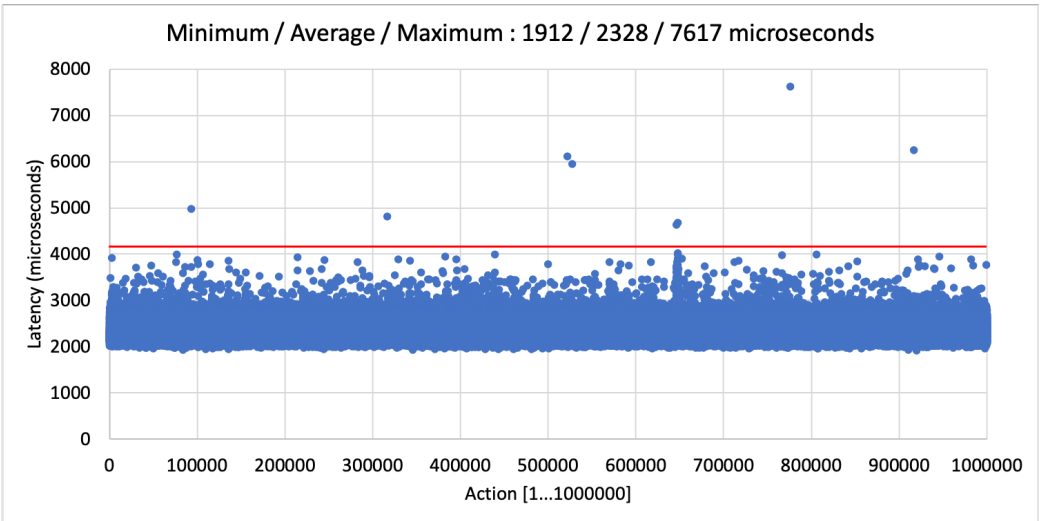


(a) Low Performance Servers Testbed Normal Kernel: Fail-Stop Fault (same as Figure 4.5a)

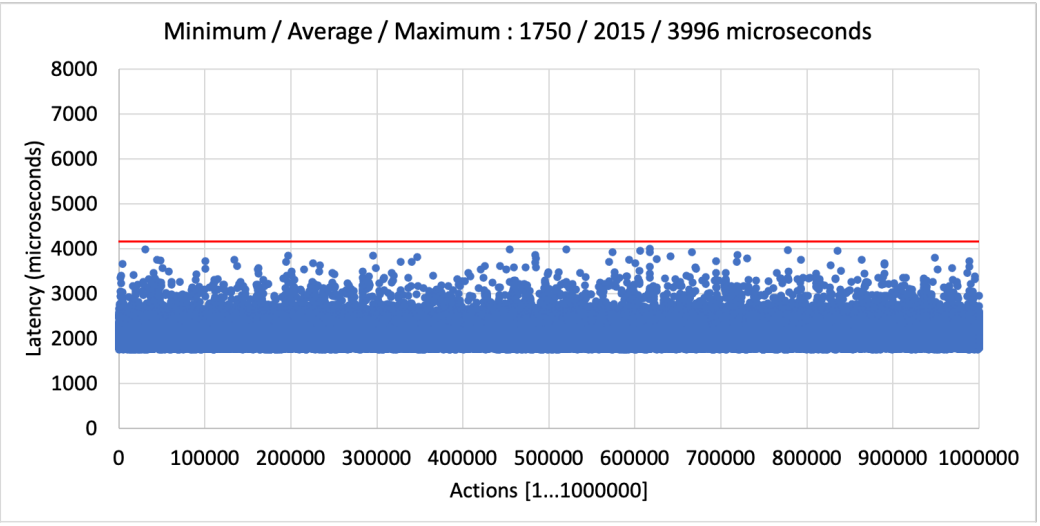


(b) Low Performance Servers Testbed Real-Time Kernel: Fail-Stop Fault

Figure 4.16: Low Performance Servers Testbed normal and real-time kernels in Fail-stop fault operating condition performance evaluations with million actions

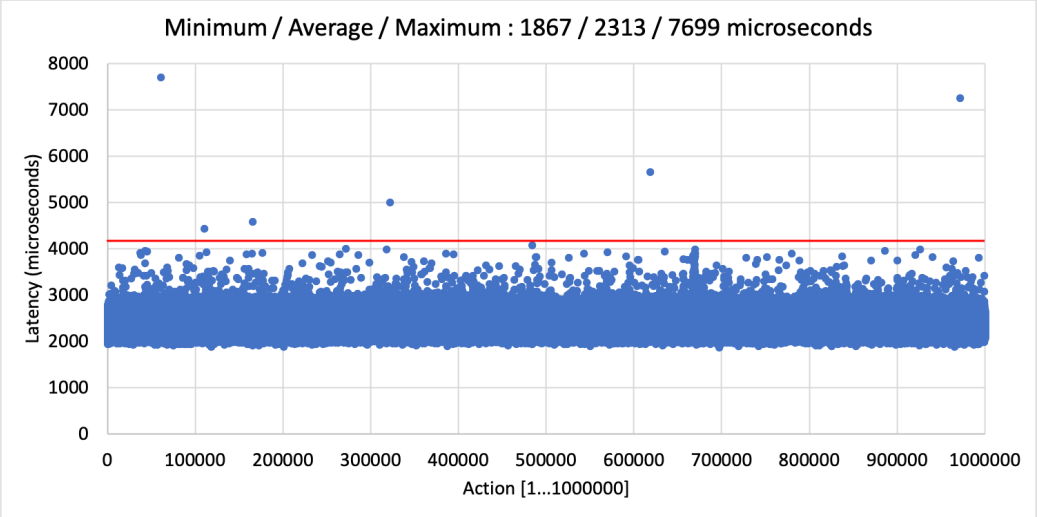


(a) Low Performance Servers Testbed Normal Kernel: Fail-Stop Fault with Proactive Recovery (same as Figure 4.6b)

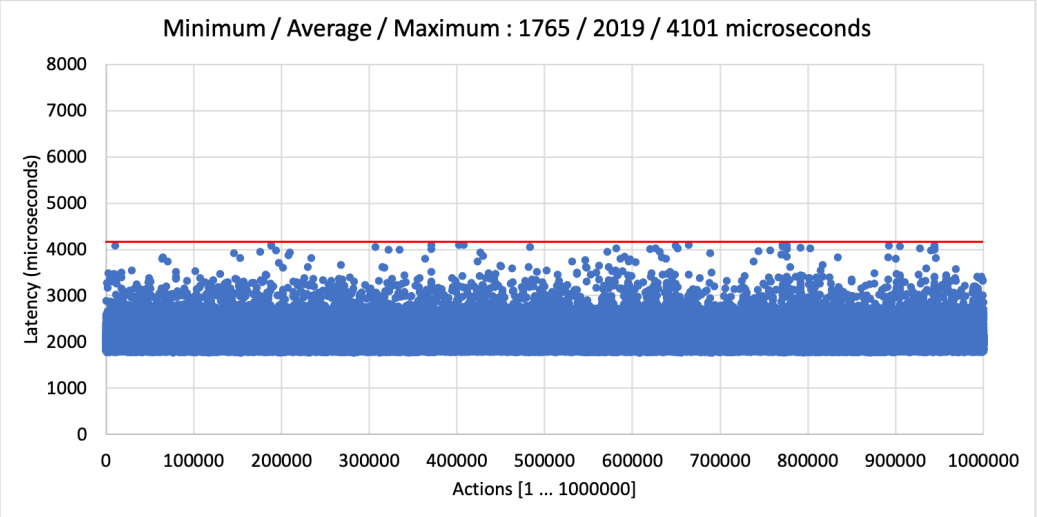


(b) Low Performance Servers Testbed Real-Time Kernel: Fail-Stop Fault with Proactive Recovery

Figure 4.17: Low Performance Servers Testbed normal and real-time kernels in Fail-stop fault and proactive recovery operating condition performance evaluations with million actions



(a) Low Performance Servers Testbed Normal Kernel: Byzantine Fault with Proactive Recovery (same as Figure 4.7b)



(b) Low Performance Servers Testbed Real-Time Kernel: Byzantine Fault with Proactive Recovery

Figure 4.18: Low Performance Servers Testbed normal and real-time kernels in Byzantine fault and proactive recovery operating condition performance evaluations with million actions

The determinism and latency stability of real-time kernel offers a new deployment option for real-time Byzantine resilient critical infrastructure, enabling us to remain in open-source realm and get the benefits of the Peer Protocol while meeting real-time requirements even in the weakest Testbed.

4.8 Discussion and Analysis of System Tradeoffs

We consider the four design challenges: meeting real-time latency requirement, managing the economic cost, supporting continuous availability over a long system lifetime, and allowing seamless substation integration.

Meeting Real-time Latency Requirement: We evaluated the system across four testbeds, each chosen to represent different segments of the grid computing landscape. These testbeds reflect not only the current state of technology but also future trends in grid deployment environments.

The Arbiter Protocol requires a single one-way communication from relay nodes to the breaker node to achieve a coordinated action, providing Byzantine resilience with the least latency. Arbiter Protocol consistently meets real-time latency requirements, even in non-optionality conditions.

The Peer Protocol, which involves additional latency due to threshold cryptography, generally meets latency requirements but experiences occasional lapses, especially in non-optionality conditions such as Fail-Stop Fault with Proactive Recovery and Byzantine Fault with Proactive Recovery.

The low performance servers testbed representative of the compute power expected to dominate current and near-future grid infrastructures has the weakest machines in the evaluations. The Peer Protocol in this testbed exhibited the highest number of latency exceptions under non-optionality conditions. However, it is important to reiterate that despite these exceptions, the testbed still maintained five-nines (99.999%) reliability. As we moved to more powerful modern industrial NUCs, the number of exceptions decreased, though they persisted. In contrast, both the Modern Low-cost Data Center Servers testbed and the Real-Time Kernel testbed showed no exceptions whatsoever. This suggests that exceptions can be effectively eliminated, and real-time latency requirements can be met in all operating conditions by either deploying more powerful compute resources or leveraging real-time kernels in less powerful machines.

Comparison with BFT SMR-based solutions: Any deployable BFT SMR protocol would need at least one additional one-way exchange of messages during Fault-Free operation for ordering (if not a full round), compared with the Peer Protocol. We can estimate that this additional one-way exchange would make the average latency over 3ms, considering the one-way exchange in the Arbiter Protocol takes on average about 1ms. Furthermore, for any leader-based protocol, a malicious leader

could trigger a leader election at a critical time. The system would then need to detect this failure and perform the leader change, further increasing the latency. Therefore, we can conclude that a BFT SMR-based protocol is unlikely to provide the latency requirement in the face of Byzantine faults, while using similar hardware, operating system, and network.

Economic Cost: Both the Arbiter Protocol and Peer Protocol need $2f + k + 1$ relay nodes to tolerate f Byzantine faults and k relays undergoing proactive recovery simultaneously. Compared to the $3f + 2k + 1$ relay nodes needed by any BFT SMR protocol for the same guarantee, this is a significant reduction that translates directly into cost savings. Given that relays are relatively expensive, this also makes our architecture much more viable for deployment.

Continuous Availability with Long System Lifetime: All approaches support diversity and proactive recovery. Proactive recovery ensures that we can reclaim relay nodes from an intruder after a successful intrusion, ensuring long system lifetime.

Seamless Substation Integration: The Arbiter Protocol, while optimal in terms of latency, has a complex breaker node: any involved circuit breaker in the substation needs to know the architecture of the relay nodes, their identities, and threshold for action ($f_s + 1$ out of $2f_s + k_s + 1$). Additionally, this sophisticated logic in breaker nodes increases the attack surface across the substation, making such nodes more susceptible to intrusions and harder to harden. Any configuration changes in the Byzantine resilient solution, such as the level of resilience associated with f_s and k_s, or even a replacement of a relay node associated with a new id and keys, would affect the configuration of any involved breaker node. This makes a seamless deployment and scaling of the Arbiter Protocol much less feasible when compared to the Peer Protocol.

Deployment Tradeoff: The Arbiter and Peer Protocols provide a deployment tradeoff: While the Arbiter Protocol always meets the real-time requirement in all situations, it presents a larger attack surface as well as added integration complexity. The Peer Protocol significantly reduces the attack surface while providing seamless integration into the substation, for the cost of a slight risk of not meeting the latency requirement in the non-optionality edge cases which can be addressed by either investment into compute power or by using real-time Linux kernel. As both protocols are implemented within the Byzantine-resilient architecture, the choice of which protocol to use can be made at deployment.

4.9 Deployment and Purple Teaming

In the period August to November 2022, the system underwent a purple team exercise conducted by a team of experienced hackers from Sandia National Laboratories (SNL). This exercise, conducted at Pacific Northwest National Laboratory (PNNL), aimed to assess and improve the system's resilience against advanced cyber

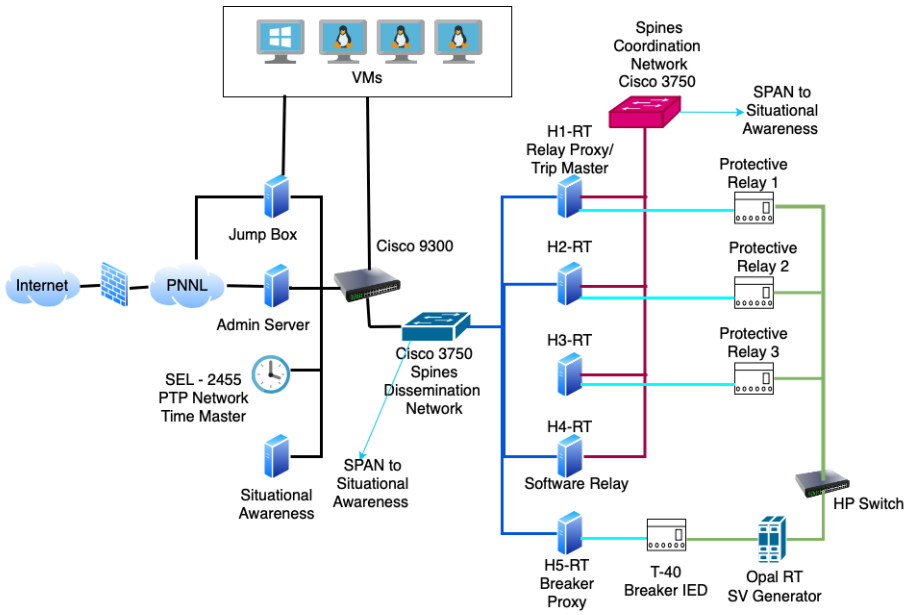
attacks. The attackers employed a combination of vulnerability research techniques and the purple team approach, a collaborative penetration testing methodology where attackers and defenders work together to uncover system strengths and weaknesses.

