Securing Clock-Synchronization in TTEthernet-Based Industrial IoT

Mogos Anday Gebremedhin

A thesis submitted in partial fulfilment of the requirements of the University of East London for the degree of professional doctorate in Information Security

School of Architecture Computing and Engineering (ACE)

University of East London

August 26, 2024

Abstract

TTEthernet is an extension of the Ethernet protocol and offers the benefits of communicating different traffic types on the same network backbone. TTEthernet clock synchronization has advantages over alternative options as it provides a strict scheduling algorithm where time-triggered traffic gets priority over event-triggered ones. Thus, the security of traffic communicating in the TTEthernet-based IIoT starting with the clock synchronization as the basis for all traffic communication is the focus of this research. As one of the leading security threats to the TTEthernet clock synchronization, the latency of synchronization frames is investigated and a fitting security solution is proposed in the form of an anomaly detection model. A simulation tool, Visualsim, is used to design three network models, run simulations to study the traffic communicated between end systems and validate the proposed security solution for the different network topologies and complexities. An anomaly detection model is designed using Python to define specific rules that detect anomalous latency of synchronization frames and flag them into a separate list before they are printed into a CSV file for the attention of network practitioners to respond with appropriate resolution. The anomaly detection model uses data from the simulation as input to execute the rules defined to nullify anomalous synchronization frames for latency. Global and local latency thresholds are designed to flag synchronization frames as anomalies and add them to the list which is printed for anomalous latency if they defy the maximum latency thresholds defined in the anomaly detection model. This research offers a novel solution that researchers and network practitioners can use to fill a knowledge gap and improve the security of TTEthernet clock synchronization in the IIoT environment.

Acknowledgment

I would like to start paying my due gratification to my supervisors Dr Amin Karami and Dr Ameer Al Nemrat. Dr. Amin, you are a great human being as well as a great supervisor to have. You have always been there when I needed you and always told me what I needed to hear. I would not be able to complete this work if it were not for your continued advice and guidance. I should also thank you, Dr. Ameer, for getting me started with this great challenge and handing me over to the great man that is Dr. Amin.

I would also like to thank Dr Shareeful Islam, Dr. Fadi Safieddine, Dr. Umaima Haider, and Dr. Nabeela Berardinelli for their contribution to the two annual progress monitoring reviews I had. All the suggestions you made and the concerns you pointed out made me a better researcher as I could see things from different perspectives. So, a big thank you goes to all of you.

I wish also to extend my gratitude to the great instructors including Professor Rabih Bashroush, Professor Allan J Brimicombe, Dr. Yang Li, and Dr Ameer Al Nemrat who were involved in the taught part of the program and helped me understand the subject matter better and the research techniques which I have employed inadvertently for this research work.

My gratitude also goes to Dr Pablo Gutierrez Peon (TTTech Auto, Vienna, Austria), Dr Daniel Onwuchekwa (University of Siegen, Germany), Dr Michele Nati, Dr Ivan Petrunin (Cranfield University, UK), and Dr Nahman Tariq Ratyal (Cranfield University) who were generous for their ideas and advice which has shaped the way I saw the subject matter as well as the perspective I needed to consider. So, thank you all for your invaluable ideas and suggestions.

A special gratitude goes to the Development team at Mirabilis Design for the simulation tool and more so to Tom Jose who was always happy to help with the Visualsim

configuration challenges. I am sure it would not be that easy without your help. So, thank you for your time and effort and forward my gratitude to your team for making Visualsim such a great tool to work with as a simulation and modelling tool.

On a personal level, I cannot thank my wife, Rahwa O Hadgu, enough. You had to take the burden of covering most things that I would have done otherwise for me to work on my research work. My children, Ruthy, Saron, Gideon and Ariam deserve a big thank you as you always understood me when I said I was working on my studies where I would be expected to do my fatherly duties. This is really for my family. Love you all!

My final thanks go to all my friends, colleagues and family but more so to Dawit S Tesfay and Kflzgi Bayru who have significantly supported me to keep persevering when I felt utterly worn out.

Dedication

To my Family!

My lovely wife Rahwa O Hadgu for always standing by me

&

My beautiful Children Ruthy, Saron, Gideon and Ariam. Remember dreams come true if you persevere!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The term IIoT (Industrial Internet of Things) encompasses the fusion of data gathering, communication, and analysis through a network of interconnected smart devices, (Zhao et al., 2019a). These devices include sensors, actuators, controllers, and other computer-network components, all working together to deliver comprehensive solutions within the Industrial realm. Intelligent device refers to the ability of a physical device to collect, process and communicate digital information within a network. Real-time temperature sensors in a food factory, for example, send instant temperature levels to the control systems (or they can be remotely monitored using mobile phones or tablets) to ensure food safety during the transit, processing and storage stages, unlike the traditional, manual measuring tools. Liquid-level sensors are another implementation of the IIoT. These specialized sensors are designed to change state or signal an alert to the control systems when the liquid level crosses the set minimum or maximum thresholds, or when the fuel enters or exits a fuel tank. This allows the activation of suitable measures to avert any hazardous consequences, (Bose et al., 2020).

One major development within the IIoT paradigm is the use of time-triggered Ethernet (TTEthernet) for synchronous data communication in real-time applications. TTEthernet is a communication protocol developed to support real-time traffic communication for mixed-criticality applications within the same network setup, (Suethanuwong, 2012). One of the main advantages of TTEthernet is its backward compatibility with the IEEE 802.3 Ethernet protocol, (Tămaş-Selicean and Pop, 2014). Thus, TTEthernet is used on an Ethernet network setup. Referring to the examples given above, TTEthernet is used for

the hard real-time communication of the status of potentially dangerous temperature or liquid levels, detected by the respective sensors, on a pre-defined static schedule, and the routine Ethernet traffic including the day-to-day file transfers on the same network backbone.

Furthermore, it helps to communicate Time-Triggered (TT) traffic on prescheduled regular time slots for the hard real-time traffic types while it still supports the less time-sensitive traffic types namely, the Rate-Constrained (RC) which involves the certain quality of service (QoS) mainly a predefined bandwidth allocation and limited temporal deviation and Best-Effort (BE) traffic for the general non-time-sensitive Ethernet traffic types which do not consider latency as QoS, (Kyriakakis et al., 2020). Clock synchronization frames also called Protocol Control Frames (PCFs) form the basis of an integrated network traffic communication in a TTEthernet network and they have the highest priority among other traffic types, (Steiner, 2013). Hence, secure PCF communication among member nodes is critical to form the foundation for secure overall network communication. Clock synchronization refers to the mechanism of synchronising the clocks of every member device in a network, (Wang, 2018). Most messages in real-time communication depend on deadlines (Lisova et al., 2016a). Hence, this could be exploited by an adversary as a breach in the clock synchronization disrupts the network functionality.

This research explores what industrial networks stand for. The traditional operational technology and the emergence of IT supplemented by the modern-day IoT infrastructure make industrial IoT more complex but efficient than ever before. Nonetheless, the security threat has become equally more complex and difficult to address. Thus, this research starts by laying down the basic background of Industrial IIoT and the security threats associated with them, to get a good grip on the subject matter.

1.1 Background of Industrial Networks and their Security Implications

Traditional Industrial networks are mainly based on Operational Technology (OT). OT refers to hardware and software used to monitor, and control events, devices and processes often related to the management of critical infrastructure in industrial plants, (Paes et al., 2019). It is usually associated with industrial control systems such as Supervisory Control and Data Acquisition (SCADA), Distributed Controller Systems (DCS) and Programmable Logic Controllers (PLC) among others, (Garimella, 2018). Hence, OT-based industrial networks mainly focus on the machinery and the operation of physical processes along the production line; physically separated from the external world, (Conklin, 2016). Consequently, they are mostly immune to external attacks from a security point of view. however, they are not bullet-proof to internal security breaches as malicious or unintentional users breach the set security policy including unauthorized access to certain areas or specialized tools risking the safety of human operators and the operating machines, (Evans et al., 2019) & (Hals, 2015). With the emergence of modern technology, however, organizations in the industrial sector started to integrate OT and IT and manage them on the same control system, (Conklin, 2016)& (Garimella, 2018). The need for more data, an easier way of collecting them using cheap sensor devices and the use of automation technologies make it attractive to integrate many intelligent devices, whether they are old legacy devices or advanced computers capable of storing, processing and communicating with other network devices, (Paes et al., 2019). Unlike the traditionally secluded industrial networks, the modern IIoT where OT and IT are integrated, Fig 1.1 lives with the major cyber security breach concern; breaches that can be had from anywhere in the world through the power of the internet.

The modern-day Industrial Internet of Things (IIoT) benefits from the internet and interconnection of most devices which are made with connection capabilities. This has enabled organizations to expand exponentially by reaching out to a wider customer base within a click of a button. Thanks to the advancement in modern technology, it is predicted that IIoT will add around $14.2Tr to the global economy by 2030, (Centerholt

et al., 2020). However, this has exposed modern industrial organizations to cyber security threats, (Paes et al., 2019) & (Tawalbeh et al., 2020). According to the same authors, vulnerabilities within the IIoT are increasing by the day as the attack surface increases due to the increasing number of connected devices. Furthermore, many organizations in the industrial sector set their priority on the fact that systems should keep functional; so, the production process does not get interrupted. Hence, it's not unusual to find an old computer, for example, running Windows XP which runs an embedded software that controls a critical production function, because an upgrade to the operating system can slow or even stop the production line, (Stouffer et al., 2011). It is also common knowledge that Microsoft stopped support for Windows XP in April 2014. Thus, a creative approach needs to be considered by applying an appropriate security solution, including but not limited to strict t firewall rules and an access control list, which keeps the production line running but keeps the legacy device as secure as it can be.

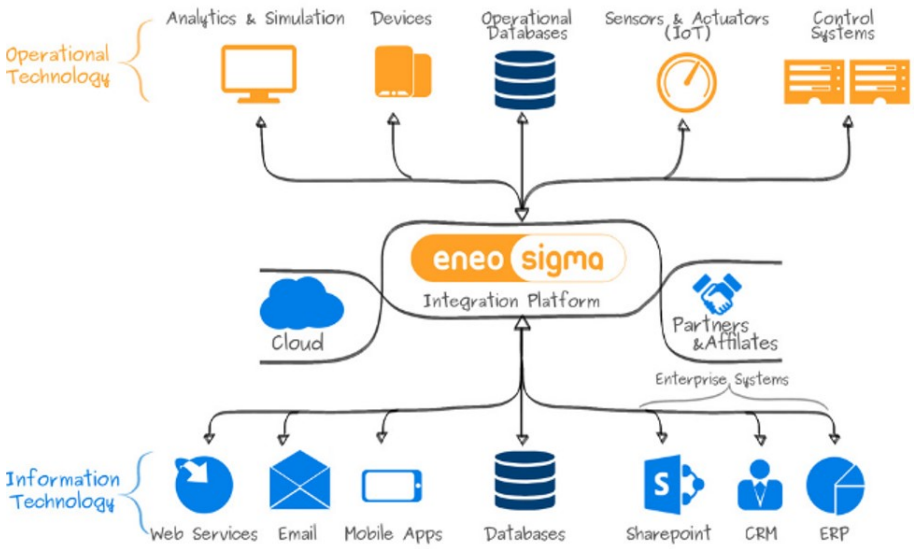

Figure 1.1: IT/OT integration in an IIoT,(EneoSigma, 2023)

Therefore, the industrial sector is at a crucial point where the balance between the adoption of IIoT on the one hand and the importance of still using legacy systems to do some important controlling or processing tasks, on the other hand, becomes important from the security point of view. The first is important for better connectivity between manufacturers and their customers, easier data collection, storage and transmission, device-to-device communication and remote monitoring using mobiles or tablets as

control systems, among others; while the latter remains integral to most modern deployments as they are too expensive to replace at once or halt the production line to patch up or upgrade firmware, if at all possible. It is given that many organizations still incorporate legacy systems with proprietary protocols which are less compatible with modern-day technological developments including the security infrastructure. Thus, more research must be carried out to balance the importance of using legacy systems with the right security measures in place and the adoption of the IIoT for efficiency and ease of production. Nonetheless, it is important to explore what industrial IoT infrastructures look like before security measures can be devised.