Purple Teaming: The expert hackers from SNL team proposed a purple team approach. Unlike traditional red and blue team exercises, where teams operate independently and report findings separately, purple teaming is a collaborative approach in cybersecurity where red teams (attackers) and blue teams (defenders) work together to improve an system's security posture. For this exercise, the blue team is us, the system developers from Johns Hopkins University (JHU). Purple teaming facilitates a continuous cycle of testing and improvement. The red team launches attacks, while the blue team responds and strengthens defenses based on the findings. This iterative process helps in rapidly identifying and addressing vulnerabilities.

System Setup: The system configuration for the exercise featured four relay nodes (H1-RT to H4-RT) interconnected according to the network diagram shown in Figure 4.19a. This setup included three physical hardware relays provided by Siemens, GE, and Hitachi Energy (Figure 4.19b), along with a software relay (hosted on H4-RT) . These relays played a pivotal role in the exercise by integrating with the Byzantine-resilient system's infrastructure to replicate realistic operational scenarios. The circuit breaker is controlled through a IED (T-40 in the Figure 4.19a) with its proxy hosted on H5-RT. A key component of the system is Opal RT, which generated Sampled Values and monitored breaker state. The Virtual Machines (VMs) represent the enterprise network and hosted multiple Windows and Kali Linux machines. The Kali Linux machines used by the SNL team to develop and launch attacks. Additionally, we worked with SNL team to develop some attacks and integrated them into our system.

Exercise: The primary focus of the exercise was to evaluate if the system's security goals could be compromised. To establish a baseline for normal system behavior, the assessment team meticulously defined expected operational norms and performed functionality testing. Deviations from this baseline during exercise would be flagged as potential security vulnerabilities.

In total, over 140 tests were conducted by the SNL team. These tests include attacks on the system from inside and outside the system. In the inside attack scenarios, the SNL team is give complete root access to a relay node and launched attacks from the node, representative of a compromise. They also attacked the system from outside i.e., enterprise network and virtual machines. Attack scenarios included preventing message processing, injecting malicious content, denial of service attacks, man-in-the-middle attacks and attempting to gain control over critical system functions such as breaker operations etc. Furthermore, the tests covered a range of scenarios including network attacks, and testing the core Byzantine Fault Tolerance (BFT) algorithms both within ($f \leq f_s$) and beyond standard operational conditions ($f > f_s$). Tests were performed across multiple protective relay vendors present at PNNL, including



(a) PNNL Testbed set up for Red Team experiments

(b) PNNL Testbed with physical relays from Hitachi Energy (Left), GE (top-right) and Siemens protection relays (bottom-right)

an emulated version of the relays, which were later released as open-source with our software. During the exercise the system operated correctly at the expected performance levels.

In the early stages of functionality testing, the SNL team identified one minor issue in the breaker proxy caused by a misalignment between design and implementation. This discrepancy led to a race condition in the code. Once detected, we quickly addressed the issue and deployed a patched version of the breaker proxy. Following the fix, all subsequent red team tests were successfully withstood, with no further issues arising. This experience highlights the advantages of a purple team approach over traditional, time-limited red team exercises. While red team testing typically focuses on a limited set of test cases within a fixed time frame, the purple team's approach was different. Over a long duration, through continuous collaboration and an iterative process, we were able to conduct over 140 tests. This allowed for a much more rigorous evaluation of the system.

While Byzantine resilience can effectively mask failures and attacks, it is important to present the system's knowledge about ongoing attacks and anomalies to the operator to allow them to react and take action. Throughout all the tests, the situational awareness module (described further in Section 6) effectively detected attacks, demonstrating that the machine learning approach is the most effective choice for attack detection.

Conclusion: Upon completion of the exercise, the SNL team concluded in their report that 'the target system under review proved to be resilient and capable of maintaining operations even under advanced attack conditions'. While this result does not mean that given more time, the hacker team would not have been able to cause damage, it shows that there is a significant difference between current industry best practices and a research-based solution designed to withstand sophisticated system and network attacks. This purple team exercise highlighted the system's robustness against sophisticated cyber threats and provided valuable insights for enhancing its security posture in future deployments.

Chapter 5

End-to-End Byzantine-Resilient SCADA

The Byzantine-resilient architecture and protocols for substations outlined in Chapter 4 provide strong resilience within the substation itself, but they function as isolated systems that primarily support real-time protective operations. In practice, substations need to communicate monitoring updates to both substation and control center operators, as well as execute commands from them. While there are Byzantine-resilient system architectures designed for the control center (as discussed in Chapter 2.2), these solutions remain isolated and do not adequately address the resilience needs beyond the control center SCADA Master.

To overcome these limitations and promote cohesive operation across the entire grid, we must develop a unified end-to-end system that integrates multiple subsystems and meets all operational requirements. This includes ensuring compatibility across different protocols, interoperability among various manufacturers, responsiveness to grid dynamics, and support for diverse types of operations while supporting full threat model and meeting the service properties.

In this chapter, we introduce the first end-to-end Byzantine-resilient architecture and protocols that facilitate seamless interoperability among SCADA components, creating an all-encompassing system framework. This framework will not only enhance overall resilience but also ensure that all operational, security, and performance requirements discussed in Chapter 1 and Chapter 3 are fully addressed.

5.1 High Level Overview

Grid infrastructures, including substations, RTUs, and PLCs, are highly diverse. They vary in design, functionality, operational requirements, and criticality. Each component serves a distinct role within the grid, with differing levels of importance depending on the system's needs and the specific substation's function. Given this

diversity and the varying operational demands, we propose a two-pronged approach to our solution design: one tailored for simpler, less complex systems, and another designed for more sophisticated, high-resilience setups. This approach ensures that both basic and advanced infrastructure components can be seamlessly integrated while meeting their specific security, performance, and operational needs.

In case of simpler systems, such as PLCs and RTUs without local SCADA directly controlled by control center, to ensure effective management and integration into larger system, we introduce a proxy based approach (illustrated in Figure 5.1). This proxy acts as an intermediary, allowing devices that may not natively support full SCADA functionality to interact with rest of the grid infrastructure.

For more sophisticated setups, such as substations with Byzantine-resilient SCADA systems (as described in Chapter 4), additional components are necessary to enhance the architecture. Specifically, the introduction of two key modules—the Control Center Connector (CC Connector) and the IED Connector. They ensure that these substations remain seamlessly integrated with both local and remote control systems. These modules are depicted in Figure 5.2.

At a high level, the connectors serve as the communication pathways between the control center and the substation, facilitating both the transmission of control messages and status/monitoring updates. The Control Center Connector (CC Connector) routes messages between the control center and the substation, while the IED Connector, located within the substation, performs authentication and verification of those messages. Once validated, the IED Connector forwards the messages to the proxy, which then translates them into a format that is compatible with the substation's SCADA protocols (such as IEC61850, Modbus). This architecture ensures smooth interoperability across various subsystems, regardless of differing protocol standards or manufacturers.

Control Center Connector: The Control Center Connector (CC Connector) serves as the critical communication bridge between the control center and the substation, ensuring the seamless flow of control messages and monitoring updates. Its primary responsibility is to route commands issued from the control center to the substation components, and vice versa, facilitating bi-directional communication over the wide-area network (WAN) or the substation's network (as shown in Figure 5.2). The connector efficiently forwards control action messages from the control center, ensuring that they reach the substation's components in a timely manner. Likewise, it relays substation updates back to the control center SCADA Masters using the intrusion-tolerant network service deployed over the WAN.

While the connector plays a pivotal role in message routing, it does not generate or alter messages. This means that a Byzantine fault affecting the connector will not result in the creation of invalid action messages; instead, the fault may cause the connector to drop, delay, or misroute messages. In more severe cases, it could be exploited to launch Denial of Service (DoS) attacks. To counter such risks, we propose a redundancy strategy involving the deployment of two or three connectors.

By introducing redundant connectors, we ensure that if one connector fails, the remaining connectors can continue to route messages, preserving communication integrity. The use of two connectors offers a basic level of fault tolerance, in the event of a failure of one connector, the second can continue to route messages. However, this setup may still leave the system vulnerable to simultaneous failures or attacks targeting both connectors. For higher fault tolerance and proactive recovery, a third connector can be added, ensuring that the system can withstand multiple failures or support proactive recovery while maintaining communication integrity. The simple design of the connector, which focuses on robust routing without complex message creation or interpretation, allows for rapid recovery and minimal overhead. While adding a third connector increases redundancy and robustness, it also introduces additional complexity and management overhead. While our system can support both, the choice between two or three connectors may depend on the specific resilience requirements of the system, with three connectors providing higher fault tolerance but at the cost of increased resource use and slightly more complex management.

IED Connector: The IED Connector serves as a key security and validation component within the substation's communication infrastructure. Its primary role is to authenticate and validate incoming control messages from the control center or substation HMI before they are forwarded to the relay proxy for further processing. Authentication involves verifying the appropriate cryptographic signatures and validation involves checking that the command is well-formed, fresh, and intended for its relay. Thus, the IED Connector ensures that only valid, authenticated messages are allowed to interact with relays, protecting the system from potential cyberattacks and erroneous commands.

Once a message is authenticated and validated, the IED Connector passes it to the proxy, which is responsible for translating the command message into a format compatible with the substation's SCADA protocols (such as DNP3, Modbus, IEC61850 MMS, or other protocol variations). This translation is essential for ensuring that different components, often from various manufacturers, can communicate effectively and interoperate.

The IED Connector, like the CC Connector, is designed with simplicity in mind. It focuses solely on message authentication and validation, ensuring that each message meets predefined security and integrity criteria. It does not interpret or execute control actions directly but ensures that only legitimate actions reach the relay proxy.

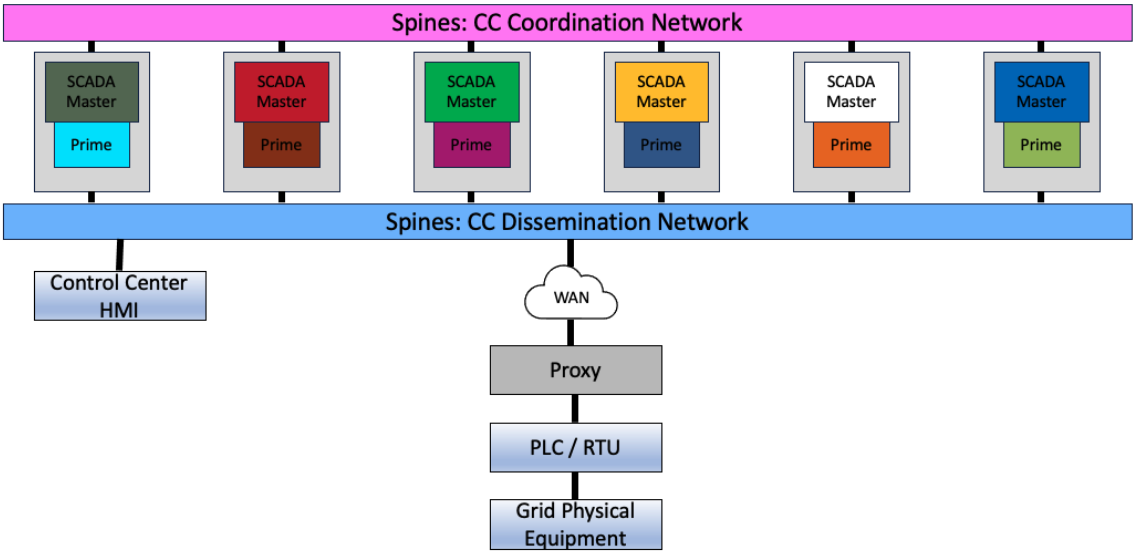


Figure 5.1: End-to-end Byzantine-resilient architecture for simple RTUs or PLCs

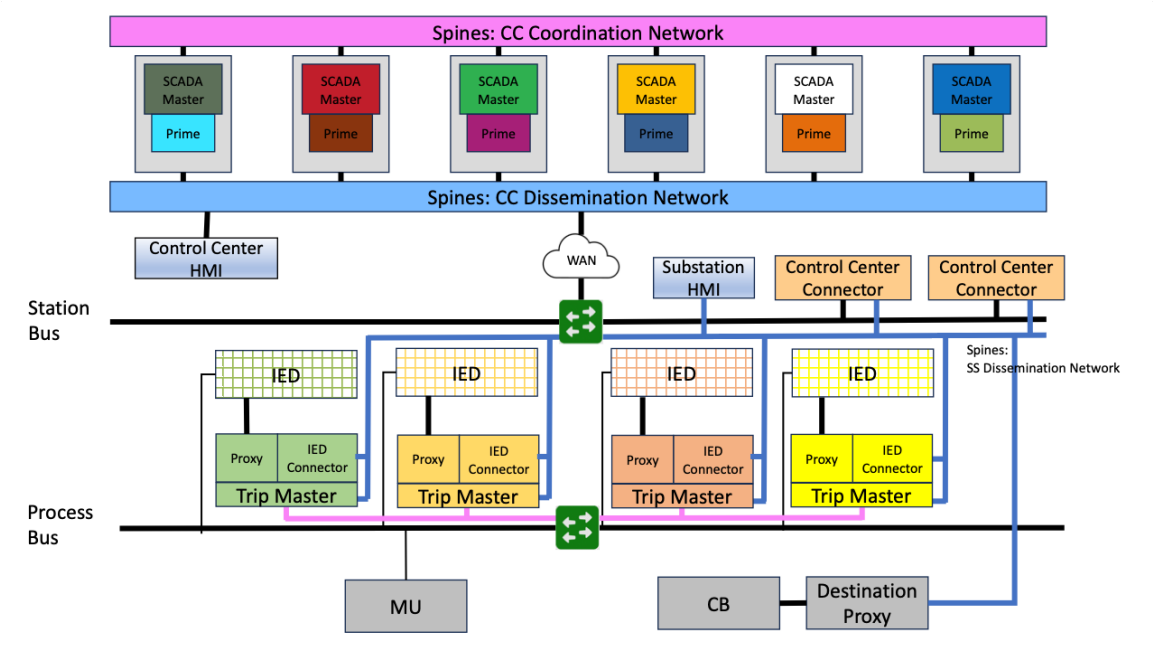


Figure 5.2: End-to-end Byzantine-resilient architecture for sophisticated substation with a local Byzantine-resilient system

5.2 End-to-End Byzantine-Resilient Architecture Design to Support Real-time Grid Operations

For a robust and secure power grid, the end-to-end architecture must be capable of supporting real-time SCADA operations: **commands from the control center, commands from a substation, substation autonomous protection operations, and status updates**.

Control Center Command Operations: The first category of operations involves commands issued from the control center to the substations. These centralized commands are essential for coordinating the overall management of the grid, enabling tasks like switching circuits, adjusting settings, and performing maintenance. The control center monitors grid conditions and, when necessary, sends commands to substations to maintain the balance of power distribution, optimize efficiency, and address issues like load fluctuations, faults, or grid congestion. This centralized approach provides high-level oversight and ensures a coordinated response to grid-wide events.

Substation Command Operations: In addition to the commands from the control center, substations must also be able to respond to the commands issued by substation operators through Human-Machine Interfaces (HMIs), where available. Sometimes substation operator interface can just be a button. This control allows operators to manage equipment, perform real-time adjustments, and react to immediate operational needs without waiting for instructions from the control center. This provides flexibility and enables a quicker response to situations that require on-site intervention, such as equipment malfunctions, maintenance tasks, or changes in local conditions.

Substation Autonomous Protection Operations: Substation autonomous protection operations are designed to detect grid faults and trigger protective actions that protect grid assets and maintain stability. A rapid response is crucial to minimize damage and prevent further disturbances. Consequently, industry standards from organizations like IEEE and IEC mandate that these critical substation autonomous protective actions must adhere to a strict real-time latency requirement of a quarter-power cycle (4.167 ms in a 60 Hz system) [8].

Status Updates: Status updates are the continuous flow of information from field devices back to the control center and substation HMIs, ensuring that operators and automated systems have real-time visibility into grid state. These messages include information on the operational state of equipment (e.g., circuit breakers, transformers, and protective relays), as well as metrics like voltage, current, and power flow. Status updates are crucial for maintaining situational awareness, enabling operators to monitor grid health, assess performance, and detect potential issues

before they escalate. They also provide feedback on the outcomes of control actions, allowing the control center and substation to verify whether commands have been successfully executed.

Together, these SCADA operations, control center commands, substation commands, substation autonomous protection operations, and status updates, form the backbone of grid monitoring and control. In the following sections, we outline the specific steps involved in each operation, using the trip action command as a case study to demonstrate how the architecture enables seamless Byzantine-resilient execution and integration across the system.

5.3 Control Center Command Operations and Status Updates

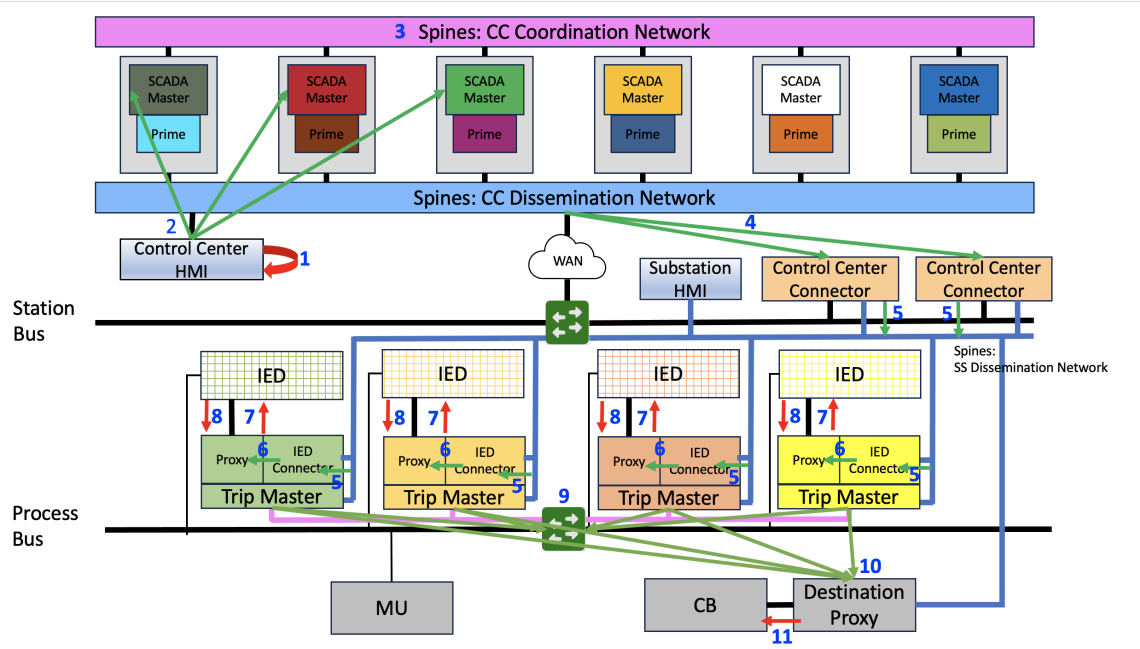


Figure 5.3: Control center to substation command operation steps

The system must support operation types that involve action commands issued by the control center to substations. Once an operation is completed, the substation must send an update back to the control center to confirm the operation's status. Additionally, if the substation is equipped with a Human-Machine Interface (HMI), the operation update should also be reflected there. This ensures consistent and accurate information across all operational levels.

In the following sections, we describe the process in two stages, using a trip command as an example. First, we will cover the flow of the trip action command from the control center to the substation breaker (Figure 5.3). Second, we will describe the flow of breaker status updates from the substation to the control center (Figure 5.4).

Stage I: Issuing the Control Center Command

The first stage involves the transmission of a trip command from the control center to the substation and its execution. This process is illustrated in Figure 5.3, with each step numbered for clarity. Below is a detailed breakdown:

1. **Control Center HMI Issuing the Trip Command**
    - **Step 1:** The control center HMI issues the trip command using a SCADA protocol (e.g., DNP3).
    - **Step 2:** In our Byzantine-resilient system design, the insecure SCADA protocol messages are not transmitted over the network. Instead, a proxy that is part of control center HMI, encapsulates and signs the command. The signed command is then wrapped in the format required by the intrusion-tolerant Spines network and transmitted over the Spines CC Dissemination network to the SCADA Master replicas at the control center.
2. **SCADA Masters Coordination and Consensus**
    - **Step 3:** The SCADA Masters verify the command, coordinate over the Spines CC Coordination network and achieve consensus using the Prime replication engine.
    - **Step 4:** Once consensus is reached, the SCADA Masters threshold-sign the command. The threshold-signed command is sent over the Spines CC Dissemination wide-area network to the substation's control center connector.
3. **Routing and Validation at the Substation**
    - **Step 5:** The control center connector routes the command to the Intelligent Electronic Device (IED) connectors through the substation Spines dissemination network.
    - **Step 6:** The IED connector verifies the command using both the SCADA Masters' threshold signature and the control center HMI's signature. Upon successful validation, the message is unpacked and handed to the proxy for further translation.
4. **Relay Command Processing**

- **Step 7:** If the relay in the substation uses the SCADA protocol IEC61850 GOOSE, the proxy converts the command into an IEC61850 GOOSE message. The message is issued to the relay (directly connected by wire to the proxy). Multiple proxies can be used to support any SCADA protocol utilized by the IEDs.
- **Step 8:** The relay processes the proxy's GOOSE message and issues the appropriate trip GOOSE command. This message is received by the proxy (directly connected by wire to the relay).

5. **Substation Byzantine-resilient Action Generation**

   - **Step 9:** The proxy forwards the trip message to its Trip Master. The Trip Masters use either the Arbiter Protocol or the Peer Protocol to coordinate and generate a valid trip command for the breaker proxy.
   - **Step 10:** The valid trip command is sent on the substation Spines dissemination network to the breaker proxy.

6. **Action Execution**

   - **Step 11:** The breaker proxy, connected directly by wire to the circuit breaker, validates the message and publishes an IEC61850 GOOSE message to execute the trip. The circuit breaker opens the circuit.

Stage II: Substation Status Updates

The second stage involves the transmission of status updates from the substation to the control center, as shown in Figure 5.4. Below is a detailed breakdown:

1. **Update Generation**

   - **Step 12:** The circuit breaker generates a standard IEC61850 GOOSE message to confirm the change of status.
   - **Step 13:** The destination proxy receives this message, signs it, and converts it into the Spines format. The signed message is then broadcast over the substation Spines dissemination network and received by three key components: the Trip Masters, the substation HMI, and the CC Connectors.

2. **Update for Relays**

   - **Step 14: Trip Masters** The Trip Masters receive the message, verify and validate the message based on the Arbiter Protocol or the Peer Protocol as explained in Chapter 4. Upon successful validation, the message is unpacked and forwarded to the proxy.

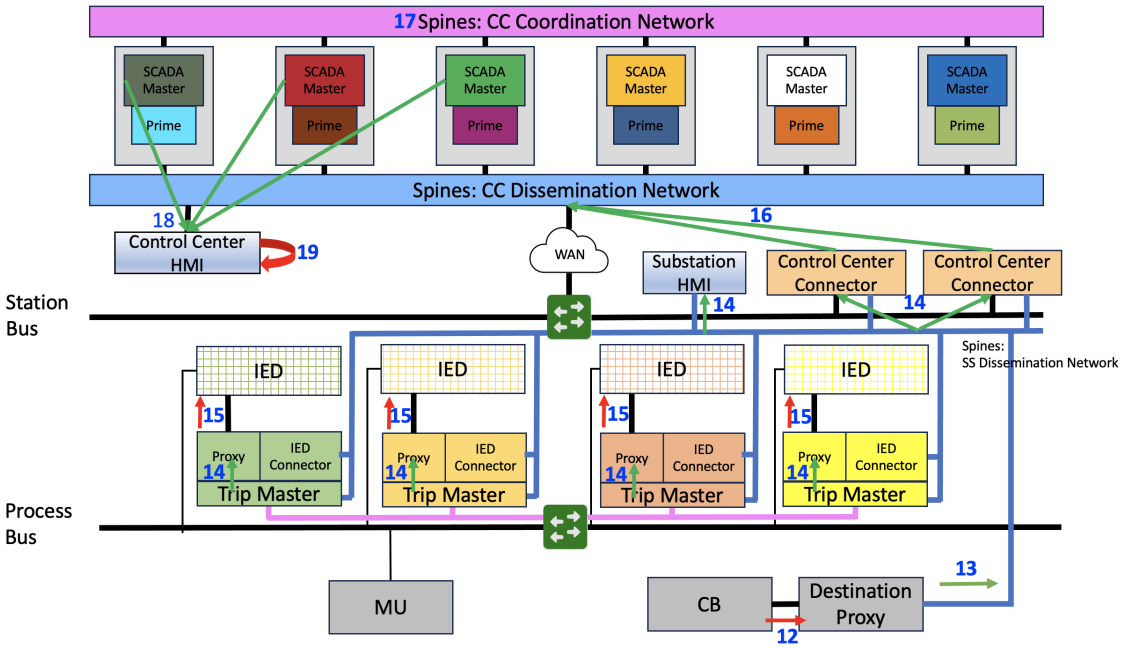


Figure 5.4: Substation to control center status update operation steps

- **Step 15:** The proxy translates the message and publishes a GOOSE message that can be processed by relays or IEDs.

3. **Updating Substation HMI**

   - **Step 14: Substation HMI** The substation HMI receives the update, verifies the message and updates the HMI view for the substation operator.

4. **Routing Update to Control Center SCADA**

   - **Step 14: Control center connectors** The control center connectors route the update message to SCADA Masters through the Spines CC Dissemination network.

5. **SCADA Masters Coordination and Consensus**

   - **Step 16:** The SCADA Masters receive the update and verify it.
   - **Step 17:** The SCADA Masters use Prime to achieve consensus on the update. Once consensus is reached, the SCADA Masters threshold sign the update and transmit over the Spines CC Dissemination network to the control center HMI.

6. **Updating Control Center HMI**

- **Step 18:** The control center HMI proxy verifies the update and if necessary translates it into a SCADA protocol that the HMI can process (e.g., DNP3).
- **Step 19:** The control center HMI view is updated, ensuring that the SCADA system remains consistent across the entire system.

Note: Even in the absence of a control center command (Stage I), Stage II can still be triggered and process updates or monitoring messages from the field without requiring any prior steps from Stage I.