1.2 Industrial Internet of Things (IIoT)

Industrial networks have evolved fast ever since the emergence of the internet. IIoT is an enabling technology for Cyber-Physical Systems (CPS) and allows these physical objects to sense, collect and communicate digital data over a network infrastructure, (Koulamas and Kalogeras, 2018). CPS is the integration of physical systems and embedded computing to control and/or monitor certain functionalities and influence actual outcomes through physical processes. (Boyes et al., 2018). Thus, the rise of IIoT is helping Industrial organizations to potentially grow financially and become more efficient with their production, control of devices and monitoring of the production processes. For example, Predictive maintenance (Poór et al., 2019), is a familiar IIoT deployment used to check the status or condition of assets in real-time using status monitoring tools to avoid costly reactive maintenance. Thanks to the interconnection of devices and sharing of information, this technology helps to fix or replace worn equipment before the operation or the equipment itself fails to function. Logistics companies have similarly embraced IIoT so they can check the location and condition of their fleet as well as prevent goods from damage due to changes in temperature or humidity levels among others, (Wu et al., 2019). Autonomous vehicles (Kirk, 2015) are another well-known example which usually accompanies the term IIoT. A car would start calculating routes after passengers put their destination address. It is made with interconnected intelligent sensors such as GPS, pollution detectors, accelerometer and video cameras (Bhoi et al., 2019); so, information

collected from these devices, using the intelligent sensors, could be used in the deep learning algorithms to identify objects within a fixed distance proximity relative to their current 3D map, dynamically, and predicts what these objects, if any, might do next, within a fraction of a second.

Therefore, IIoT is a technology that is transforming the modern-day industry to another level, in so many different aspects. However, it's bound to change the safety and security aspects as the threat landscape changes with the complexity of the technology infrastructures used. Consequently, monitoring and protection technologies need to change to cope with the implementation of IIoT.

1.2.1 Safety and Security Implications in the CPSs

IIoT is the interconnection and communication of intelligent CPSs using the internet. It's, therefore, important to understand the types of CPSs to understand the varieties of Industrial networks and application areas they are used for Industrial control systems (ICS) to monitor industrial processes; smart grids to help ease communication between suppliers and customers of electricity supply companies; and robotics systems for automation tasks are some examples of CPSs and their respective application areas.

In traditional industrial networks, OT systems are primarily concerned with the safety of human/device operators alongside the operational functionalities, disregarding the risks associated with network connectivity, (Conklin, 2016). However, IT focuses on the secure delivery of digital data. Hence, with the emergence of IoT, the convergence of IT and OT systems doubled the safety and security concerns as two interdependent requirements within industrial networks, (Lisova, 2018). Although the IT/OT convergence offers businesses with an exponential opportunity to grow and maximize revenue, the interconnection of these CPSs through the internet has increased the existing entry points for the bad actors to attack industrial networks and exploit the vulnerabilities within the OT systems which have long been closed from the external world, (Paes et al., 2019) & (Tawalbeh et al., 2020). Furthermore, some CPS devices, including the medical testing and monitoring tools, are mostly vendor-specific. Besides, they usually don't use standard operating systems and still support insecure protocols.

Sometimes, they are unmanaged with default credentials; hence, offer easy access for malicious attackers. Some of the well-publicized security breaches in the IIoT include the Stuxnet (Falliere et al., 2011), a malware attack that targeted the PLCs which are used to automate machine processes in an Iranian nuclear plant and the Ukrainian power grid attack, (Liang et al., 2016), where hackers compromised three energy companies' information systems and temporarily disrupted the electricity supply for part of the capital city. The 2016 DDoS attack on IoT devices at a record-breaking rate of 1.2 Tbps, in the name of MIRAI, was another devastating security breach on IIoT, (Kolias et al., 2017).

A detection module has been proposed in (Kawamura et al., 2017) for the DDoS attacks on IoT. Therefore, organizations are required to invest to secure their data, network infrastructure as well as their CPSs or end systems. This has in turn created an opportunity for researchers to dig into the ever-growing collected data and learn new patterns and gain new knowledge.

Clock synchronization is the foundation of most network infrastructures; hence, a potential target to be exploited to disrupt network communication. Clocks of devices within the same network need to synchronize so scheduled traffic gets delivered to the intended recipient at the expected time, (Wang et al., 2018). If a device's clock goes out of sync, however, it misses out on important messages as it would not be able to communicate with other network devices in a timely manner; thus, it creates an undesired traffic collision in the process, (Lisova et al., 2016a). Considering this in an industrial setting where automated CPSs communicate in a time-trigged fashion where traffic is communicated on fixed, prescheduled time slots, a delayed delivery of data from a fuel level sensor, for example, can cause hazardous reaction injuring or even worse taking the life of human operators.

Therefore, as industries start to modernize and enhance their production capabilities using the latest technologies, an equivalent investment in the security sector becomes a requirement as they can perish at a faster pace than they have developed. Hence, this project focuses on protecting the clock synchronization protocol to help secure the TTEthernet-based IIoT network communication; although, it goes without saying that not all security breaches in the industrial sector target the clock synchronization protocol.

1.2.2 TTEthernet for IIoT and the Security Implications

Ethernet has been commonly used as the medium of communication for Local Area Network (LAN) applications including industrial control applications, (Peón et al., 2015). With the emergence of wireless sensor devices and IoT, however, organizations have started to demand easier, safer, and more cost-effective solutions. Hence, a technological solution with better capabilities than the existing standard communication technology has become a requirement. Therefore, TT Ethernet (TTEthernet), among other alternatives, has emerged to address certain mixed-criticality data communication requirements where real-time data transmission is scheduled to access the channel of transmission with relative priority. TTEthernet, (Steiner et al., 2009), standardized by SAE International as SAE S6802, (Ethernet, 2016) is a communication platform on the Ethernet network which generates deterministic communication for mixed-criticality systems to allow synchronous & asynchronous communications, (Abuteir and Obermaisser, 2013) which is quickly adopted by aerospace, vehicle, and industrial applications, among others, (Wang et al., 2018). It is compatible with the IEEE 802.3 Ethernet standard, (Dutertre et al., 2012). Hence, it has become an ideal solution for sectors which depend on real-time data communication, of which Industrial Automation is one. In a production line, in the manufacturing industry, for example, a series of workstations are connected by a transfer system so they can process and move a product or part of it to the next workstation in line to add some more processing of its own and move on to the next workstation until the product is finalized in the last workstation. This benefits from the use of TTEthernet as real-time data communication between workstations and the Control Centre are required so the right processing step can be applied at the right time as a delay by a fraction of a second could potentially pose safety hazards. Thus, industrial applications are the focus of this paper. Traffic in a TTEthernet-based network is classified as TT traffic, RC traffic and BE traffic, (Tămaş-Selicean et al., 2012) & (Abuteir and Obermaisser, 2015). TT traffic packets do have precedence followed by the RC and BE traffic packets, respectively, in a TTEthernet-based network. Hence, TTEthernet is more important for organizations focused on TT traffic communication than the event-triggered RC and BE traffic communications.

With the advancement of new technologies and the demand for more cost-effective and secure services, it has become apparent that organizations should find ways of replicating the added functionality guarantee that a wired TTEthernet offers on the wireless segment of an IIoT (Peón et al., 2014). Wireless sensor devices should be able to report a certain temperature or humidity level drop or increase on the set level in a real-time manner, for example, so, that a technician at the control centre can react in a timely manner.

Thus, a TTEthernet setup at the IoT level, where wireless sensors are important components, brings much more success to industries than the traditional wired Ethernet network system. Nonetheless, it cannot be stressed enough that the networking and communication among these wired and wireless devices within the TTEthernet-based IIoT is secure. Furthermore, for safe and secure communication, it is important that these devices are properly synchronized. Hence, all member nodes have the same notion of time which is critical in real-time applications.

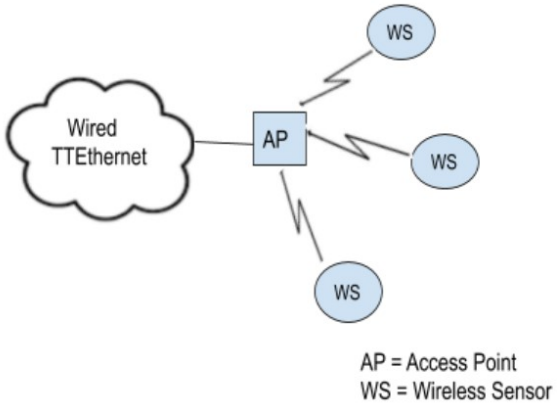

Figure 1.2: A sample setup of TTEthernet-based IIoT

1.2.3 Clock Synchronization in TTEthernet-based IIoT

The clock-synchronization protocol is an important component of the TTEthernet setup. It is used to coordinate or synchronize otherwise independent clocks on member devices within a network. A synchronized time notion is important for networked devices to perform fundamental network applications like data fusion, transmission scheduling, target tracking and energy management, (Wang et al., 2018) & (Mazur et al., 2016). Naturally, clocks tend to drift after some time due to temperature fluctuation or clocks

counting at slightly different rates. The reason clocks drift in a network could be that clocks have differences in frequency, different starting points, or different actual tick intervals between them, (Zhang et al., 2016). Hence, communicating endpoints within the same network are required to sync regularly to maintain seamless timing among them all. Most importantly, in a TTEthernet-based IIoT where real-time data communication is required, synchronization PCFs need to be communicated regularly, for safety and security reasons. Clock synchronization frames, also called PCFs, are communicated between all member nodes regularly so that the follow-up data transfer is delivered promptly. According to, (Zhang et al., 2016) the PCF communication in a TTEthernet clock synchronization, Fig 1.3, starts with a few Synchronization Masters (SMs) collecting their baseline global clock from an external source like a GPS receiver or they can use their internal clock as the time base that every member node needs to adjust itself to. In this research, it's assumed the time reference which feeds the SMs to be perfect, whether it is external or internal to the SMs, to focus on the clock synchronization within a TTEthernet-based network. The SMs then send PCFs to the Compression Masters (CMs). The CM, usually the network switch, then aggregates the PCFs received from all SMs and uses its algorithm to calculate the average clock while it also considers the time spent during the PCF communication from the SMs before it compresses and broadcasts the final sync message to all the end systems within the network: the SMs and the Synchronization Clients (SCs). Just like the SMs, there are multiple CMs in a TTEthernet-based clock synchronization for fault tolerance purposes. Thus, if one channel fails due to a failed switch (hence a CM), then another switch is activated keeping the traffic transaction unaffected.

Different synchronization protocols, including the GPS, NTP, IRIG-B, and PTP, have been suggested in various researches, (Yang et al., 2016b). These time correction protocols have been compared for the accuracy, lock time, cost, Ethernet support and reliability properties, as shown in Table 1.1. Precision Time Protocol (PTP) of the IEEE 1588 standard has prevailed regarding the cost, support for Ethernet and reliability. However, considering the requirements for tighter clock synchronization, the development of TTEthernet synchronization services becomes necessary for certain

Property	GPS	BD system	NTP	IRIG-B	IEEE1588
Accuracy	20ns	100ns	10ms	10us	100ns
Lock Time	40s	60s	30ns	60ns	60ns
Cost	High	High	Low	Low	Low
Ethernet	No	No	Support	Support	Support
Reliability	Medium	Medium	High	High	High

Table 1.1: Clock synchronization protocols comparison,((Yang et al., 2016a))

application scenarios, (Daniel and Roman, 2018). TTEthernet supports multiple SMs and CMs making it more fault tolerant as compared to the single point of failure where a single Grand Master (GM) or Best Master Clock (BMC) all member nodes are synchronized to PTP. PTP supports an automatic selection of a new GM clock from all member clocks to replace a malfunctioning one, but this means it's not as tight compared to the hard real-time synchronization service TTEthernet offers. Therefore, although the security side requires further research as is the case with all other synchronization protocols, it is evident that certain applications requiring real-time traffic communication, mainly in the industrial and aviation sectors, benefit more from the services of TTEthernet clock synchronization as compared to the protocols analysed in Table 1.1

1.2.4 Security Threats in the Clock Synchronization for TTEthernet-Based IIoT

Industrial networks mostly depend on real-time data transmission. Thus, it's imperative that safe and secure communication within an IIoT, for which the TTEthernet protocol plays an important role by creating a congestion-free medium for traffic transmission, is achieved. This is done by implementing prescheduled time slots so that the medium of communication is always available for the right type of traffic. An unsynchronized node in a network does not communicate properly as it can't adhere to the scheduling algorithm setup by TTEthernet for traffic communication as time is the main frame of reference between network devices. This causes certain security issues including the reliability and availability of resources, (Lisova et al., 2016a). Thus, securing the availability and reliability alongside other important security objectives needs to be protected against cyber-related security breaches.

Securing an IIoT network starts with the security of the clock synchronization

protocol. However, PCFs in a TTEthernet-based clock synchronization are broadcasted by the CMs to all network devices using the UDP protocol, which is usually susceptible to security threats, mainly distributed denial-of-service (DDoS) type attacks, among others. (Herrera et al., 2018) compared TCP SYN flooding and UDP flooding concerning susceptibility to the DDoS attack and concluded that UDP flooding is more attractive to DDoS when it comes to preventing access to services by overloading target systems.

(Kyriakakis et al., 2020) states that the correct execution of real-time applications is dependent on functional as well as temporal correctness. In their illustration, these authors asserted that a collision avoidance system, for example, needs to detect the presence of an object in their close vicinity, but the temporal transmission and processing of the images involved becomes important for correct decision-making.

Clock synchronization security breaches usually involve a breach on the communication channel or the network devices themselves. However, other attack types are usually involved in performing clock synchronization breaches. For example, a man-in-the-middle (MITM) type of attack would be required, to perform the node manipulation attack. Furthermore, most of the security attacks targeting the clock synchronization protocol aim to disrupt, mainly, the availability and reliability of security objectives which are the most important in the industrial sector. Table 1.2, summarizes the main attack types that target the communication channel or end systems. Therefore, Considering the types of attack discussed in this section and the vulnerabilities within the TTEthernet clock synchronization protocol, a security solution must be sought in this research to secure safety-critical applications in the IIoT infrastructure

1.2.5 Outlining a Security Solution for TTEthernet Clock Synchronization in an IIoT

PCFs have the highest priority in the TTEthernet-based clock synchronization, (Daniel and Roman, 2018). This helps to maintain the integrity of TTEthernet traffic transactions. Contention-free communication in TTEthernet is maintained by using three different integration techniques; namely: preemptive, timely-block and shuffling, (Zhao et al., 2018). Preemptive refers to the integration technique where an RC frame transaction is

Attack types	Description of attack
Node insertion	A malicious node is inserted with the intention of disrupting the clock synchronization protocol
Node compromising	An existing node is compromised; hence, besides the internal functionality of the node, its role in the clock synchronization: sending and forwarding of sync frames is compromised.
Message deletion	the ability to attack the communication link in the form of an MITM attack to intercept and drop PCF messages.
Message insertion	This is an attack on the communication channel and involves inserting a malicious frame among the benign traffic to disrupt the clock synchronization.
Message manipulation	refers to the ability of a bad actor to replay PCFs by delaying, re-sending or forging them for malicious reasons.

Table 1.2: Clock synchronization specific attack types

interrupted to give way for a prescheduled TT frame. The RC frame is then re-transmitted after the TT frame is communicated in full. In a Timely-block, however, an RC frame is started and transmitted only if there is enough time to transmit the whole frame before the next scheduled TT frame starts. Otherwise, it would have to wait until the next opportunity where an RC frame can be transmitted without interruption by a TT frame. This tends to be less efficient as compared to the preemption integration technique. The shuffling integration technique, on the other hand, lets the RC frame transmission take place to finish before the prescheduled TT frame can be started. This technique goes against the main essence of TTEthernet for a hard real-time traffic transmission by making the TT traffic wait. Thus, for a secure and efficient TTEthernet clock synchronization, it is important to clearly define which integration technique suits better one's security requirements. It can be said that pre-emptive and timely-block serve well for this research as both prioritize PCFs over other traffic types; although, it can be argued that other QoS requirements can be affected depending on what those requirements are.

This research project is focused on investigating the security of clock synchronization protocol in a TTEthernet-based IIoT and the effect a security breach can have on an organization as a business. Therefore, it is vital to explain the need for TTEthernet clock synchronization and the importance of security to validate the dimensions taken in this

research.

TTEthernet is important for real-time IIoT applications because it supports mixed-criticality traffic on a single Deterministic Ethernet (DE) backbone. PTP of the IEEE1588 does likewise in supporting mixed criticality traffic but TTEthernet is fault-tolerant by design as it supports multi-SMs and CMs for hard real-time traffic communication. Nonetheless, event-triggered, standard Ethernet traffic can still be communicated on the same channel whenever there is no TT traffic communicated or scheduled to communicate.

According to the SAE S6802 std, (Ethernet, 2016), TTEthernet only covers the network aspect for mixed-criticality traffic communication. Therefore, the fundamental requirement is to have all network devices regularly synchronized by making sure traffic transactions follow the expected schedule for secure and safe network operation. Thus, the seamless time notion among all network devices offered by clock synchronization protocol is key to the concept of TTEthernet for real-time IIoT systems.

Security, as with most computer networks, is a major issue in the modern IIoT. Availability and reliability are two of the most important security objectives in industrial networks. Most often, industrial organizations can't afford to halt the production line to run a security patch or upgrade firmware for financial reasons. Hence, this is one of the weak links in the industrial sector and an attractive spot that bad actors can exploit. Furthermore, the operational legacy systems and wireless segment of an IIoT, where proprietary sensors have increased the potential attack surface, increase the security vulnerability within IIoT networks. Therefore, this is a timely and important research topic for organizations that depend on the services of TTEthernet clock synchronization for real-time data communication.

Security breaches can take different forms including the MitM attack which is the case in most computer networks and delay attack which is a typical attack type that targets the clock synchronization protocol. Similarly, the nature of the attack may come from internal or external attackers that have different motives 4. Most importantly, attackers can access different resources to accomplish their objectives. With the rise of AI tools, AI-powered security attacks can be more sophisticated and difficult to stop

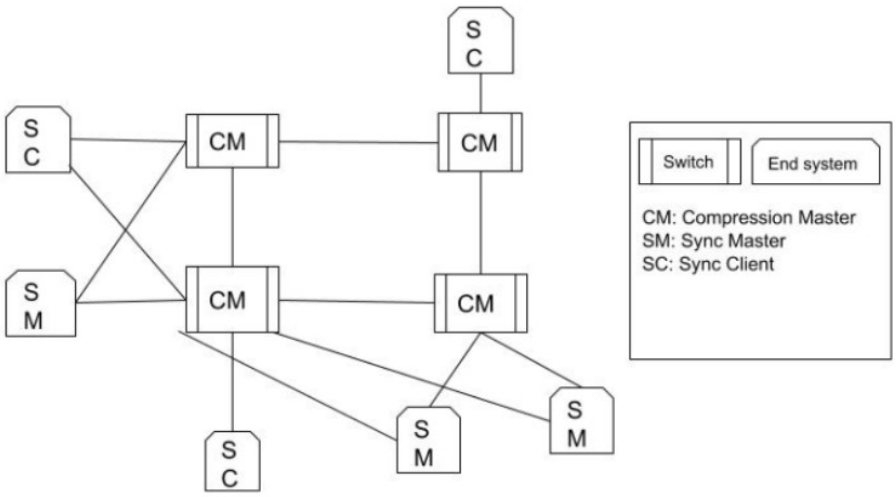

Figure 1.3: Fault-tolerant TTEthernet-based clock synchronization

than traditional security breaches (Tabassum et al., 2022). This is mainly because they can adapt in real-time and learn from defences they encounter to modify their strategies to bypass protection systems while traditional attacks mostly depend on predefined and more predictable patterns. Moreover, AI tools can be used to analyse data and find the weakest link to target and launch an attack, (Alzarqawee and Fritsch, 2023). More worrying yet, such an attack can be automated to launch multiple attacks simultaneously at a scale making it difficult for traditional defence systems to cope with. Nonetheless, AI tools can also be used to defend against AI-powered security threats by leveraging their capability to detect, predict, and respond to such attacks. Considering the focus is on detecting anomalous sync frames and designing a security algorithm to defend against security attacks that manipulate the latency of sync frames, anomaly detection is a defence mechanism investigated in this project. Anomaly detection is a technique used in data analysis to identify patterns or data points that deviate significantly from the expected behaviour or statistical norm in a dataset. Anomalies, also known as outliers or novelties, are data points that do not conform to the regular conduct of the system or population and may represent significant changes, errors, or anomalies in the underlying data. It is a critical tool for detecting and addressing issues in various industries, including finance, healthcare, manufacturing, and security among others. It can help organizations identify potential problems early, improve operational efficiency, and prevent costly errors and losses. Anomaly detection techniques include rule-based detection, machine learning

techniques, time series analysis, and statistical methods, among others, (Jayabharathi and Ilango, 2022). Considering the requirements in this research, the first two are investigated further in more detail in chapter 8.