5.4 Substation Command Operations

The system must support operation types that involve action commands issued by the substation operator through a substation HMI. Once an operation is completed, the substation must send an update back to the substation HMI to confirm the update. Additionally, the update should also reflect in control center HMI ensuring consistent views of system across all operational levels.

In the following sections, we describe the process in two stages, using a trip command as an example. First, we will cover the flow of the trip action command from the substation HMI to the substation breaker (Figure 5.5). Second, we will describe the flow of breaker status updates from the substation to the substation and control center HMIs (Figure 5.4). As the second half is identical to the Stage II (steps 12 - 21) of 5.3, we only describe Stage I in detail.

Stage I: Issuing the Substation Command

The first stage is the transmission of a trip command from the substation HMI to the breaker and its execution. This process is illustrated in Figure 5.5, with each step numbered for clarity. Below is a detailed breakdown:

1. **Substation HMI Issuing the Trip Command**
   - **Step 1:** The substation HMI issues the trip command using a SCADA protocol.
   - **Step 2:** In our Byzantine-resilient system design the insecure SCADA protocol messages are not transmitted over the network. Instead, a proxy that is part of the substation HMI encapsulates and signs the command. The signed command is then wrapped in the format required by the intrusion-tolerant Spines network and transmitted over the substation Spines dissemination network to the IED Connectors.
2. **IED Connectors Validating Trip Command**

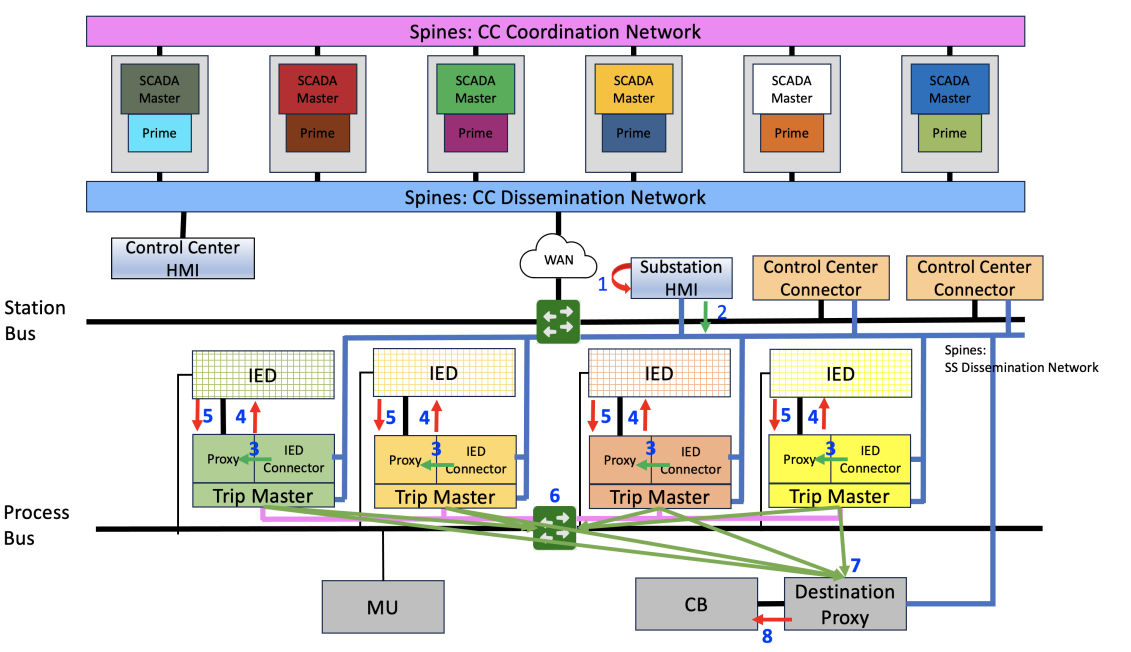


Figure 5.5: Substation command operation steps

- **Step 3:** The IED connector verifies the command using the substation HMI's signature. Upon successful validation, the message is unpacked and handed to the proxy for further translation.
- **Step 4:** If the relay uses the SCADA protocol IEC61850 GOOSE, the proxy converts the command into an IEC 61850 GOOSE message. The GOOSE message is issued to the relay (directly connected by wire to the proxy). Multiple proxies can be used to support any SCADA protocol utilized by the IEDs.

3. **Relays Command Processing**

- **Step 5:** The relay processes the proxy's GOOSE message and issues the appropriate trip GOOSE command. This message is received by the proxy (directly connected by wire to the relay).

4. **Substation Byzantine-resilient Action Generation**

- **Step 6:** The proxy forwards the trip message to the Trip Master. The Trip Masters use either the Arbiter Protocol or the Peer Protocol to coordinate and generate a valid trip command for the breaker proxy.
- **Step 7:** The valid trip command is sent on the substation Spines dissemination network to the breaker proxy.

5. **Action Execution**

   - **Step 8:** The breaker proxy, connected directly by a wire to the circuit breaker, validates the message and publishes an IEC61850 GOOSE message to execute the trip. The circuit breaker opens the circuit.

Stage II: Substation Status Updates

The second stage involves the transmission of status updates from the breaker to the substation relays, substation HMI and control center HMI. This procedure is identical to the Stage II of 5.3 (Steps 12 - 19) explained above.

Note: Even in the absence of a substation command (Stage I), Stage II can still be triggered and process updates or monitoring messages from the field without requiring any prior steps from Stage I.

5.5 Substation Autonomous Protection Operations

The system must support automated protective action commands generated by relays within the substation. These operations have an exact real-time latency requirement of quarter-power cycle (4.167ms in a 60Hz system). Once an operation is completed, the substation must send an update back to the substation HMI to confirm the update. Additionally, the update should also reflect in the control center HMI ensuring consistent views of the system across all operational levels.

In the following sections, we describe the process in two stages, using a trip command as an example. First, we will cover the flow of the trip action command from the relays to the substation breaker (Figure 5.6). Second, we will describe the flow of breaker status updates from the substation to the substation and the control center HMIs(Figure 5.4). As the second half is identical to the Stage II (steps 12 - 21) of 5.3, we only describe Stage I on detail.

Stage I: Issuing the Autonomous Protection Operation

The first stage is the transmission of a trip command from the substation relays to the breaker and its execution. This process is illustrated in Figure 5.6, with each step numbered for clarity. Below is a detailed breakdown:

1. **Relays Issuing Trip Command**

   - **Step 1:** The relays continuously monitor the grid state through Sampled Value messages. Upon detecting a risky situation, such as measured grid parameters breaching safety envelope, the relays issue a trip IEC61850 GOOSE command. The trip command is received by the proxies (directly connected by wire to the relays).

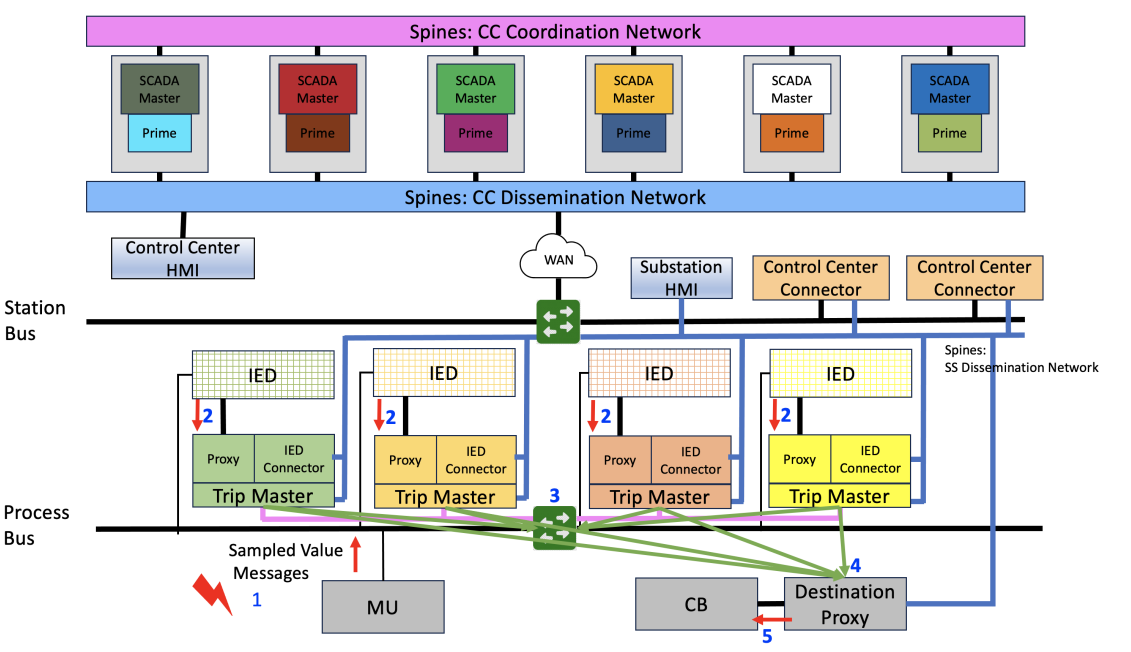


Figure 5.6: Substation autonomous protection operation steps

2. **Substation Byzantine-resilient Action Generation**

   - **Step 2:** The proxy forwards the trip message to the Trip Master. The Trip Masters use either the Arbiter Protocol or the Peer Protocol to coordinate and generate a valid trip command for the breaker proxy.
   - **Step 3:** The valid trip command is sent on the substation Spines dissemination network to the breaker proxy.

3. **Action Execution**

   - **Step 4:** The breaker proxy, connected directly by wire to the circuit breaker, validates the message and publishes an IEC61850 GOOSE message to execute the trip. The circuit breaker opens the circuit.

Stage II: Substation Status Updates

The second stage involves the transmission of status updates from the breaker to the substation relays, substation HMI and control center HMI. This procedure is identical to the Stage II of 5.3 (Steps 12 - 19) explained above.

Note: Even in the absence of a substation command (Stage I), Stage II can still be triggered and process updates or monitoring messages from the field without requiring any prior steps from Stage I.

5.6 Service Properties

The system is designed to enable end-to-end Byzantine-resilient grid operations. Formally, the following guarantees apply, based on the system model, threat model, and assumptions outlined in Chapter 3:

- **Substation Safety:** If a valid protective operation is executed by the breaker node, at least one correct relay node issued that action within the last reaction quarter.
- **Substation Real-Time Reaction:** If the grid state necessitates a protective action due to an event, that action will be issued for that event's reaction quarter.
- **SCADA Safety:** If two correct SCADA Master replicas execute the i^{th} operation, then those operations are identical.
- **SCADA Bounded-Latency Reaction:** If a correct system component initiates an operation, the latency for that operation to be executed by a correct SCADA Master replica is upper bounded.

Note that the correctness proofs for the first two guarantees, Substation Safety and Substation Real-time Reaction, are in Chapter 4.5. While we can fully prove Substation Safety, for Substation Real-time Reaction, we can only prove the issuance of required protective action by relay nodes for that event. We demonstrate by rigorous evaluation through benchmarks in Chapter 4.6 the extent to which the real-time latency requirement is met under different operating conditions in different relevant environments.

We provide SCADA Safety and SCADA Bounded-Latency Reaction properties through Safety and Bounded Delay definitions of Prime as originally defined in [30] and later in [51].

5.7 End-to-End Byzantine-Resilient Architecture for Large-Scale Deployments

The proposed end-to-end Byzantine-resilient system framework is designed to enable large-scale deployment by emphasizing compatibility, abstraction, modularity, and scalability. Our approach not only provides Byzantine resilience to SCADA systems but also provides flexibility, adaptability, and ease of deployment across diverse environments. In this section, we discuss the implications of the proposed architecture, highlight its key features, and summarize its potential impact on SCADA system deployments.

Architecture Abstraction for Seamless Deployment A core component of our system design is the use of architecture abstraction, which serves as a key enabler of seamless deployment. By decoupling the grid devices from the intricate mechanisms that ensure Byzantine fault tolerance and resilience, the proposed architecture simplifies the operation for individual end devices like relays, HMIs and breakers. For instance, Remote Terminal Units (RTUs) are unaware of the number of SCADA Master replicas, while other devices like breakers do not need to be informed of the underlying relay nodes in substations. This abstraction shields end devices from the complexity of the deployed Byzantine-resilient mechanisms deployed.

Additionally, the use of threshold cryptography enables the change of configurations, further enhancing the flexibility of the architecture. For instance, the number of relay nodes using Peer protocol, can be adjusted without impacting the functionality of the breaker node. This level of abstraction not only improves the resilience of the system but also allows for incremental changes to be made to the architecture without disrupting the overall operation of the system. The resulting simplified operational model enables end devices to remain unchanged and focus solely on their primary functions, such as measuring grid states or executing commands, rather than dealing with the complexities of the system.

Modularity for Independent Operation and Incremental Deployment Another notable feature of the proposed architecture is its modularity. The system is designed as a system of systems, where each sub-system can be deployed independently and also integrate and inter-operate with the larger system. This modularity is particularly beneficial for organizations that need to deploy Byzantine-resilient SCADA systems without overhauling their entire infrastructure. The ability to deploy these systems step by step allows for a more manageable transition towards enhanced security and resilience.

Compatibility by Supporting Wide Range of SCADA Protocols: Our architecture also addresses a common challenge in SCADA system design - protocol heterogeneity. SCADA systems often consist of devices from different manufacturers, each potentially using different communication protocols. Despite this variability, our design ensures that devices, regardless of their manufacturer and communication protocol, can still interoperate within the Byzantine-resilient system. This is achieved through proxy-based design that ensures insecure protocols are not transmitted on the network, maintaining the integrity of communications without necessitating changes to the end devices or protocols at the endpoints. This wide protocol support ensures that our Byzantine-resilient system framework can be applied in a variety of real-world SCADA environments. In fact, this ensures that our system remains relevant and effective in the face of ever-changing and evolving industry standards.

Scalability and Future-Proofing: The scalability of the proposed architecture is another significant advantage. The abstraction and modularity of the design enable it to scale easily, accommodating the addition of new end devices or expansion of the SCADA network without requiring significant changes to the underlying system

architecture. Furthermore, the system's ability to evolve as the number of endpoints increases ensures that the Byzantine-resilient system can grow in parallel with the demands of modern SCADA environments.

Advanced Deployment Configurations to Support Different Levels of Resilience: An essential feature of the proposed end-to-end Byzantine-resilient system framework is its ability to support advanced deployment configurations tailored to varying levels of resilience based on the unique needs of the utility or operator. Drawing from the findings in the paper "Network-Attack-Resilient Intrusion-Tolerant SCADA for the Power Grid", our framework provides flexibility to deploy advanced configuration with multiple control center and/or data centers. More details on specific deployment configurations' resiliency capabilities are discussed in [51].

In conclusion, the proposed end-to-end Byzantine-resilient system framework offers a robust solution for enhancing the resilience of large-scale SCADA systems. By incorporating abstraction and modularity, the architecture simplifies the operation of individual end devices while ensuring Byzantine resilience and seamless interoperability across a variety of SCADA protocols. This flexibility allows the system to be deployed incrementally, supporting both legacy and modern devices without disrupting existing infrastructure. Its ability to scale and adapt to changing requirements makes it an ideal choice for power grids and other critical infrastructure. Ultimately, this architecture not only strengthens the security posture of SCADA systems but also provides a future-proof foundation that can evolve with emerging threats and technological advancements in grid control systems.

Chapter 6

Machine Learning based Intrusion Detection for Situational Awareness

Byzantine resilience or Intrusion tolerance techniques are specifically designed to maintain system correctness and operational continuity in the face of successful intrusions, emphasizing resilience rather than immediate intrusion detection or situational awareness. While intrusion detection systems (IDS) focus on identifying and alerting operators to potential attacks or anomalous behavior, they operate within a limited scope, mainly for detection and response. These two techniques, Byzantine resilience and Intrusion detection, fulfill distinct roles, they are complementary, contributing to the overall security posture of the system. Hence, for our Byzantine-resilient systems, we designed, developed, and deployed a machine learning-based IDS that enables real-time detection and alerts operators to potential intrusions or anomalous behavior in the system.

Traditional SCADA systems rely on conventional intrusion detection techniques, such as rule-based, signature-based, and packet inspection methods. However, these approaches are often inadequate in the face of evolving threats, as evidenced by numerous documented security incidents [77]. Even machine learning-based IDS deployed in traditional SCADA systems typically depend on deep domain knowledge and insecure protocols that expose sensitive payload data in network messages [78].

As we transition to a Byzantine-resilient system, we move away from these traditional practices, replacing insecure SCADA communication protocols with secure, encrypted alternatives, employing redundant components, and utilizing Spines intrusion-tolerant network services. This strategic shift renders conventional IDS approaches insufficient, necessitating a new, more flexible solution that provides enhanced situational awareness and security in the context of modern, evolving threats.

Moreover, the IDS must be carefully designed to operate without introducing vulnerabilities that could compromise the integrity of the grid infrastructure or impact-

ing the grid operations. This requires seamless integration of the IDS with SCADA and other Operational Technology (OT) components, while adhering to the stringent operational constraints

To address these challenges, we have developed a machine learning-based IDS that operates independently of SCADA-specific protocol knowledge or unencrypted traffic. This system functions out-of-band, passively capturing network packets, ensuring minimal impact on SCADA operations. Given the critical nature of grid SCADA systems, our IDS is engineered to minimize both false positives (incorrectly classifying normal activity as an intrusion or anomaly) and false negatives (failing to detect a genuine intrusion or anomaly). Achieving the right balance between sensitivity and accuracy is complex, particularly in the dynamic SCADA environment. As such, it requires careful consideration of real-time detection, scalability, and overall system performance.

In the following chapters, we outline the design and implementation of our data collection, processing, and storage pipeline, followed by the feature engineering techniques and core machine learning modules employed.

6.1 Data Collection, Processing and Storage Pipeline

Our data collection, processing, and storage pipeline is designed to facilitate the efficient out-of-band analysis of network traffic. Figure 6.1 illustrates the deployment on IDS in and end-to-end Byzantine-resilient system. In our isolated substation system (shown in Figure 4.19a), network taps are strategically positioned to capture traffic from the testbed environment. This system is part of our purple teaming deployment described in Chapter 4.9, where the situational awareness framework consists of two distinct machine learning modules: a Network Intrusion Detection System (NIDS) utilizing network traffic and a physics-based state estimator leveraging data from electrical modules, such as Opal-RT, relays, and circuit breaker (CB) outputs [79]. In the rest of the chapter, we focus on the Network Intrusion Detection System (NIDS) that operates exclusively on network traffic tapped from network switches via Switched Port Analyzer (SPAN) to detect network attacks, intrusions and anomalies.

The architecture of the NIDS modules in any deployment is illustrated in Figure 6.2. In our testbed, network switches forward complete raw packets, but advanced switches in other deployments may forward only packet headers or aggregated traffic statistics to enhance scalability and performance. The captured network traffic is directed to a packet parser module, which serves as a data publisher for the transformer module. These network traces are sourced from real testbed environments, ensuring the data's authenticity and relevance.

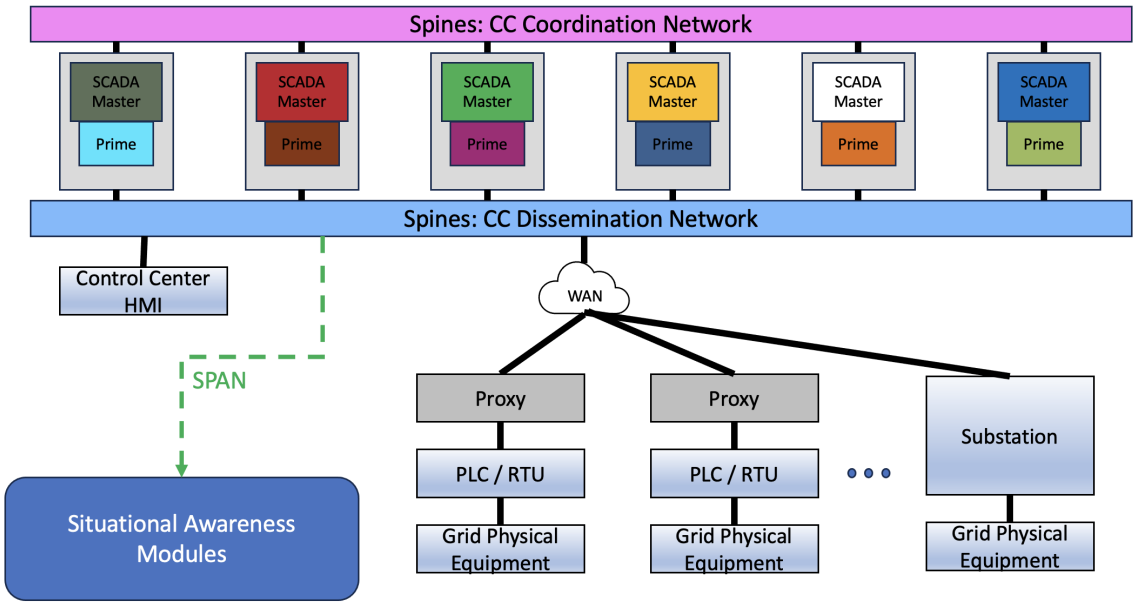


Figure 6.1: IDS integration with Spire

The transformer module plays a crucial role by transforming packet headers, capturing metadata, and computing traffic statistics, a process that partially encompasses feature engineering. These modules operate in real-time, with scalability considerations allowing for the deployment of multiple parser and transformer modules in parallel as needed. The transformed features are then forwarded to a database insertion daemon, which inserts the data into a storage database. This database serves as the foundation for offline model training.

During the prediction phase, the transformer module continues to manage data insertion into the database, while also forwarding transformed features to the core machine learning module for real-time prediction. This design ensures continuous situational awareness, providing rapid detection capabilities.

6.2 Feature engineering

Feature engineering is guided by two primary observations: First, due to the encryption of network traffic, only packet headers, encrypted payloads, and metadata are accessible for processing. Second, while system configurations remain largely stable, traffic patterns exhibit dynamic variability over time. Leveraging these insights, we adopt a dual approach that captures both the micro-level details of individual network packets (Packet-Level Features) and the macro-level patterns of traffic flows (Traffic-Level Features) that may indicate anomalies or intrusions.

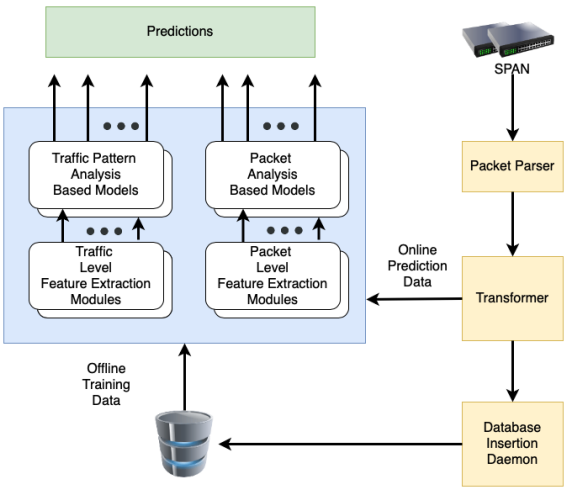


Figure 6.2: ML Data Pipeline

Packet-level feature engineering focuses on extracting detailed features from the headers of individual network packets. Each packet contains a variety of attributes, such as source and destination IP addresses, port numbers, and protocol types, that can offer insights into normal or malicious network behavior. The strength of this approach lies in its fine-grained analysis of network communications, enabling the identification of specific attack signatures at the packet level. For instance, in the case of port scanning attacks, the packet headers might reveal unusual patterns, such as a high number of SYN packets targeting a wide range of ports on a single destination IP, typically without completing the TCP handshake.

Traffic-level feature engineering, on the other hand, focuses on higher-level statistical patterns of network flows. These features capture metrics such as the number of packets directed to or originating from specific sources, port usage distributions, packet size variations, and other aggregate traffic statistics. We employ a rolling window technique to aggregate these metrics over time, which helps in identifying recurring traffic patterns while facilitating efficient storage management through periodic data purging. The advantage of this approach is its ability to provide a coarse-grained analysis of network activity, enabling the detection of flow-based anomalies. For example, in a Distributed Denial-of-Service (DDoS) attack, we often observe a large volume of packets originating from similar or identical source IP addresses, targeting a specific server or service.

This dual approach, focusing on both packet-level details and traffic-level patterns, ensures a comprehensive and robust feature engineering process. It effectively captures both the granular characteristics and temporal dynamics of network traffic, which are crucial for accurate and timely detection.

6.3 Machine Learning Models, Training and Prediction

At the heart of our Network Intrusion Detection System (NIDS) are a set of machine learning (ML) models that serve as the core decision-making engine, classifying network traffic as either benign or malicious. Given the dynamic nature of network traffic, especially in grid SCADA environments, the chosen ML models must be highly adaptable, capable of detecting emerging attack patterns, and scalable to evolving network behavior.

A key challenge in model selection stems from the highly imbalanced nature of the dataset, where the majority of network traffic is benign, and malicious traffic is relatively scarce. Acquiring labeled malicious traffic—especially for zero-day threats (novel attacks not previously observed) is inherently difficult. To address this, synthetic attack data is often generated to simulate abnormal network behavior. However, generating realistic attack data is fraught with challenges, as cyber threats are constantly evolving, and synthetic data may not capture the full spectrum of real-world attacks. Relying solely on synthetic data can also create a false sense of security, particularly if novel or zero-day threats are not adequately represented. To mitigate these risks, we turn to unsupervised anomaly detection models that do not require ground truth labels.

Unsupervised learning techniques, particularly one-class classification, are explored as a way to detect anomalies in network traffic. This method focuses solely on identifying deviations from the normal behavior established during training, rather than relying on predefined labels for different types of attacks. Among the models considered for this task are one-class Support Vector Machine (SVM), Isolation Forest, Local Outlier Factor (LOF), Elliptic Envelope and Autoencoders. These models are well-suited for intrusion and anomaly detection in scenarios where the normal class significantly outnumbers the abnormal class.

One-Class SVM, for instance, seeks to find a boundary that encapsulates the good/benign data, and any data points falling outside this boundary are flagged as potential anomalies.

LOF is a density-based anomaly detection algorithm that identifies anomalies by comparing the local density of a point with the densities of its neighbors. The primary advantage of LOF is its ability to detect anomalies in data that do not fit into the global density pattern of the dataset. This makes it particularly useful for identifying subtle anomalies that may not be apparent in a broader context but are irregular in local neighborhoods of network traffic. For instance, LOF can detect abnormal traffic patterns in a specific flow, which might indicate an ongoing attack such as lateral movement.

The Elliptic Envelope model is a robust method for detecting outliers that is particularly effective when data follows a Gaussian (normal) distribution. It fits

an elliptic envelope around the majority of data points and flags data points that lie outside this envelope as outliers. The Elliptic Envelope model is well-suited for detecting attacks that manifest as deviations from a normal Gaussian-like distribution of traffic, such as botnet activity or DoS attacks or other sophisticated intrusion attempts that involve unusual communication patterns.

Isolation Forest model isolates anomalies by partitioning data recursively. It is efficient for identifying outliers, especially when the dataset is large and high-dimensional.