Hence, hypothetical solutions are outlined in this section. These solutions are mainly in relation to the security vulnerabilities identified in the previous sections and a few more highlighted in this section. Although, AI-powered security breaches are a threat, a fitting AI-powered security solution can be devised. In this research, considering the main security threats highlighted MitM and delay to synchronization frames, anomaly detection is investigated as a fitting security solution. Thus, anomaly detection techniques are explored in more detail. Finally, having investigated the subject matter in general terms, it is imperative that the main background of study is explained below to give a general perspective to the selection of the topic for this level of research.

1.3 Background of Study

The researcher started this doctoral program after seven years of working as an IT Network Engineer. He used to manage more than 40 network switches and around 25 physical and virtual servers as well as more than 600 workstations. The network he managed included 32 wireless access points (WAPs) feeding thousands of mobiles and other wireless devices. Hence the effect of an unsynchronized server on network security was clear to him all along. Moreover, he studied wireless sensor networks (WSN) for his master's thesis which happened to be the most interesting subject that he has done so far. He has since been interested in doing further research work on the subject area which led him to this project. WSN has been an attractive research subject for some time. It's been labelled as one of the most important technologies expected to change the way we live in the 21st century, (Ruiz-Garcia et al., 2009). However, because sensor nodes are mostly vendor-specific with minimal energy resources, more research is required to formulate standardized protocols for effective and secure functionalities. Recently, it has been incorporated as one component of yet another more attractive emerging technology – the Internet of Things (IoT). IoT, which is mostly composed of wired and wireless devices within the same network, has now been widely adopted by organizations for its cost

and performance-related benefits. However, it is also clear that most organizations have deployed vendor-specific, out-of-the-shelf, cheap legacy sensor devices which cannot be easily standardized or configured with the complex security solutions which are important in the current environment where a security breach has become the norm.

The wired Ethernet network, however, has been more stable ever since it was officially standardized by the IEEE in 1985, (SA, 2003) with globally standardized TCP/IP, HTTP and other commonly used protocols compared to the wireless IoT networks. Nonetheless, vendors and the research community have been drawn to improve the existing wired Ethernet network to support mixed-criticality requirements, mainly the TT and event-triggered (ET) traffic communication. One of the recently adopted developments in the industrial sector, among others, is the TTEthernet data communication platform. This has enabled the TT real-time traffic communication with an absolute priority over the RC as well as BE traffic types.

However, IIoT is widely adopted by organizations but it would benefit them if more research were done on the security side for a safe and secure data flow within their networks. Hence, as a professional doctorate student, the researcher aspires to investigate the security implication of vulnerable clock synchronization on a TTEthernet-based IIoT network, from a business point of view and explores the effect of such a security breach on the business as a whole and ways of monitoring and protecting against such security threats.

Finally, the outcome of this research project is an important addition or extension to the existing knowledge domain among vendors and researchers alike. Hence, all organizations interested in the deployment of a TTEthernet-based IoT; mainly, the Avionics, energy, transport and industrial automation sectors among others, would benefit from the knowledge contributed in this paper.

1.4 Significance of the Study

This research work is a continuation of the TTEthernet-based IIoT revolution. Quite a few researchers have started the initial work on the security issues within the TTEthernet infrastructure. Thus, it can be said that this is a continuation of those works with a specific

focus on the security vulnerability within the clock synchronization protocol.

A growing body of literature has recognized the security challenges inherent in the TTEthernet infrastructure mainly because it is primarily designed for wired LAN communication,(Lisova et al, 2014; Peon et al, 2014; & Fan et al, 2018). Although this research cannot promise to make TTEthernet work on a wireless medium of communication, the solution offered enhances the fault-tolerance aspect by providing multiple routes of communication for traffic between the network switches and WAPs. TTEthernet-based clock synchronization involves broadcasting of sync frames using UDP which is potentially a target for malicious attacks, (Herrera, 2018). Attack types targeting the clock synchronization protocol in a TTEthernet-based IIoT include Denial of Service (DoS), Delay manipulation, interception and modification, rogue master, relay and spoofing as well as Absolute Slot Number (ASN) attack among many others. These are addressed by setting two-layered-latency thresholds to nullify maliciously delayed sync frames after a thorough investigation of how and why these attack types target the clock synchronization protocol.

The potential research outcome will be an important addition to the existing knowledge within the research community as well as interested organizations. All organizations which have already implemented TTEthernet-based network infrastructure including but not limited to the Industrial automotive, Avionics, Transport, and Energy sectors, as well as those planning to implement will benefit from a secure TTEthernet-based network.

1.5 Potential Contribution to Knowledge

Industries are still employing various legacy and vendor-specific sensor devices which have very little and varied security capabilities. Hence, it is a tough task to design a seamless TTEthernet network which is cross-sector with a standardized security solution. Nonetheless, after the successful completion of the research project, there will be a contribution to knowledge as follows:

A delay attack on the clock-synchronization protocol has been considered, (Lisova et al., 2016a). This research, however, intends to investigate all types of attacks on

end systems exploiting the clock-synchronization methods. Therefore, a detection and protection technique against such exploits is a novel contribution to knowledge.

Existing clock synchronization protocols in an IIoT setup are compared in relation to some important deciding factors. This should help interested organizations to clearly see why a certain TTEthernet clock synchronization is a better option in relation to some relevant IIoT deployment objectives.

1.6 Research Development

This thesis is composed as follows: The Introduction Chapter lays down the introductory knowledge base by including sections on the Introduction of the research work itself, the background of Industrial Networks, and the Industrial Internet of Things (IIoT) which explores the safety and security implications in the cyber-physical systems, TTEthernet for IIoT and the security implications, clock synchronization in a TTEthernet-based IIoT, security threats in the clock synchronization for a TTEthernet-based IIoT, and outlining a security solution for TTEthernet clock synchronization in an IIoT. This is followed by the background of the study that covers the reason behind the selection of the topic in this research which is further followed by the significance of the study and potential contribution to knowledge.

The literature review, in Chapter 2, critically explores previous research work and is divided into three sections as more details are sought by investigating the three main dimensions of the topic, mainly, exploring TTEthernet clock synchronization, the security of TTEthernet clock synchronization, and the types and implications of security breaches in a TTEthernet clock synchronization.

A Chapter on the security threats and attack scenarios in TTEthernet clock synchronization for IIoT follows the literature review. It explores the subject even more by delving into the security threats in the TTEthernet clock synchronization that covers subsections on threat models, system assets and their potential attack surface, system vulnerability, adversary goals, attacker types, and security objectives. This precedes a section on attack scenarios on the TTEthernet clock synchronization which, in turn, explores two of the commonly known security attacks targeting the TTEthrnet clock

synchronization, mainly the MitM attack and Delay attack and investigates hypothetical solutions to those attacks.

Research formulation, in Chapter 4, presents the main research aim and objectives followed by the research design which describes the type of research, benchmark resources used for this research, and the research strategy employed. This is further followed by the data collection and analysis that explains how data is collected and analysed, the research population, the mechanism to assure the quality of study, and the resources required to carry out the research.

This is followed by the environmental setup for the network models which presents the required steps for the network modelling and simulation, end system configuration for the model implementation, modelling and simulation software, and the network topology.

Chapter 7 discusses the impact of communication channels on the latency of synchronization frames: a comparative analysis of wired and wireless channels. It follows the discussion of the simulation results which tests the effect of fault injectors on the designed network model in Chapter 6. It explores the selection of a fitting integration technique and the concept of latency in TTEthernet clock synchronization before it presents the results before the fault-injector is applied, results where the injected fault breaches the maximum latency threshold, and results where the injected fault does not breach the maximum latency threshold.

Chapter 8 presents the comprehensive analysis and implementation of anomaly detection solutions which starts by introducing the Chapter that covers the comprehensive analysis of anomaly detection models which, in turn, explores the machine learning methods for anomaly detection, rule-based anomaly detection, and the case for rule-based anomaly detection as the fitting solution to the research. This precedes sections that introduce the security solution from a presentation perspective and the core algorithm: conceptual design and pseudocode breakdown.

The design and configuration of multiple network models to test the efficacy of the core algorithm and discussion of simulation results is presented in Chapter 9. This discusses two sections on designing three network models and evaluating the proposed security solution across the three network models to mimic the different network

environments TTEthernt-based IIoT can be deployed. This section covers subsections on the introduction, configuring network environments for simulation execution, testing protocols, and results and analysis of simulation.

This is followed by a Chapter on interpreting and contextualizing the proposed security solution which presents the overview of the proposed security solution, interpretation of findings that covers the test results from simulations run on wired LAN and implication of the wired and wireless communication, proposed security solution against existing literature, implications of proposed security solution, and a conclusion to the chapter.

The overall conclusion is presented in Chapter 11 which covers sections on the summary of the whole research, limitations and future research work which is composed of subsections on limitations and future research. The research is concluded by presenting the final thoughts, a section under the overall conclusion.

Appendices and bibliography sections are presented towards the end of the research document, helpful for researchers to understand the details of the research.

Chapter 2

Literature Review

2.1 Exploring TTEthernet Clock Synchronization

Previous researchers have shown the importance of having added capabilities on the traditionally used Ethernet platform to address the growing demand of organizations for synchronous and asynchronous traffic communication, on the same channel. This has led manufacturers and academics to investigate alternative add-ons to the standard transmission platform – the Ethernet, or a new platform solution altogether. WirelessHART (Song et al., 2008), for example, an extension to the wired HART, is considered an option to address the real-time requirements for the IIoT applications. However, it only supports one type of traffic - the TT traffic. Besides, WirelessHART allows dynamic allocation of certain time slots on a contention basis which, in turn, creates undesired latency. Time Sensitive Networking (TSN), however, supports mixed traffic types and better flexibility in scheduling over the statically configured schedules used by TTEthernet, (Zhao et al., 2018). nonetheless, considering the focus on clock synchronization, in this research, TSN still uses the PTP protocol of the IEEE 1588 standard where only one Grand-Master with a single point of failure provides the clock synchronization frames to all member nodes; hence, it lacks the fault-tolerant synchronization mechanism provided by TTEthernet where multiple SMs and CMs are used to communicate PCFs to all member nodes, to align their clock drift, (Li et al., 2023); (Zhao et al., 2018). on unaffected, for the multi-cluster synchronization operation. It was proposed in (Wang et al., 2018) that inter-cluster clock synchronization could be done

using a gateway to forward PCFs between two networks within a cluster. Nonetheless, TTEthernet- SAE AS6802 std, (Ethernet, 2016) has simplified this for TTEthernet-based clock synchronization by specifying a method for intra-cluster clock-synchronization to guarantee the required QoS in a multi-cluster TTEthernet network as in (Tang et al., 2018). They proposed a synchronization mechanism where the intra-cluster compression is modified leaving the inter-cluster synchronization.

TTEthernet is a wired technology by design, (Lisova et al., 2014); hence, it falls short of being the complete solution in today's IIoT applications where wireless sensors are the main components. Thus, an extension to the wired TTEthernet setup for the wireless link of an IIoT is required. (Peón et al., 2014) has studied the possibilities of achieving this wired/wireless TTEthernet-based IoT network and has suggested that an alternative or an extension to the SAE AS6802 clock-synchronization protocol should be investigated for the wireless segment. They further recommended that a Global Positioning System (GPS) like approach should be considered.

Therefore, it is clear that TTEthernet is an important clock synchronization protocol for certain applications dependent on real-time traffic communication. The exploration of TTEthernet clock synchronization from different researchers' viewpoints, in this section, is followed by the security implications which is an important objective in this research.

2.2 The Security of TTEthernet Clock Synchronization

With regards to securing the TTEthernet-based IoT networks, in general, efforts have been made but previous researchers have mostly focused on the security requirements and the standardization of certain security policies and procedures but not technical security solutions (Choi et al., 2018). The same authors have recommended their own security solution in the shape of an information security sharing management procedure. They have designed an information security sharing reference model by dividing the information sharing system into information collection, information verification, information analysis and information sharing. Similarly, three different medium access control (MAC) protocols have been proposed and comparatively analyzed in (Peón et al., 2015) for a more suitable security solution on the extended TTEthernet platform for the

wireless segment of an IIoT. However, this still falls short of proposing a concrete solution for the clock synchronization protocol, for example.

Authentication has been a common theme among a few of the contemporary research to secure messages communicated between the control system and networking devices or end systems in TTEthernet networks. (de las Morenas et al., 2020) proposed TLS/SSL encryption while (Zhao et al., 2019b) took a slightly different approach by investigating the scheduling aspect of TTEthernet-based systems and proposed a mixed linear programming formulation which considers the authentication mechanism. (Zhang et al., 2016), on the other hand, studied the clock discrepancies' in a TTEthernet clock synchronization and proposed a compensated algorithm based on modified least squares which is not purely a security solution. In (Martins et al., 2014), Hash-based message authentication is proposed for networked control systems to ensure safety, reliability, and fault tolerance factors. However, considering the limited processing resources available to wireless sensors, this is designed for the wired TTEthernet-based platforms but not the wireless segment. Similarly, (Fan et al., 2018) suggested blockchain as a security solution for clock synchronization. However, there is not enough research to support or negate this other than the fact that it has been long established that blockchain is not energy efficient. Hence, it is not an ideal solution for an IIoT which includes sensors that can easily be drained of power in a complex security configuration, (Banđur et al., 2019).

With the rise of AI tools, AI-powered security breach is a threat that all computer network administrators are concerned about. Thus, it's imperative that it is explored further to understand the complexity and find ways to stop such security threats. With vehicles advancing towards increased connectivity and autonomy, the need for flexible and high-bandwidth network architectures becomes critical for their security challenges (Toghuj and Turab, 2023). The article highlights significant security vulnerabilities within the clock synchronization in TSN-based networks leading to delays, jitters, and ultimately the failure of time-critical applications. AI-powered security attacks could be more sophisticated than the traditional forms of security attacks and use machine learning algorithms to identify vulnerabilities to launch highly targeted attacks (De Vincenzi et al., 2024); (Przybylski et al., 2023). Thus, they can be particularly challenging to

detect and mitigate due to their adaptive nature and ability to mimic legitimate network behaviour (Toghuj and Turab, 2023); (De Vincenzi et al., 2024). Solutions proposed to counteract the AI-powered security threat on the TTEthernet clock synchronization in IIoT include the implementation of robust encryption and authentication methods, the use of intrusion detection systems, and the redesign of network architectures to isolate critical communication paths from potential sources of interference. (Tariq et al., 2020); (Toghuj and Turab, 2023); (Lichtsinder, 2022). Furthermore, (De Vincenzi et al., 2024), added the use of intrusion detection systems specifically designed to detect anomalies in TT networks, and employed AI-driven defence mechanisms that can learn and adapt to emerging threats. This article stressed the importance of ongoing research and development of new security frameworks that can effectively counter AI-powered attacks on TTEthernet and other time-sensitive networks in automotive and industrial environments.

Considering the nature of IIoT where legacy devices and miniature sensor devices are included, heavy authentication or blockchain-related security solutions are not ideal for most deployment scenarios. Nonetheless, complex AI-powered security threats are surfacing due to the quickly surging AI tools. Thus, a fitting security solution should be investigated to counteract these security threats. It's important to understand the magnitude of these threats by exploring the types and nature of security threats before a security solution can be devised to resolve security threats on the TTEthernet clock synchronization in the IIoT.

Author / publication date	TTEthernet scenario	Data	Problem considered	Methods employed	Solution sought	Result	Note
(de las Morenas et al., 2020)	Securing IoT-based applications for building and factory automation.	Existing literature, as well as use case studies, are used.	Security threats in the messages communicated between agents or between agents and remote controlling systems.	Two case studies are analyzed to establish the security threats and validate the solutions suggested.	Security solution for IoT networks	Keys and certification for MQTT broker and client, TLS/SSL based encryption and JADE-S framework to encrypt data.	
(Zhao et al., 2019b)	Security-aware scheduling for TTEthernet-based real-time automotive systems	Modeling and simulation using the LINGO solver	Information security and functional safety in scheduling design of TTEthernet based automotive systems	Experiments based on security mechanism and security models	Securing the scheduling design in TTEthernet-based automotive systems	Proposed a mixed integer linear programming formulation subject to authentication mechanism constraints and other design constraints	Not a security solution for communication protocols or clock synchronization.

Table 2.1 – continued from previous page

Author / publication date	TTEthernet scenario	Data	Problem considered	Methods employed	Solution sought	Result	Note
(Zhang et al., 2016)	Clock synchronization compensation of time-triggered Ethernet	Simulation based on C++.	Understanding and resolving clock discrepancies in TTEthernet	A simulation is used to prove compensated algorithm based on modified least squares	Investigating and resolving the clock discrepancies in TTEthernet	Compensated algorithm based on modified least squares is proposed.	Not a security solution
(Tariq et al., 2020)	Explores the synchronization of time-triggered networks using external synchronization sources	External setups like PTP and GPS are used measure synchronization accuracy, latency, and jitter	Achieving precise and reliable TTEthernet clock synchronization	Theoretical analysis and practical experiments are used to analyse and access the impact of external synchronization sources on the TTEthernet synchronization accuracy	Securing TTEthernet clock synchronization using external synchronization methods	External synchronization methods effectively improve the accuracy and reliability of TTEthernet clock synchronization	

Table 2.1 – continued from previous page

Author / publication date	TTEthernet scenario	Data	Problem considered	Methods employed	Solution sought	Result	Note
(Toghuj and Turab, 2023)	Explores automotive Ethernet architecture on connected and autonomous vehicles	Data collected from simulating practical automotive networks	Securing the Ethernet-based communication for autonomous vehicles	Theoretical analysis and practical experiments are used to validate new encryption and authentication systems for in-vehicle networking	Improving the security and performance of automotive Ethernet networks	Implementing TSN standards to improve the reliability and security of Ethernet-based in-vehicle networking	Not a TTEthernet-based solution
(De Vincenzi et al., 2024)	Systematic review of security threats and their countermeasures in the automotive Ethernet architecture	Comprehensive review of existing literature	Security threats against automotive Ethernet infrastructure	Systematic review methodology is used to analyse existing literature	Providing an effective countermeasure for security threats against automotive Ethernet networks	Categorized known security threats along with their impact on automotive Ethernet networks and their countermeasures	Not a TTEthernet-based solution

Table 2.1 – continued from previous page

Author / publication date	**TTEthernet scenario**	**Data**	**Problem considered**	**Methods employed**	**Solution sought**	**Result**	**Note**
(Przybylski et al., 2023)	Explores communication systems for Ethernet-based networks in aircraft	Data used includes results of simulations and empirical data from existing literature	Vulnerabilities within the Ethernet communication systems in aircraft	Theoretical analysis and practical simulations are carried out	Providing robust communication protocol that enhances the reliability and security of aircraft communication systems	identified Ethernet-based communications that improved bandwidth and integration capabilities but also identified new security challenges	Not a TTEthernet-based solution

Table 2.1: Sample of existing literature on the subject matter

2.3 Types and Implications of Security Breaches in TTEthernet Clock Synchronization

Considering the adversary types (internal/external) and adversary targets, clock-synchronization can be susceptible to different malicious attacks, (Steiner, 2013). Synchronization PCFs are broadcasted by the CM to all the SMs and SCs, (Steiner et al., 2009) using the UDP protocol, which is potentially a target for malicious attacks, (Herrera et al., 2018). Hence, securing clock synchronisation is of utmost importance in the TTEthernet-based IIoT networks.

Researchers have studied different attack types targeted at the clock synchronization protocol within the IIoT. Delay attack was investigated as a major security attack in (Ullmann and Vögeler, 2009). Nonetheless, it is only one of the many attack types targeting clock synchronization protocol, (Lisova, 2018). DoS, delay manipulation, interception and modification, rogue master, spoofing, relay, interception and removal, time source spoofing and cryptographic performances are some more types of attacks pointed out (Mizrahi, 2014); (Lisova, 2018). Another approach taken to understand possible security attacks against the clock synchronization protocol in an IIoT involves a malicious attack on the Absolute Slot Number (ASN) which targets the specific time slots allotted to nodes for the exchange of a frame and an acknowledgement with their neighbouring nodes; and Time Synchronization Tree Attack which aims to disrupt the network traffic forwarding capability (Yang et al., 2016a). A thorough examination of the Flooding Time Synchronization Protocol, (Huang et al., 2013) revealed the Sequence Number (SeqNum) attack, node replication attack, global time attacks and traitor attacks as important security breaches on the clock synchronization protocol which they deemed to require further scrutiny.

DDoS is another form of security attack that is used to breach the TTEthernet clock synchronization. (Kawamura et al., 2017) investigated the detection techniques for DDoS-type security breaches on the clock synchronization in IIoT. They proposed an event detection module that uses information collected from NTP during the clock synchronization. Similarly, (Yin et al., 2018) explored ways to defend against the DDoS

type of security attacks and suggested a detection and mitigation mechanism using a software-defined IoT framework. (Guo et al., 2018) was more concerned about the end-to-end delay in RC traffic in a TTEthernet network and proposed a system that involves an end-to-end latency analysis. Although this project is more focused on the TT traffic, it helps to understand how decisions to secure the TT traffic affect other traffic types for which TTEthernet is known for. Similarly, (Alghamdi and Schukat, 2017) explored methods of protection against internal attack in PTP clock synchronization. They suggested that a supervisor node can be set up to analyse member nodes' clock drift. This is still focused on protecting the PTP protocol; unlike the TTEthernet which is the main subject in this project. Most importantly, this solution directly refers to the very distinction between these two protocols. PTP depends on a single grand master while TTEthernet is based on multiple synchronization and CMs.

Security breaches have targeted clock synchronization, in an IIoT, and carried out as universal system disruption as it remains the main asset in most industrial networks due to the real-time requirements associated, (Lisova et al., 2016b). For example, accurate timing signals have been exploited across the electric power systems from the generation plant down to the distribution substations and individual smart grid components, (Shepard et al., 2012). Similarly, smart grid IoT deployments depend on a coherent time notion throughout the deployment setup as the fault detection and protection systems, fault recording, as well as the substation and wide-area monitoring systems, require a shared notion of time, (Lévesque and Tipper, 2016).