Autoencoders are neural networks that learn to compress and reconstruct the input data. Anomalous traffic that deviates from normal patterns will have a higher reconstruction error, allowing the model to flag it as suspicious. Since it is impractical to check every packet, we apply this technique to randomly sampled packets within a specific time interval window.

To enhance the robustness and reliability of anomaly detection, majority voting is implemented to combine the outputs of these different models. This ensemble approach helps mitigate the risk of false positives that could arise when relying on a single model, leading to more reliable detection of potential threats. By integrating multiple models, the system increases its ability to detect a wider range of anomalous activities, ultimately improving the security of the network.

As noted earlier, in isolated substation system (Chapter 4.9), in addition to the machine learning-based NIDS, the system also incorporated machine learning models for electrical state estimation. These models use Convolutional Neural Networks (CNN) for steady-state estimation and transient-state estimation. This integration provides a comprehensive view of both network and system states, combining cybersecurity insights with electrical system modeling to offer a more holistic monitoring approach in critical environments like grid SCADA systems.

Model training and Deployment

After completing feature engineering, we proceed with model training, including hyperparameter tuning, cross-validation, and evaluation using a small test dataset with synthetic attack traffic. We iterate through different model combinations and select the best-performing ensemble for deployment. The chosen models are then deployed in a real-time, passive, out-of-band mode, where they continuously monitor network traffic without interfering with the grid's control systems.

During prediction phase, once features are extracted from incoming network traffic, they are fed into the deployed ML models for classification. The models then determine whether the traffic is benign or malicious. This prediction process is designed to be efficient and low-latency, ensuring near-instantaneous detection of potential intrusions. The system's two detection types, packet-level analysis and traffic pattern analysis, complement each other. For instance, while packet-level detectors may fail to identify a DoS attack if the attack is masked by random traffic, traffic pattern analysis models can still detect the attack by observing unusual flow characteristics.

As network traffic evolves over time, the system can periodically retrain and up-

date the models offline using newer traffic data. This process ensures that the system remains resilient to data drift and adapts to new attack techniques, maintaining high levels of detection accuracy and minimizing false negatives.

In summary, the combination of unsupervised anomaly detection, ensemble learning, and integration with electrical state estimation provides a comprehensive, adaptive solution for real-time intrusion detection in dynamic network environments, enhancing both security and situational awareness in critical infrastructure systems.

6.4 NIDS in action

To evaluate the performance of our Network Intrusion Detection System (NIDS), we replicate a series of well-known network-level attacks, including port scanning, Denial-of-Service (DoS), ARP poisoning, and replay attacks, within the testbed environment shown in Figure 6.1. For each attack type, we systematically vary the attack parameters and assess the system's detection capabilities. A key feature of our hybrid NIDS is its ability to leverage the complementary strengths of both packet analysis-based models and traffic pattern-based models.

For example, port scanning attacks, which are often difficult for traffic pattern analysis models to detect due to their discrete, low-volume nature, are effectively captured by the packet analysis models. In contrast, replay or DoS attacks—which may evade detection by packet analysis-based models due to the use of previously captured packets—are successfully detected by the traffic pattern-based models.

As illustrated in Table 6.1, our hybrid NIDS achieves an impressive overall detection rate of 96.5%, combining the strengths of both detection methodologies. Specifically, packet-analysis-based models detect 89.2% of attacks, while traffic-pattern-based models identify 78.6% of attacks. Together, these models ensure high detection accuracy across a wide range of attack types.

	Packet-analysis based models	Traffic-pattern based models	Overall System
Detected Attacks / Total Attacks	25/28(89.2%)	22/28(78.6%)	27/28(96.5%)

Table 6.1: The table below shows the detection performance of the packet-analysis based models, traffic-pattern based models, and the overall system across a range of attacks in testbed illustrated in Figure 6.1

In conclusion, our machine learning-based Intrusion Detection System (NIDS) enhances the intrusion-tolerant SCADA architecture by providing advanced situational awareness, but it is not a essential module for the system's core Byzantine resilience functionalities. The proposed Byzantine-resilient techniques embedded into

the system ensure the continuous correct operation of the system, even in the event of a successful attack. However, the NIDS acts as a supplementary layer, offering real-time detection of potential threats.

Unlike traditional IDS solutions, which often struggle with detecting unknown or evolving threats, our ML-based approach customized to our Byzantine-resilient system is adaptable to emerging attack patterns, operates out of band and is not dependent underlying SCADA knowledge or protocols. These functionalities allow the NIDS to provide more accurate and responsive detection capabilities, improving the overall security posture of the system and enabling proactive detection of both known and novel threats.

Chapter 7

Implementation

We implement our solution modules, including the Intrusion Detection System (IDS) components, within the open-source Spire Toolkit [67], envisioning it as a comprehensive bag of tools for grid operators. This toolkit offers both flexibility and scalability, making it ideal for deploying Byzantine-resilient SCADA systems. The Spire Toolkit is designed to address the diverse and evolving security needs of modern power grid infrastructures.

While the original Spire system was primarily focused on securing control center operations, particularly Byzantine-resilient SCADA Master operations, we have extended its capabilities to encompass Byzantine-resilient subsystems, such as substations, as well as end-to-end system frameworks. This extension also integrates a Network Intrusion Detection System presented in Chapter 6. The Spire Toolkit supports a variety of SCADA system configurations — ranging from individual subsystems (like a single substation or control center) to fully integrated end-to-end systems (like few substations and control centers or entire grid). This adaptability empowers power grid operators to deploy customized solutions that meet both security requirements and operational demands, whether securing isolated subsystems or the entire grid infrastructure.

7.1 Supporting SCADA Communication Protocols

Grid components utilize a variety of SCADA protocols, which are inherently insecure. For any deployable Byzantine-resilient system, supporting these legacy protocols is essential. To achieve this, we employ a proxy-based approach, where SCADA components connect to proxies through a single wire connection and use insecure protocols on wire. The proxies, in turn, secure the communications and leverage an intrusion-tolerant network for secure data transmission.over network.

Initially, the RTU/PLC proxy supports two widely-used open standard SCADA protocols, Modbus and DNP3, to ensure broad compatibility with various PLCs and RTUs. As the system evolved, we extended support to include IEC61850 substation automation protocols of Generic Object Oriented Substation Event (GOOSE), Sampled Values (SV), and Manufacturing Message Specification (MMS).

To support these protocols, we leverage open-source libraries like pvbrowser [80], OpenDNP3 [81], and libiec61850 [82]. Over time, the system proxies can be extended to support additional SCADA protocols as needed. This enables our system to be future-proof in environments like grid where standards and protocols are ever changing and evolving.

For each supported protocol, the proxy spawns a dedicated process to manage all relevant RTU, PLC, or IED connections. Upon startup, the proxy reads its configuration file to determine its responsibilities, connects using the appropriate SCADA protocol, and begins collecting updates. For instance, if a proxy manages three Modbus and two DNP3 PLCs,it will spawn both Modbus and DNP3 processes. A proxy handling four DNP3 RTUs will only spawn the DNP3 management process. Similarly, for IEC61850, the proxy will support the necessary protocol-specific processes.

The IEC61850 GOOSE protocol is designed to facilitate fast, event-driven communication in substation automation systems, where real-time data exchange is critical for protection, control, and monitoring. It follows a multicast publisher-subscriber model. The publisher generates and sends GOOSE messages, while the subscriber receives these messages. A typical publisher could be a protection relay that detects events, such as faults or changes in grid parameters.

Our system implementation includes both the publishers (proxy to IED communication, and emulated relays) and subscribers (IED to proxy communication). Our implementations adhers to the GOOSE message and communication pattern standards. This ensures compatibility and seamless interoperability with different manufacturers and emulated power devices. The GOOSE Control Block (GCB) Reference of GOOSE message is central to this process, allowing the system components to uniquely identify and bind to specific control blocks within IEDs, ensuring that GOOSE messages are correctly sent and received. The publisher uses the GCB Reference to include event data, such as relay statuses or breaker operations, in the GOOSE message. On the subscriber side, the system components filters and subscribes to specific GOOSE messages based on the GCB Reference, ensuring that only relevant messages are processed. In fact, we leverage the GCB Reference conventions to identify hardware relays from different manufacturers and implemented our proxies to automatically detect the connected hardware or software relays in our testbed (Chapter 4.9).

When a new event occurs, the publisher sends a message containing the event data and a new state number. This message is retransmitted multiple times, with the retransmission period doubling until it stabilizes, typically at around 1 second. Once a new event occurs, the state number is incremented, and the retransmission process starts again. Our GOOSE publishers implementation adhere to this communication

standard.

In our testbed, we utilize Opal-RT grid models to generate and transmit waveform data as SV messages to physical hardware relays from various manufacturers. As physical hardware relays are costly devices, to reduce costs, we also opt for software relays explained further in Chapter 7.2.

7.2 Software Relays and Emulated Relays

As discussed in Chapter 4, our system integrates both hardware relays from various manufacturers and fully implemented software relays. The software relays in our system contain complete relay protection logic, offering a cost-effective solution. These software relays are utilized to reduce costs while maintaining the full functionality required for the system.

Software Relay: This custom-built software-based relay has been developed by the Pacific Northwest National Laboratory team [include project and grant details]. The codebase executes over a preemptive, real-time Linux kernel. The system has been extensively tested to ensure its reliability and to ensure it detects faults within detection quarter. The relay design is based on IEEE C37.91-2008 recommendations and is configurable using a JSON file. Further technical details can be found at [79].

Additionally, the Spire Toolkit provides emulated relays, which are open-sourced and designed to make relay based research more accessible, especially for environments where acquiring physical relays may be challenging or costly. These emulated relays, while lacking internal protection logic, are ideal for end-to-end performance benchmarking. Their simplicity allows researchers and industry professionals to focus on evaluating system performance without the need for expensive relay hardware, ensuring that even those without direct access to physical relays can still conduct meaningful experiments and further the line of research.

Emulated Relays: In the end-to-end benchmarks presented in Chapter 4.6 we utilize emulated relays. The emulated relays are controller through emulated merging unit called event generator (included in Spire toolkit). The emulated relays receive SV messages from event generator and based on the SV message data payload, they generate and publish IEC61850 GOOSE message. To generate GOOSE they spawn libIEC61850 based-publishers and operate as described in Chapter 7.1. This design supports two key functionalities: as there is no state estimation delay, it helps us benchmark end-to-end performance and we can emulate different relay faults.

7.3 pvbrowser-based HMIs Implementation

The Human Machine Interfaces (HMIs) provided in Spire Toolkit are implemented using the open-source pvbrowser SCADA software suite [80]. pvbrowser is a mature SCADA solution that has been used to manage a real power grid deployment in Romania spanning 10,000 square kilometers with 50 power switches. The pvbrowser HMI provides the system operator with the ability to define a graphical user interface (GUI) (e.g., including custom images, buttons, and knobs) that supports the target SCADA scenario.

However, as its design aligns with traditional SCADA it does not work out of box with our Byzantine-resilient systems. Hence, we re-architected the HMI to to communicate with the rest of our Byzantine-resilient SCADA system. We implement HMIs for multiple scenarios but would like to focus on integrated scenario developed to demonstrate end-to-end Byzantine-resilient system operations in the next Chapter 7.4.

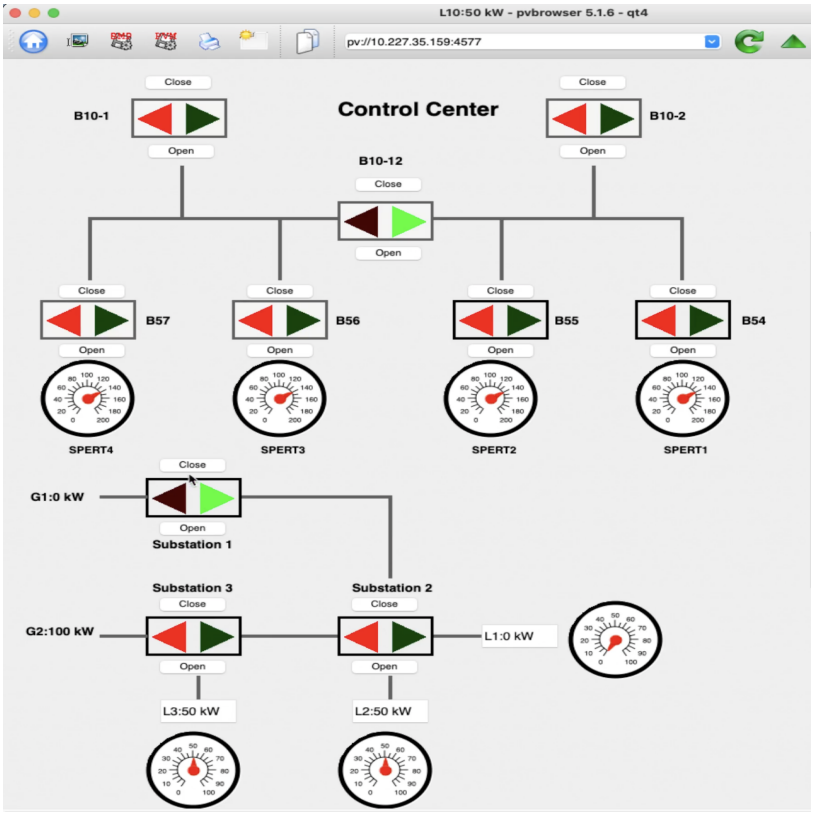
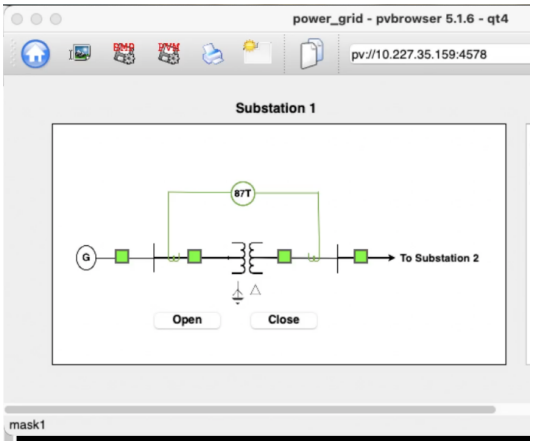
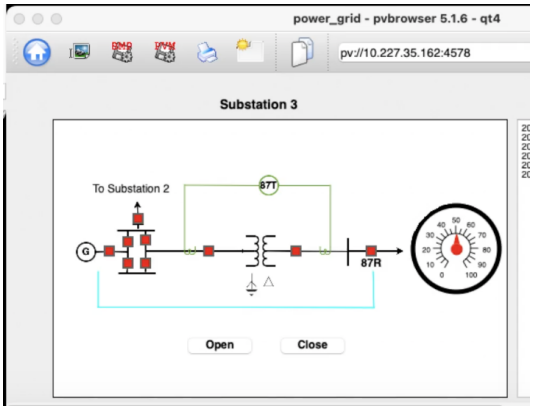