It has long been acknowledged in (Ullmann and Vögeler, 2009) that delayed synchronization frames can damage the reliability and network performance which includes the interruption of operations and unintended behaviour of industrial applications. This has recently been updated in (Lei et al., 2022) which stresses that delay in synchronization frames can cause potential performance degradation and high risks in safety-critical applications, including but not limited to autonomous driving. They went on to assert that this can negatively impact collaborative perception, in a network. An error in the clock synchronization of an IIoT, maliciously exploited or otherwise, causes sensors to produce faulty reports which in turn affects control decisions

and the general functionality of critical services which normally feed off the sensor reports, (Annessi et al., 2018). Synchronization attacks targeting the communication protocol have been a common trend as synchronization protocols are not outlined with security support (Smache et al., 2019). They further stressed that they attract security breaches due to their distributed and real-time nature. Hence, they suggested a machine learning-based detection system as one way of defending against such types of security breaches. Synchronization attack, sometimes referred to as de-synchronization, involves injecting packets with fake sequence numbers of the control frames by the malicious actor, to de-synchronize endpoints, (Tsiknas et al., 2021). These authors suggested authentication as a possible solution to this type of attack. One major constraint when it comes to securing IIoT is the energy limitations associated with sensor nodes (Qiu et al., 2017). Hence, the energy efficiency of sensor nodes is a key factor which needs to be considered when devising a security solution for clock synchronization.

Although every aspect of the TTEthernet-based IIoT security deserves further research, this paper focuses on the security threats within the clock-synchronization protocol, which is a very important component of the TTEthernet as the fundamental steppingstone for a safe and secure data exchange among the IIoT network end systems. Below is a summary of the relevant publications and their approach to resolving their respective research questions, which are presented in table 2.2.

Author / publication date	TTEthernet scenario	Data	Problem considered	Methods employed	Solution sought	Result	Note
(Lisova, 2018)	Monitoring for secure clock synchronization.	Existing literature, as well as use case studies, are used.	Delay attack on clock synchronization	AVISPA and OMNET++ were used for attack and solution evaluation as well as Matlab for calculation and simulation. Game theory investigates the interaction between adversaries and monitors.	Resolving cyber-attacks pertaining to the delay of clock synchronization frames delivery.	Distributed monitoring solution is proposed.	PTP of the IEEE 1588 standard is used as a clock synchronization protocol.
(Steiner, 2013)	Candidate security solutions for TTEthernet.	Overview of existing literature	General security issues in TTEthernet-based networks	Designed threat model followed by exploring security mechanisms	Cross-industry security solution for TTEthernet-based networks	Outlined general security solutions in line with the sketched threat model.	
(Guo et al., 2018)	End-to-end delay analysis for RC traffic in TTEthernet	Overview of existing literature	Understanding and rectifying the RC traffic delay in TTEthernet	Simple case study to compare the proposed way of analysis to existing methods.	Exploring better ways of analysis for the RC traffic in TTEthernet	Fine-grained Worst-case end-to-end delay Analysis (FWA)	

Table 2.2 – continued from previous page

Author / publication date	TTEthernet scenario	Data	Problem considered	Methods employed	Solution sought	Result	Note
(Alghamdi and Schukat, 2017)	Methods of protection against internal attacks in PTP time synchronization networks.	Attack types and categories statistics as well as drift between slave and grand-master timestamps with and without PTP	Byzantine and other internal security threats on PTP clock synchronization	PTP clock synchronization is used	A supervisor node to analyze devices clock drift.	Protecting against internal attacks in PTP-based clock synchronization.	
(Kawamura et al., 2017)	Detection technique for DDoS attack on IoT	Number of events detected	DDoS attack against IoT	Experiment carried out on a compact computer board.	Event detection module using information collected from NTP during synchronization service.	Ability to detect DDoS attacks on IoT.	

Table 2.2 – continued from previous page

Author / publication date	TTEthernet scenario	Data	Problem considered	Methods employed	Solution sought	Result	Note
(Yin et al., 2018)	Detection and mitigation mechanism for DDoS attack on IoT	Experimenting with statistics data on the number of data packages received by the SD-IoT switch and SD-IoT controller	DDoS attacks against IoT networks	cosine similarity of the vectors of the packet-in-message rate to determine the presence of DDoS attack.	Detection and mitigation mechanism with software-defined IoT framework for DDoS attack	Finding ways of detection and mitigation against DDoS attack on IoT	
(Smache et al., 2019)	Detecting synchronization attack in IIoT	IIoT synchronization attack data.	Attacks targeting the synchronization protocol in IIoT.	Machine learning algorithm for the detection of IIoT synchronization attacks. OpenWSN simulator is used.	Detection of synchronization attack based on characteristics of synchronization protocol	A methodology for the detection of synchronization attacks based on characteristics of the synchronization protocol is proposed.	

Table 2.2 – continued from previous page

Author / publication date	**TTEthernet scenario**	**Data**	**Problem considered**	**Methods employed**	**Solution sought**	**Result**	**Note**
Our contributions	Securing clock synchronization in a TTEthernet-based IIoT	sync frames communicated within the network of devices in a TTEthernet-based IIoT	Latency-related Security breach against the TTEthernet-based clock synchronization protocol in an IIoT	Anomaly detection model uses fault injector technique in a simulation environment to identify anomalous sync frames.	To be able to detect sync frames delayed by more than the accepted latency threshold	a rule-based anomaly detection model is designed to monitor sync frames and flag those breaching the rules set up for the acceptable latency threshold.	

Table 2.2: Sample of existing literature on the subject matter

Chapter 3

Security Threats and Attack Scenarios in the TTEthernet Clock Synchronization for IIoT

3.1 Security Threats in the TTEthernet Clock Synchronization

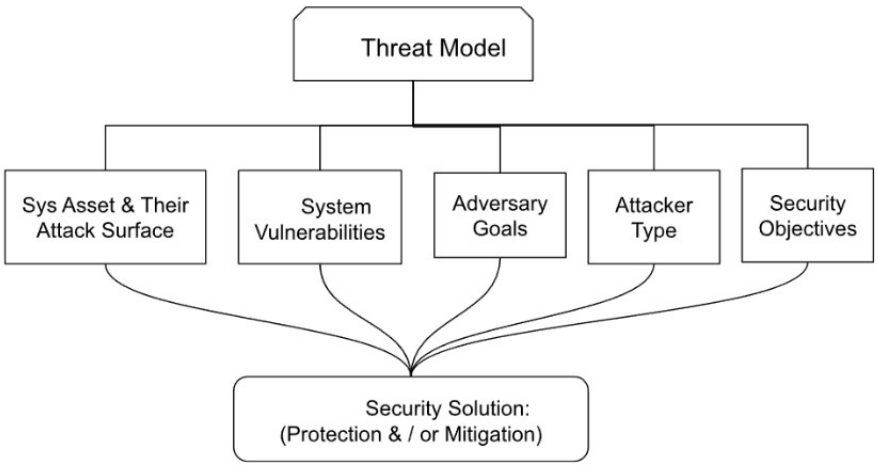

Figure 3.1: Threat Model Analysis

Understanding the different scenarios, a security breach can take place on the clock synchronization protocol, is a good start to defend and/or mitigate against the attack. Hence, this chapter addresses the threat model which exposes many of the avenues

whereby malicious attackers would perform their ill-intentioned act. This threat model should investigate the main system asset which requires protection and its attack surface, the system vulnerability of the identified asset, the adversary goals which motivate the attack, and the specific security objectives this research pledges to find ways to improve the security of the clock synchronization, the main asset in this case. These are the steps required to get to the ultimate security solution in this research, be it a protection or mitigation mechanism.

3.1.1 Threat Model

The threat model can be considered a guide to help understand the possible ways an attacker would perform their act to break a target network or database to meet their ends. Understanding security attackers in terms of the main reasons why they would want to perform their malicious attack and how they would perform such an attack in a typical attack scenario helps to understand how to defend and/or mitigate these attacks. It's, therefore, important to identify the main asset that requires protection and the attack surface on which attackers would instigate their attacks, to realize their final adversary goal.

3.1.2 System Assets and their Potential Attack Surface

Almost everything digital can be secured better as the security process dynamically changes; trying to cope with emerging technologies and techniques for implementing them securely. The unfortunate effect of the IT/OT convergence has increased the attack surface requiring the security competence to grow parallel to it. Thus, one can always work to better the security of a digital asset. A security attack targeting the weak authentication capabilities of Process sensors is a glaring example of the IIoT evolution and the consequence of the exposure of OT due to the IT outlet. In this research, however, the main asset under consideration for a security solution is the clock synchronization protocol in a TTEthernet-based IIoT. It's the foundation for a network where TT traffic is communicated. Every member device in a cluster where traffic is communicated in a pre-defined time slot needs to have a common notion of time. Thus, every time a member

node deviates, from the ideal time, it's expected that it re-syncs itself to the master or source node to avoid undesired consequences ranging from a simply delayed file transfer to a fatal chemical spill caused by a delayed or dropped synchronization frame. The criticality of an unsynchronized node in an IIoT network where sensors collect safety critical readings including but not limited to hazardous chemical levels, humidity and temperature levels has been discussed in Chapter 1.

The fact that clock synchronization is critical to TT traffic in an IIoT makes it an interesting target for malicious attackers. The main attack surface must be explored before one can contemplate the existence of vulnerability within the clock synchronization which can be exploited by nasty hackers. One way of identifying these attack surfaces is to understand the IoT architecture in terms of the specific layers outlined by researchers. Although there have been so many suggestions by vendors and research groups, as of today, there are no universally standardized layered architectures for IoT (Burhan et al., 2018). However, different researchers have suggested mostly similar but slightly different three to five-layered architectures. The three-layered architecture, which refers to the perception, network, and application layers, was conceived during the initial stages of the IoT architecture development; although it has still been predominantly maintained by researchers as recently as 2020 (Rehman et al., 2020). Others have come up with a four-layered architecture which includes the Sensors & Actuators, Networking, Data Processing and Application layers (Varga et al., 2017). This classification remains similar to the three-layered architecture with the exception that the network layer has grown into a network layer and data processing layer. Note that every layer of the IoT ecosystem is briefly described below to show which layers would most likely attract an attacker's longing to perform their ill-intentioned act. The five-layered approach as explained in (Sethi et al., 2017), includes the perception, transport, processing, application, and business layers. As compared to the four-layered approach, this has shown certain changes in the names used without changing the role they reflect too much. For example, the perception layer performs the same role as the sensors and actuators layer in the four-layered architecture. Likewise, the transport layer does the same role as the network layer, although they may seem completely different from the OSI model perspective. The

business layer is a new perspective added to the five-layered architecture. As pinpointed below, these distinct layered structures should help to formulate a security solution for threats at every layer of the IoT architecture.

IoT Layer	The role played in the IoT Structure
Sensors & Actuators / Perception	The sensors and actuators layer sometimes referred to as the perception layer, are physical devices or end systems which are capable of sensing certain parameters set or the environment and collecting information.
The Network Layer / Transport Layer	This layer refers to the interconnection of intelligent physical devices, networking devices and network servers. It involves routing and communication of sensor data to and from the perception layer and the Data processing layer using some communication channels; mainly, the LAN, and Wi-Fi among others.
Processing Layer	As the name signifies, the processing layer collects data from the network layer and does the task of data storing, analysis, processing, and classification. Technologies employed at this layer include databases and cloud computing among others.
Application Layer	This is where application-specific services are rendered for users.
Business Layer	The business layer refers to the general IoT ecosystem including all the applications and business modules.