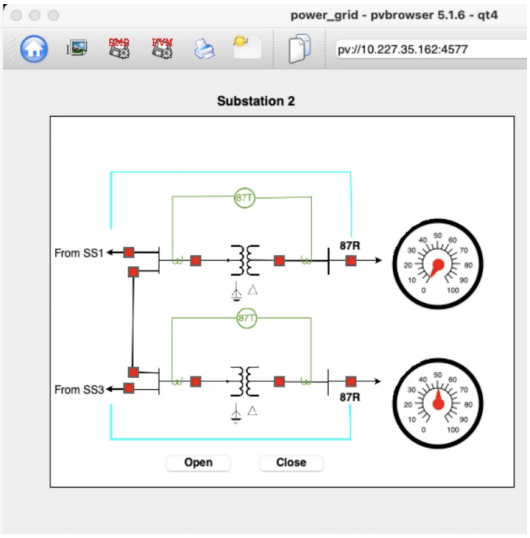
Figure 7.1: Control Center HMI for Integrated Scenario with DNP3 and Modbus PLCs at top and three IEC61850 substations at bottom



(a) Substation 1 HMI



(b) Substation 3 HMI



(c) Substation 2 HMI

Figure 7.2: Three Substation HMIs for Integrated Scenario

7.4 Integrated Scenario and Power Topology Emulator

To demonstrate the functioning of an end-to-end Byzantine-resilient system, we designed an integrated scenario that simulates realistic power grid behavior using a power emulator. This scenario involves a combination of Programmable Logic Controllers (PLCs), Remote Terminal Units (RTUs), and substations, operating in conjunction with a control center.

The scenario consists of seven PLCs and RTUs, communicating with the control center using the Modbus and DNP3 protocols (the top portion of Figure 7.1). These PLCs and RTUs are directly controlled and monitored by the control center. Additionally, the scenario includes three IEC61850 substations, that connect two gen-

erators to three loads (the bottom portion of Figure 7.1). Each substation is equipped with local substation HMIs for local control and monitoring (Figure 7.2).

In a real-world power grid, the physical state of system components—such as whether a breaker is open or closed—directly impacts the flow of electricity and the overall grid state. These state changes are captured by local sensors or merging units, which then broadcast the updated information across the system. For example, if the breaker in Substation 1 opens, Load L1 is shed, and the merging unit in Substation 1 detects this change, broadcasting it throughout the grid.

Our power emulator replicates these underlying dynamics by simulating the grid's response to state changes in key components like generators, transformers, and breakers. The emulator ensures that the merging unit is explicitly notified to generate the appropriate state change messages. For instance, when a breaker trips, the corresponding emulated merging unit is alerted and updates the substation's status. This, in turn, adjusts load distribution and simulates the real-time effects typically observed in actual power grids.

Chapter 8

Deployment Strategies

In this chapter, we outline a comprehensive deployment strategy designed to enhance the resilience of power grids against evolving cyber threats and system failures. The strategy emphasizes securing the network within the context of SCADA systems, followed by incremental deployment of a Byzantine resilient architecture and protocols that support Byzantine-resilient grid operations. This incremental approach enables operators to address grid vulnerabilities in a strategic way while reducing risk, economic burden and disruption to grid operations.

The proposed deployment strategy revolves around two core elements:

- **Securing the Network** with intrusion-tolerant network service to safeguard against both external and internal cyberattacks.
- **Incremental Deployment of the Complete Solution** to gradually transition vulnerable grid control systems systems into robust, Byzantine resilient grid control systems, by incrementally deploying the needed Byzantine resilient techniques.

8.1 Securing the Network

As discussed in the network model in Chapter 3, a local area network (LAN) connects the various SCADA components within a substation or control center. These LANs interconnect through a wide area network (WAN), which also links the control centers and data centers, enabling communication and data exchange across the entire SCADA system. This interconnectedness makes the grid particularly vulnerable to both external and internal network attacks. For instance, a malicious attacker that compromises a single node in the network and therefore can authenticate correctly, can cause significant disruption of SCADA communication or do even more severe, damage to the grid.

Cyber attacks targeting the power grid often exploit vulnerabilities within the communication networks that underpin SCADA systems. These attacks can take the form of Distributed Denial of Service (DDoS) attacks, malware infections, routing attacks and other network attacks, that compromise both wide area and local area SCADA network infrastructure. Malicious network attacks are one of the leading causes of security breaches, with adversaries gaining unauthorized access to the grid control systems by exploiting weak or unpatched network layers [83, 84].

As demonstrated in our solution, we can address these IP network resiliency concerns through use of an intrusion-tolerant network service, Spines. The Spines intrusion-tolerant network service operates by creating a secure and resilient layer that spans across multiple Internet Service Providers (ISPs), ensuring that even if one or more ISPs or network segments are compromised, the overlay can reroute traffic and maintain secure communication channels. The overlay is deployed on standard IP networks, leveraging existing infrastructure while mitigating the risk of attacks that target conventional Internet routing protocols.

By not relying solely on traditional IP routing, the system can dynamically navigate around compromised segments of the network. This approach can prevent or minimize the impact of attacks that exploit routing table manipulations, such as BGP (Border Gateway Protocol) hijacking or man-in-the-middle attacks or ISP meltdown. The intrusion-tolerant network service allows for better fault isolation, faster recovery from attacks, and a more secure network environment for the grid's operational needs.

Supporting Legacy Applications: One of the significant challenges when working with SCADA systems is the integration of legacy applications. Many power grid control systems rely on outdated software that cannot easily accommodate modern security protocols or architecture upgrades. Directly retrofitting or replacing these legacy systems could be complex, expensive, and risk introducing instability. Additionally, legacy SCADA applications may lack the flexibility to support new security measures without significant modification.

Fortunately, the Spines intrusion-tolerant network service is designed to integrate seamlessly with existing network infrastructures, even when they involve legacy applications. Instead of requiring a complete overhaul, the solution uses a virtual network device (e.g., a TUN or TAP interface) at the operating system level to route traffic through the secure, resilient network layer. This approach ensures that legacy applications continue using the standard IP networking stack, while their communication is intercepted and securely routed through the intrusion-tolerant network without requiring any changes to the underlying applications.

By leveraging the Spines intrusion-tolerant network service, operators can enhance the security of their legacy infrastructure gradually, reducing both the risk and the economic burden typically associated with large-scale migrations.

8.2 Incremental Deployment of the Complete Solution

Transitioning vulnerable, traditional power grid control systems to Byzantine resilient control systems requires a careful, incremental approach, particularly because many existing power grid control systems rely on legacy infrastructure that cannot be easily replaced or upgraded in a single step. A complete solution that integrates Byzantine resilience techniques into grid operations can be deployed gradually, reducing the risks and economic burden while ensuring that the control system becomes more resilient over time. The step by step transformation strategy can be treated as a system of systems, where each component or use case represents a distinct subsystem within a larger, integrated system. This modular approach allows for flexibility, enabling grid operators to prioritize high-risk or high-value use cases first, while ensuring that legacy systems can continue to function without being completely overhauled.

In context of Control Centers: By enabling Byzantine resilience in the control center, we ensure that the most critical part of the SCADA system is protected and resilient against system-level and network-level cyberattacks and intrusions. Introducing Byzantine fault tolerance (BFT) here ensures that decision-making processes remain intact, even in the event of cyberattacks, network failures, or system compromises. Recent research has even introduced reconfigurability [85], further enhancing system ability to withstand cyberattacks that occur in the aftermath of a natural disaster.

However, rather than implementing BFT as a rigid, one-size-fits-all solution, a more flexible approach can be employed. The transformation of BFT modules within the control center can be done as a service, offered through a cloud-based hybrid management model, detailed in [86]. This hybrid cloud-based service approach provides significant advantages, including scalability, easier integration, and the ability to continuously improve the resilience of the control center systems without heavy upfront infrastructure investments. By offering BFT as a service, utilities can subscribe to the necessary level of protection as needed, adapting to evolving threats without the burden of maintaining a large, complex internal BFT architecture and associated protocols.

In context of Substations: A grid will have substantial number of nodes - remote units, substations and other end points. Each node has varying level of criticality, due to the nature of assets or functionality. It can be impractical to enable Byzantine resilience in all nodes for all functionalities at once (from economic, efforts and practicality perspective). Hence, we can upgrade nodes in a strategic phased manner. This strategy can be further refined based on operational priorities, risk assessments, and evolving technological capabilities. For example, substations that serve high-population areas or are part of key transmission paths can be prioritized, with the Byzantine resilient solution tailored to each location's specific needs.

Expanding to More Functionalities and Use Cases: As the core high value components, control centers and critical substations are upgraded, the deployment can expand to other grid functionalities and use cases. This gradual expansion allows utilities to test and refine the implementation of Byzantine resilience in different contexts, ensuring that the entire grid operates cohesively. Over time, additional protocols and services can be integrated as needed, further enhancing Byzantine resilience.

Chapter 9

Conclusion

The power grid is critical for modern society and the economy, yet it is increasingly vulnerable to sophisticated cyberattacks, particularly from nation-state actors. These attacks have the potential to compromise and take control of grid components, such as SCADA Masters and protective relays, resulting in devastating disruptions and the loss of assets worth billions of dollars. Network attacks targeting grid communications further exacerbate the vulnerability, undermining the grid's resilience.

To address these threats, this thesis has developed comprehensive solutions designed to transform vulnerable power grid systems into Byzantine-resilient infrastructures capable of withstanding sophisticated cyber threats. Through rigorous evaluations, we demonstrated that our solutions met the stringent real-time latency requirements essential for grid operations, even under the most challenging operating conditions with successful intrusions and attacks.

A key contribution of this thesis is the novel and comprehensive threat model we developed, which encompasses broad grid infrastructure, including control centers, substations, RTUs, PLCs, and field devices. This model incorporates Byzantine faults at both the system and network levels, providing a foundation for addressing a wider range of power grid vulnerabilities than those previously considered.

We presented the first real-time Byzantine-resilient architecture and protocols for the substation, ensuring correct protective operations even in the presence of protective relay compromises and network attacks. We evaluated our implementation across a comprehensive range of fault-free and faulty operating conditions in relevant testbeds. The results demonstrated its ability to meet strict real-time latency requirements (4.167ms in a 60Hz system) even in the worst operating conditions under the threat model.

We also introduced the first end-to-end Byzantine-resilient system framework for the broad grid infrastructure spanning from the control center to the substation and field devices, under the comprehensive threat model. We demonstrated the proposed system framework's ability to support real-time grid operations in a Byzantine-resilient manner. To enhance situational awareness, we integrated unsupervised ma-

chine learning models for anomaly detection.

The solutions we proposed meet other critical domain requirements, including continuous availability over a long system lifetime and seamless integration with the grid. We implemented all modules and protocols and made them available to the community within the open-source Spire Toolkit, encouraging collaboration and further innovation. The system successfully withstood a rigorous purple team exercise and was transitioned to SCADA manufacturers GE and Siemens, along with two national laboratories PNNL and SANDIA. Finally, we proposed a practical, incremental deployment strategy for large-scale real-world power grid topologies.

Bibliography

[1] "Cost of a data breach 2022." [Online]. Available: https://www.ibm.com/reports/data-breach

[2] T. Group, "2022 thales data threat report." [Online]. Available: https://cpl.thalesgroup.com/critical-infrastructure-data-threat-report

[3] Mar 2021. [Online]. Available: https://www.gao.gov/assets/720/714089.pdf

[4] "Us electric grid growing more vulnerable to cyber-attacks, regulator says — reuters." [Online]. Available: https://www.reuters.com/technology/cybersecurity/us-electric-grid-growing-more-vulnerable-cyberattacks-regulator-says-2024-04-04/

[5] [Online]. Available: https://www.eia.gov/beta/states/data/dashboard/energy-infrastructure

[6] [Online]. Available: https://www.energy.gov/policy/articles/cyber-threat-and-vulnerability-analysis-us-electric-sector

[7] [Online]. Available: https://jhu-dsn.github.io/spire

[8] "Iec." [Online]. Available: https://webstore.iec.ch/publication/6007

[9] IEEE, "Ieee standard communication delivery time performance requirements for electric power substation automation," *IEEE Std 1646-2004*, pp. 1–24, 2005.

[10] K. Tweed, "Attack on nine substations could take down u.s. grid," Jun 2021. [Online]. Available: https://spectrum.ieee.org/attack-on-nine-substations-could-take-down-us-grid

[11] A. Avizienis, "The n-version approach to fault-tolerant software," *IEEE Transactions on software engineering*, no. 12, pp. 1491–1501, 1985.

[12] J. C. Knight and N. G. Leveson, "An experimental evaluation of the assumption of independence in multiversion programming," *IEEE Transactions on software engineering*, no. 1, pp. 96–109, 1986.