Table 3.1: IoT Architecture explored.

Given the nature of sensor nodes, physical tampering could be an obvious example at the sensors & actuators layer as DDoS or MitM attacks could be good examples of security threats at the networking layer (Lisova et al., 2015). These authors believe disruption, eavesdropping and hijacking are three of the main adversary targets in IoT networks in general. Having said that, the main attack surface attackers would be drawn to in the clock synchronization protocol are the networking layer as explained above as well as the sensors and actuators layer, where node insertion and node manipulation among others are involved. Therefore, the other layers mentioned in the table above are less likely to attract attackers; hence, they are discarded from further investigation in this paper. Furthermore, it should be noted that TTEthernet is designed for traffic communication in a network where clock synchronization sets out the foundation for further data transaction, (Ethernet, 2016). Thus, the focus in this research is limited to the network layer of the IoT architecture as set out in Fig 3.2; although, the sensors and actuators layer is also being

considered as a potential entry point or means to instigate a MitM type of attack at the network layer. Hence, it can be considered an important attack surface for this paper.

The main attack surface in the TTEthernet clock synchronization is narrowed down to the sensors & actuators layer and the network layer in the IIoT infrastructure. Hence, the security of a TTEthernet clock synchronization starts by identifying system assets and understanding how these system assets can be secured. Therefore, the main vulnerabilities within the identified system assets and attack surfaces within the IIoT infrastructure must be considered when designing a security solution.

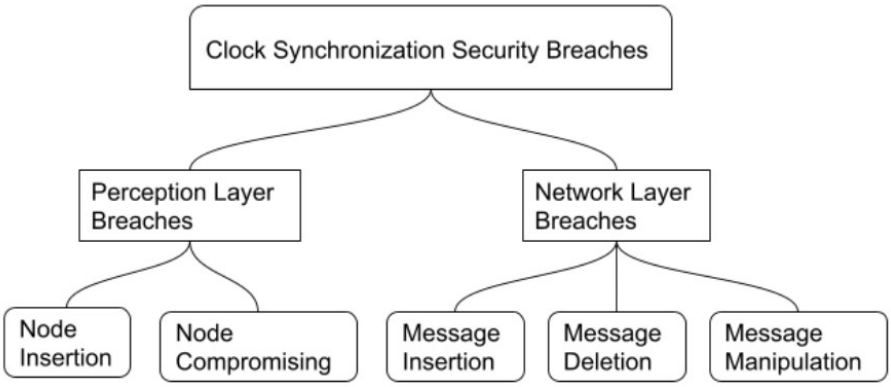

Figure 3.2: Clock synchronization Security Breaches Classified

3.1.3 System Vulnerability

Clock synchronization is the foundation for most network communications, especially in networks dependent on real-time traffic transmission. Hence, clock synchronization may be an important target for many malicious attackers. The network layer is the main attack surface intruders would most likely choose to target to disrupt the TTEthernet-based clock synchronization protocol; although, the sensors & actuators layer have also been explored for potential entry points. This may include the design and implementation of the TTEthernet clock synchronization. Thus, it's imperative that this research digs deep and explores the potential vulnerabilities within the clock synchronization protocol and/or the use of TTEthernet for the fault-tolerant clock synchronization mechanism at the network layer of the IoT architecture.

It is confirmed in (Daniel and Roman, 2018) that clock synchronization PCFs have higher priority than the TT frames. However, there is no clear indication of how

contention resolution techniques affect the transmission of PCFs. Thus, it cannot be ruled out that the shuffling technique, for example, does not affect PCF communication. However, PCFs are naturally TT traffic as they are communicated on a regular time interval to synchronize drifting clocks; hence, they can be treated as TT traffic, although, they are considered a higher priority than other TT traffic (Daniel and Roman, 2018). Integration techniques are designed to help decide what happens if a higher priority frame arrives while a less priority frame is being transmitted, to avoid contention. Therefore, it is important to explore what potential vulnerabilities each contention resolution technique poses to the clock synchronization protocol.

In pre-emptive synchronization, an RC frame transmission is stopped to give way for a prescheduled TT frame and continues from where it was stopped to finish transmission after the TT frame finishes transmission. This is not the worst integration scheme from the clock synchronization point of view as the PCFs are transmitted as pre-scheduled time slots. However, there is a potential that RC frames will be dropped as the time duration between consecutive frames is too long due to the TT frame transmitted in between. In a Timely-block, on the other hand, RC frames get transmitted only if there is enough time before the next TT frame is scheduled to transmit. Similarly, this is an equally favourable integration scheme for clock synchronization as TT frames are transmitted according to prescheduled time slots. However, this does not mean it is immune to other forms of security breaches, including but not limited to a system disruption, hijacking, or DoS to initiate a man-in-middle type of attack; because these integration schemes can only protect against contention-related traffic dropout. Shuffling allows the RC frame transmission to finish leaving the TT frame to wait until the current transmission finishes. If this is allowed in a TTEthernet clock synchronization, it becomes easy for an intruder to delay sync frames by increasing the frequency of RC and BE frames. This would probably require a malicious intruder to access an end system to inject low-priority traffic, so the TT traffic gets delayed.

According to (Daniel and Roman, 2018), frame injection is one way a malicious attacker uses to disrupt the clock synchronization protocol. Three techniques of breaking the frame communication within a TTEthernet clock synchronization are discussed as

follows:

- **frame jamming** - where an end system fails to send or receive messages because it receives frequent unintended messages, or the transmission link fails to deliver messages.
- **Byte injection** - involves a corrupt, unintended, or untimely message transmission among network nodes.
- **Frame delay** - is a fault injection technique designed to delay the arrival of messages at the receiving end.

Narrowing the attack surface to the network layer for the TTEthernet clock synchronization, vulnerabilities within the integration techniques have been investigated. It's concluded that although every integration technique has pros and cons, pre-emptive and timely-block serve well for the TTEthernet clock synchronization as they guarantee complete priority for sync frames against other traffic types. Thus, vulnerabilities within these integration techniques are explored. Nonetheless, the likelihood of a system vulnerability being exploited or the level of threat it poses depends on the adversary's objectives and the resources they have available for him/her. Thus, the adversary goals are investigated in the next section.

3.1.4 Adversary Goals

The intention of an intruder decides which system vulnerability is exploited and the extent of the damage caused. In regards to clock synchronization, as seen in table 1.2, node insertion and node compromising are two of the major security breaches that take place at the perception layer. Node insertion is a phenomenon where a malicious intruder deliberately authenticates a new node to the network to perform their act and abuse the clock synchronization. Node compromising, on the other hand, refers to the ability of an attacker to compromise a legitimate network node with the intention of message forwarding to a different destination, dropping messages instead of forwarding them or even forwarding their messages by dropping the message they receive, among others. Similarly, message insertion, message deletion and message manipulation are the main security breaches that target the clock synchronization at the network layer of the IoT architecture. The Network Layer attack techniques are mainly employed by the MitM

type of attackers for varied reasons. An intruder manages to access the communication channel and does his/her bad acts by inserting their pre-concocted message, deleting benign messages intended to reach a destination or manipulating messages by forging them, delaying them, or directing them to a different route. These are some of the main security threats that malicious attackers can initiate to disrupt the clock synchronization protocol.

Security breaches are initiated for one of two reasons. It is either an unintentional security breach which includes human error and computer failure among others or a malicious attack where an individual or organized cyber criminals deliberately seek unauthorized access to a private network for financial gain or deny network services. Hence, it is the malicious security breach that is being investigated in this research to find ways of protection and/or mitigation techniques. Most often, the obvious targets of a security breach include financial gain and network disruption. Thus, it is important to identify the type of attacker in question to deduce their potential end goal. For example, while external attackers are usually money-driven, internal attackers could be disgruntled employees determined to disrupt network services so their employer is somehow defamed or hurt.

3.1.5 Attacker Types

The different types of security attackers must be distinguished before one can explore the reason behind every malicious attack. In general terms, there are two types of security attackers: internal and external (Mizrahi, 2014). Security breaches which stem from internal users can be unintentional from user error or computer failure; although they can also be malicious users working independently or as part of organized cyber-criminals for similar goals as external cyber-attacks which are mostly malicious. Internal attackers have physical access to their target network devices. On top of that, they have authorized access rights to certain network resources. These attacker types can take control of device behaviour or configuration to perform packet removal, packet delays, and traffic generation to execute DoS attack (Alghamdi and Schukat, 2021). These types of attackers can be individuals working on their own such as disgruntled employees or working for an

organized cyber-criminal group. Their goals can be as small as breaking a network device or as big as running malware for financial reasons.

External attackers usually are more skilled, well-equipped, and well-prepared to run the security attack. Their mission is usually financial gain or political motives. These types of attackers include **script kiddies** who are motivated mainly by curiosity and mostly not maliciously driven with minimal technical skills and resources available to them; **hacktivists** who are individuals or groups motivated to disrupt an operation or draw attention for ideological or political purposes and usually have moderate technical skills and hacking resources; **cybercriminals** who are groups of well-organized attackers with significant technical skills and advanced resources, motivated mostly by financial gain; **nation-state actors** who are the most sophisticated attacker types with the most advanced intelligence and other attack tools available to them, mostly motivated by political agendas; finally, there are **competitor** type attackers who are primarily motivated by business advantage and can have internal or externally hired attackers with moderate to advanced technical expertise.

Regardless of the attacker type, the emergence of AI tools has enabled attackers to use sophisticated AI tools freely and run AI-powered security attacks to initiate security breaches including the MitM and DDoS attacks. Therefore, understanding the attacker type gives an insight into their motivation and target of the attack which is helpful to devise a security solution. This also helps to set indestructible security objectives for why a certain system asset is secured or protected.

3.1.6 Security Objectives

The refining process in the threat model has gone through, identifying the main asset that needs protection, the potential malicious intruders that would consider attacking the identified system asset and their hypothetical goal of launching an attack. This ultimately leads to asking the all-important question which is why protect this asset? What is the security objective? General answers may include, getting a safe delivery of traffic or protecting the integrity of messages communicated through the availability of network devices for a reliable delivery or the confidentiality of messages being accessed

by the intended recipient only. It should be noted that different application scenarios have different security requirements. The integrity of messages being delivered to a specified destination unmodified, and the confidentiality of messages being accessed by the rightful user only are two of the main security objectives for military applications, for example. Similarly, the availability of network resources and the integrity of delivering intact messages, the same as they were sent, are two of the main security objectives for real-time health monitoring medical applications. In this section, some of the main security objectives around TTEthernet clock synchronization, as the main asset needing protection, are described briefly to give more clarity with regards to why securing this asset is important as well as what aspects of it need securing.

Availability: This refers to the necessity that all network devices including end systems and communication channels, are always available, so synchronization frames are communicated in a timely fashion. The security objective in this regard is to make sure adversaries don't compromise sensor nodes or communication links to deny the secure delivery of synchronization frames.