[13] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, "Os diversity for intrusion tolerance: Myth or reality?" in *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2011, pp. 383–394.

[14] F. B. Cohen, "Operating system protection through program evolution." *Comput. Secur.*, vol. 12, no. 6, pp. 565–584, 1993.

[15] S. Forrest, A. Somayaji, and D. H. Ackley, "Building diverse computer systems," in *Proceedings. The Sixth Workshop on Hot Topics in Operating Systems (Cat. No. 97TB100133)*. IEEE, 1997, pp. 67–72.

[16] V. Pappas, M. Polychronakis, and A. D. Keromytis, "Smashing the gadgets: Hindering return-oriented programming using in-place code randomization," in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 601–615.

[17] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[18] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the works of leslie lamport*, 2019, pp. 203–226.

[19] G. Bracha and S. Toueg, "Asynchronous consensus and broadcast protocols," *Journal of the ACM (JACM)*, vol. 32, no. 4, pp. 824–840, 1985.

[20] L. Lamport, "The part-time parliament," in *Concurrency: the Works of Leslie Lamport*, 2019, pp. 277–317.

[21] M. Castro and B. Liskov, "Proactive recovery in a {Byzantine-Fault-Tolerant} system," in *Fourth Symposium on Operating Systems Design and Implementation (OSDI 2000)*, 2000.

[22] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Verissimo, "Highly available intrusion-tolerant services with proactive-reactive recovery," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 4, pp. 452–465, 2009.

[23] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup, "Secure and efficient asynchronous broadcast protocols," in *Annual International Cryptology Conference*. Springer, 2001, pp. 524–541.

[24] A. Patra, A. Choudhury, and C. P. Rangan, "Asynchronous byzantine agreement with optimal resilience," *Distributed computing*, vol. 27, no. 2, pp. 111–146, 2014.

[25] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: speculative byzantine fault tolerance," in *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, 2007, pp. 45–58.

[26] Y. Amir, C. Danilov, D. Dolev, J. Kirsch, J. Lane, C. Nita-Rotaru, J. Olsen, and D. Zage, "Steward: Scaling byzantine fault-tolerant replication to wide area networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 1, pp. 80–93, 2008.

[27] Y. Amir, B. Coan, J. Kirsch, and J. Lane, "Customizable fault tolerance forwide-area replication," in *2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007)*. IEEE, 2007, pp. 65–82.

[28] Y. Mao, F. P. Junqueira, and K. Marzullo, "Mencius: building efficient replicated state machines for wans," 2008.

[29] A. Bessani, J. Sousa, and E. E. Alchieri, "State machine replication for the masses with bft-smart," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 355–362.

[30] Y. Amir, B. Coan, J. Kirsch, and J. Lane, "Prime: Byzantine replication under attack," *IEEE transactions on dependable and secure computing*, vol. 8, no. 4, pp. 564–577, 2010.

[31] G. S. Veronese, M. Correia, A. N. Bessani, L. C. Lung, and P. Verissimo, "Efficient byzantine fault-tolerance," *IEEE Transactions on Computers*, vol. 62, no. 1, pp. 16–30, 2011.

[32] R. Kapitza, J. Behl, C. Cachin, T. Distler, S. Kuhnle, S. V. Mohammadi, W. Schröder-Preikschat, and K. Stengel, "Cheapbft: Resource-efficient byzantine fault tolerance," in *Proceedings of the 7th ACM european conference on Computer Systems*, 2012, pp. 295–308.

[33] Y. Mao, F. P. Junqueira, and K. Marzullo, "Towards low latency state machine replication for uncivil wide-area networks," in *Workshop on Hot Topics in System Dependability*. Citeseer, 2009.

[34] G. S. Veronese, M. Correia, A. N. Bessani, and L. C. Lung, "Ebawa: Efficient byzantine agreement for wide-area networks," in *2010 IEEE 12th International Symposium on High Assurance Systems Engineering*. IEEE, 2010, pp. 10–19.

[35] R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 bft protocols," in *Proceedings of the 5th European conference on Computer systems*, 2010, pp. 363–376.

[36] J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin, "Separating agreement from execution for byzantine fault tolerant services," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003, pp. 253–267.

[37] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[38] A. Clement, M. Kapritsos, S. Lee, Y. Wang, L. Alvisi, M. Dahlin, and T. Riche, "Upright cluster services," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, 2009, pp. 277–290.

[39] M. Eischer and T. Distler, "Resilient cloud-based replication with low latency," in *Proceedings of the 21st International Middleware Conference*, 2020, pp. 14–28.

[40] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356.

[41] M. Baudet, A. Ching, A. Chursin, G. Danezis, F. Garillot, Z. Li, D. Malkhi, O. Naor, D. Perelman, and A. Sonnino, "State machine replication in the libra blockchain," *The Libra Assn., Tech. Rep*, vol. 7, 2019.

[42] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 31–42.

[43] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," Ph.D. dissertation, University of Guelph, 2016.

[44] M. Kogias and E. Bugnion, "Hovercraft: achieving scalability and fault-tolerance for microsecond-scale datacenter services," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–17.

[45] M. Poke and T. Hoefler, "Dare: High-performance state machine replication on rdma networks," in *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, 2015, pp. 107–118.

[46] A. Katsarakis, V. Gavrielatos, M. S. Katebzadeh, A. Joshi, A. Dragojevic, B. Grot, and V. Nagarajan, "Hermes: A fast, fault-tolerant and linearizable replication protocol," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 201–217.

[47] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "Rdma over commodity ethernet at scale," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 202–215.

[48] W. Zhao and F. E. Villaseca, "Byzantine fault tolerance for electric power grid monitoring and control," in *2008 International Conference on Embedded Software and Systems*. IEEE, 2008, pp. 129–135.

[49] J. Kirsch, S. Goose, Y. Amir, D. Wei, and P. Skare, "Survivable scada via intrusion-tolerant replication," *IEEE Transactions on Smart Grid*, vol. 1, no. 5, pp. 60–70, 2014.

[50] N. A. C. Medeiros, "A fault-and intrusion-tolerant architecture for edp distribuicao scada system," Ph.D. dissertation, 2011.

[51] A. Babay, T. Tantillo, T. Aron, M. Platania, and Y. Amir, "Network-attack-resilient intrusion-tolerant scada for the power grid," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 255–266.

[52] A. Nogueira, M. Garcia, A. Bessani, and N. Neves, "On the challenges of building a bft scada," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 163–170.

[53] W. Beaton, "Eclipse neoscada," May 2024. [Online]. Available: https://projects.eclipse.org/projects/iot.eclipsescada

[54] [Online]. Available: https://www.bgpmon.net/chinese-isp-hijacked-10-of-the-internet/

[55] A. Studer and A. Perrig, "The coremelt attack," in *European Symposium on Research in Computer Security*. Springer, 2009, pp. 37–52.

[56] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *2013 IEEE symposium on security and privacy*. IEEE, 2013, pp. 127–141.

[57] A. N. Bessani, P. Sousa, M. Correia, N. F. Neves, and P. Verissimo, "The crutial way of critical infrastructure protection," *IEEE Security & Privacy*, vol. 6, no. 6, pp. 44–51, 2008.

[58] M. Garcia, N. Neves, and A. Bessani, "Sieveq: A layered bft protection system for critical services," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 511–525, 2016.

[59] S. Mavrovouniotis and M. Ganley, "Hardware security modules," in *Secure Smart Embedded Devices, Platforms and Applications*. Springer, 2013, pp. 383–405.

[60] N. Aaraj, A. Raghunathan, and N. K. Jha, "Analysis and design of a hardware/software trusted platform module for embedded systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 8, no. 1, pp. 1–31, 2009.

[61] M. Jafari, A. Kavousi-Fard, T. Chen, and M. Karimi, "A review on digital twin technology in smart grid, transportation system and smart city: Challenges and future," *IEEE Access*, vol. 11, pp. 17 471–17 484, 2023.

[62] "Communication networks and systems for power utility automation – part 5: Communication requirements for functions and devices," Geneva, Switzerland, 2013, this part of IEC 61850 specifies the communication requirements for functions and devices in power utility automation systems.

[63] "Ieee standard for synchrophasor measurements for power systems," *IEEE Std C37.118.1-2011 (Revision of IEEE Std C37.118-2005)*, pp. 1–61, 2011.

[64] "Ieee 1588-2019 - ieee standard for a precision clock synchronization protocol for networked measurement and control systems." [Online]. Available: https://standards.ieee.org/content/ieee-standards/en/standard/1588-2019.html

[65] D. Obenshain, T. Tantillo, A. Babay, J. Schultz, A. Newell, M. E. Hoque, Y. Amir, and C. Nita-Rotaru, "Practical intrusion-tolerant networks," in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2016, pp. 45–56.

[66] A. Babay, C. Danilov, J. Lane, M. Miskin-Amir, D. Obenshain, J. Schultz, J. Stanton, T. Tantillo, and Y. Amir, "Structured overlay networks for a new generation of internet services," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1771–1779.

[67] [Online]. Available: https://Spines-org.github.io/

[68] "nerc.com," [Accessed 05-12-2024]. [Online]. Available: https://www.nerc.com/pa/Stand/Frequency%20Response%20Project%20200712%20Related%20Files%20DL/BAL-003-1_Procedure-Clean_20120210.pdf

[69] E. Ela, V. Gevorgian, A. Tuohy, B. Kirby, M. R. Milligan, and M. J. O'Malley, "Market designs for the primary frequency response ancillary service—part i: Motivation and design," *IEEE Transactions on Power Systems*, vol. 29, pp. 421–431, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:12862004

[70] [Online]. Available: https://www.nerc.com/pa/Stand/Reliability%20Standards/PRC-024-2.pdf

[71] S. Bommareddy, D. Qian, C. Bonebrake, P. Skare, and Y. Amir, "Real-time byzantine resilience for power grid substations," in *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2022, pp. 213–224.

[72] S. Bommareddy, M. Khan, D. S. Cardenas, C. Miller, C. Bonebrake, Y. Amir, and A. Babay, "Real-time byzantine resilient power grid infrastructure: Evaluation and trade-offs," in *43rd IEEE Real-Time Systems Symposium*, 2022.

[73] A. Babay, J. Schultz, T. Tantillo, S. Beckley, E. Jordan, K. Ruddell, K. Jordan, and Y. Amir, "Deploying intrusion-tolerant scada for the power grid," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2019, pp. 328–335.

[74] T. Roeder and F. B. Schneider, "Proactive obfuscation," *ACM Transactions on Computer Systems (TOCS)*, vol. 28, no. 2, pp. 1–54, 2010.

[75] B. Fitzgerald, "The transformation of open source software," *MIS quarterly*, pp. 587–598, 2006.

[76] "S.4913 - 117th congress (2021-2022): Securing open source software act of 2022," accessed: 2022-9-30. [Online]. Available: https://www.govinfo.gov/content/pkg/BILLS-117s4913is/pdf/BILLS-117s4913is.pdf

[77] D. Bhamare, M. Zolanvari, A. Erbad, R. Jain, K. Khan, and N. Meskin, "Cybersecurity for industrial control systems: A survey," *computers & security*, vol. 89, p. 101677, 2020.

[78] M. A. Umer, K. N. Junejo, M. T. Jilani, and A. P. Mathur, "Machine learning for intrusion detection in industrial control systems: Applications, challenges, and recommendations," *International Journal of Critical Infrastructure Protection*, vol. 38, p. 100516, 2022.

[79] C. Bonebrake, D. J. Sebastian-Cardenas, C. H. Miller, S. Bommareddy, Y. Amir, K. Cornelison, C. Eyre, P. Skare, S. N. G. Gourisetti, A. Ashok, and B. Johnson, "Byzsec — a multi-layered byzantine resilient architecture for bulk power system protective relays," in *2024 IEEE Power and Energy Society General Meeting PESGM*, 2024, pp. 1–5.

[80] [Online]. Available: https://pvbrowser.de/pvbrowser/index.php

[81] [Online]. Available: https://dnp3.github.io/

[82] M. Zillgith and M. Zillgith, "libiec61850," Sep 2024. [Online]. Available: https://libiec61850.com/

[83] K. Kimani, V. Oduol, and K. Langat, "Cyber security challenges for iot-based smart grid networks," *International journal of critical infrastructure protection*, vol. 25, pp. 36–49, 2019.

[84] "Chinese government poses 'broad and unrelenting' threat to u.s. critical infrastructure, fbi director says." [Online]. Available: https://www.fbi.gov/news

[85] S. Bommareddy, M. Khan, H. Nadeem, B. Gilby, I. Chiu, J. W. van de Lindt, O. Nofal, M. Panteli, L. Wells, Y. Amir *et al.*, "Tolerating compound threats in critical infrastructure control systems," in *2024 43rd International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2024, pp. 66–79.

[86] M. Khan and A. Babay, "Making intrusion tolerance accessible: A cloud-based hybrid management approach to deploying resilient systems," in *42nd International Symposium on Reliable Distributed Systems (SRDS)*, Marrakesh, Morocco, September 2023, pp. 254–267.

Vita

Sahiti Bommareddy grew up in Hyderabad, India. She earned a Bachelor of Technology degree in Electronics and Communication Engineering (ECE) from Jawaharlal Nehru Technological University in 2011, where she was also honored with the Spirit of ECE Award.

Before pursuing her research career, Sahiti gained industry experience as a Performance Engineer at Deloitte and Aktrix. She then went on to receive a Master of Science in Computer Science from Duke University in 2019.

During her PhD program, Sahiti was a member of the Distributed Systems and Networks (DSN) lab at The Johns Hopkins University. Her research interests include Byzantine-resilient systems and networks, resilient critical infrastructure, and the application of machine learning for cybersecurity.