Integrity: This is another important security objective for the identified system asset. Synchronization frames should not be modified in transit or at rest. However, malicious intruders can always try to get access to the network through the physically accessible sensor nodes and then run a MitM-type attack. Similarly, AI-powered automated security attacks can target timing manipulation, predictive attacks or adaptive attacks to disrupt clock synchronization. Hence, the integrity of traffic communication must be protected for accurate feedback from the Control Centre.

Reliability: This is one of the most important considerations when it comes to the safe delivery of synchronization frames. It could mean that all resources required for the transmission of frames are available as well as that the frames communicated are delivered to the right recipient unscratched and not delayed. Hence, this is probably the combination of the most important security objectives for secure clock synchronization, discussed in this section, including the availability and integrity objectives.

Confidentiality: This relates to the fact that messages communicated should only be accessible to the right recipient. However, it is not high on the list of security objectives

for the identified system asset as there is no direct damage that comes from the mere fact that sync frames have been revealed but never tampered with or compromised in any way. Having said that, if the confidentiality of sync traffic is exposed, this may grow to a different type of attack, hence, it should also be considered for protection.

Authentication: This is more of a control measure rather than a security objective. However, they are closely related in that it is through control measures such as authentication that a security objective like the integrity of messages communicated can be achieved. Therefore, it is important that only users with the right access privilege or authenticated member nodes, in the case of clock synchronization, can take part in the synchronization frames communicated.

This section presented a broader perspective of identifying the important system assets in an organization that need securing; the vulnerabilities within these systems assets; and the how and why an attacker targets these system assets are explored in detail. Thus, it is helpful that some of the commonly known security attack types that could target the TTEthernet clock synchronization are investigated in detail and how this can hypothetically be resolved is explored further in the next section.

3.2 Attack Scenarios on the TTEthernet Clock Synchronization

A security attack can target an end system, a networking device, or a communication link. Advanced adversaries usually do their vulnerability analysis and target the weakest link in a network to then attack their target end system, information repository, or communication link. It's assumed, in this research, that an adversary manages to breach a communication link to access PCFs on the fly. This is a MitM type of attack which manages to place itself on the communication channel between two end systems or an end system and a network switch.

This section explores certain scenarios including how an adversary would manage to breach a communication link and designate him/herself for further attack as pointed out above and the kind of security attack that can follow from a compromised communication

link. These include investigating the MitM attack and hypothetical solution to MitM attack as well as delay attack on real-time applications and provisional solution to delay attack.

3.2.1 MitM Attack

In an ideal situation, data is communicated between end systems and servers seamlessly. Where an adversary places him/herself on the communication channel, however, data transfer can be manipulated to serve the adversary's objectives. Different techniques have been used to attack the communication channel in the form of a MitM attack (Conti et al., 2016). Some of the commonly used attack types that constitute the MitM attack are explored below.

Different attack techniques have been used to breach the communication channel in the form of a MitM attack. These include: DNS/IP Spoofing (Maksutov et al., 2017) - where users are tricked into sharing certain details with fake websites thinking the request has come from a genuine source; email hijacking where an email of an organization gets compromised and is used to contact the compromised organization's customers to surrender their financial and/or other important information thinking the email address is one they have communicated with before with no issue afterwards; SSL/TLS related attacks including session hijacking where an attacker hijacks an established session by gaining access to the session ID to gain access to the login details of users for the specific website the session was created for; SSL/TLS downgrade attack where a web server is tricked to establish a connection using an older SSL/TLS version which is deprecated due to its security vulnerabilities which is followed by manipulating the known vulnerabilities within the deprecated SSL/TLS version; SSL/TLS Stripping attack, Truncation attack and SSL/TLS vulnerability attacks are some more varieties of SSL focused security attacks. ARP (Address Resolution Protocol) poisoning or spoofing attack which involves an attacker getting access to the routing table of a LAN which contains the combination of IP addresses and their corresponding MAC addresses is another prominent form of attack which leads to a MitM attack. An attacker targeting the ARP would use the routing table knowledge to reply to a connection request by pretending to be the intended recipient

to get the MAC address registered in the LAN routing table for further attack. This is another one of the commonly used techniques to launch a MitM type of attack on the communication link.

AI-powered MitM attack is another form of a security attack which can target the communication channel and compromise traffic on the fly by stealthily intercepting and manipulating data exchanged between two parties. Machine learning tools can be used to predict and intercept the timing of messages, placing the attacker in the communication channel without detection. Such an attack can also employ AI algorithms to analyse and modify intercepted messages in real time to alter content, steal information, or disrupt communication. This compromises the integrity and confidentiality security objectives by gaining unauthorized access to sensitive information and altering messages. AI-powered security breaches can be more sophisticated and difficult to defend by mimicking normal communication patterns, (Shad et al., 2024). All the techniques mentioned above can be considered possible ploys to deploy in the network model outlined in this research; however, considering the provision of wired and wireless segments to the network, Wi-Fi-Eavesdropping is another potentially viable technique used to lead connections to a rogue server prepared by the malicious attackers. Wi-Fi- Eavesdropping is a technique commonly used where public Wi-Fi is used to connect customers or the public to a Wi-Fi hotspot. Sometimes referred to as 'Evil Twin', this method is used to fool users into thinking that they are connecting to the right hotspot although they have actually fallen into the hands of the malicious attacker who set up a Wi-Fi hotspot that resembles the actual hotspot. However, because the wireless device in this scenario is connected to a private network and reports from the sensor node are monitored by the control centre it is unlikely that Wi-Fi eavesdropping is going to cause a sustained network breach as in the network model outlined here; hence, it can be safely illuminated as a possible security attack on the communication link between the wireless end system and the WAP.

This subsection has highlighted the main security breaches which target the communication channel and outlined the ones most likely to be used in the communication channel for TTEthernet clock synchronization. It also reveals how the AI-powered security attacks make the MitM attack more complex. Therefore, this can be

used as a foundation to outline a hypothetical security solution as presented below.

3.2.2 Hypothetical Solution to a MitM Attack

The steps taken by an attacker to place themselves and control the communication link are out of the scope of this research as the focus here is mainly to investigate, analyze and defend against the subsequent security attacks targeted to disrupt the clock synchronization in a TTEthernet-based IIoT. Nonetheless, it is important that this is briefly explored here to give the general essence of how a MitM attack would take shape to establish the base for further attacks and how hypothetically this could be protected against.

It is implied that the intruder only needs to hold on to the synchronization frames for a period of time before forwarding them at a later time, in the nanoseconds. To underline the intention of the MitM attacker in this situation, it is not to crack open and manipulate the content of individual frames. Hence, any protection against this type of attack which does not need to open a frame communicated but drop or delay its delivery on the receiving end should involve stopping the intruder from positioning themselves in the communication channel. Thus, any security solution including frame encryption that defends against breaking the integrity and confidentiality of communicated frames to get an insight of or get access to information, change or delete content among others cannot be considered an option in this case. Therefore, the most viable protection against this type of attack should include the traditional security measures against ARP poisoning, SSL/TLS focused breaches, IP/DNS spoofing or other forms of security breaches to stop or at least make it difficult for an attacker to place him/herself on the communication link. Strong anomaly detection methods could be considered to continuously monitor and detect anomalous frames. AI or ML anomaly detection methods can be used for more adaptive and flexible monitoring of anomalous frames.

Therefore, it can be suggested that finding a deterring technique against those MitM attack types discussed above is a possible security solution. Nonetheless, message encryption may not be a viable solution as an intruder would not necessarily need to open the messages to drop or delay a synch frame. It is also suggested that strong anomaly

detection can be used to detect the presence of an anomalous sync frame if the malicious attacker starts to drop or delay a sync frame.

3.2.3 Delay Attack on Real-Time Applications

Delay attack refers to the breach of a security objective whereby certain traffic gets delayed or is delivered at the receiving end at a later time than the expected time frame. In any given network, different end systems collaborate to carry out a specific functionality. In a factory, for example, the integration of network devices at the plant floor as well as the management layer of the network infrastructure is required for consistent productivity, real-time operation, and cyber security among others. This, however, requires that all network devices share the same notion of time, in some applications more so than others.

The requirement of time synchronization among network devices varies for different applications. Real-time applications including industrial automation could be more delay-sensitive than the standard event-triggered applications where events are more important as traffic is generated because a certain threshold is reached, for example. This is where TTEthernet comes in to address different temporal necessities among different applications within the same network, namely event-triggered and TT traffic communication. Thus, although most networks require time synchronization, applications dependent on TT traffic are highly sensitive to delay attacks. Hence, delay attacks remain the main security threats to clock synchronization for real-time applications. If synchronization frames are delayed beyond the tolerated threshold, the mentioned seamless integration may be broken leading to potentially serious security breaches in the factory. The same can be said about a consistent delay which crosses the maximum threshold. Therefore, delay attack is one of the main security threats to the clock synchronization protocol within the IIoT.

The delay in synchronization frames can be equated to the time difference between the expected delivery time and the actual delivery time. As in the permanence function, every end system registers its own expected delivery time for every synchronization frame, also called the transparent clock. If a delay attack breaches the maximum delay threshold, every TT traffic gets dropped leaving the network device in question desynchronized

until a defence mechanism is deployed to isolate the desynchronized end system and rectify the problem. While the end system is desynchronized, however, event-triggered traffic, namely the RC and standard Ethernet traffic also called BE are communicated as normal. This exposes industrial applications to hazardous situations where critical sensor reports do not come in time to trigger control measures before bad accidents happen. This indicates the importance of having a correction mechanism as the severity of the effect of delayed delivery of PCFs or desynchronizing an end system can range from a simply delayed delivery of inconsequential data to fatal consequences risking the life of human beings.

Therefore, it is evident that different applications have different tolerance to latency-related security breaches. Applications dependent on real-time traffic are more sensitive to delayed frames as compared to applications not bothered by it. Thus, understanding the delay attack helps to outline a provisional security solution against these types of attacks.

3.2.4 Provisional Solution to Delay Attack

Different security solutions can be considered for delay attacks in a TTEthernet clock synchronization in the IIoT. Understanding the attacker type and motivation helps to devise a protection technique. some scenarios are designed in this subsection to understand the way a delay attack can be instigated and provisionally resolved.

It is important to start by exploring the effect of integration techniques, mainly the pre-emptive, timely-block and shuffling techniques, used in TTEthernet networks, as the first step to securing the communication channel. It is given that all three techniques have pros and cons associated with them. Pre-emptive refers to a technique where traffic with lesser priority gives way for TT traffic so it can be transmitted after the priority traffic finishes. The positive of this technique is that it is perfect for clock synchronization. However, RC traffic can be dropped because of a too-long wait until the TT traffic is completed. On the other hand, timely-block keeps less priority traffic wait if TT traffic is scheduled to use the communication channel. This is again a good fit for clock synchronization, but this technique tends to waste network resources while

RC traffic waits for TT traffic to arrive and finish transferring. Finally, shuffling is the technique where a first-come, first-serve type of communication is observed. This is not a good fit for clock synchronization as PCFs would have to wait for lesser-priority traffic to finish communicating before they can start using the communication channel. Therefore, pre-emptive and timely-bock make sure that PCFs have the highest priority and that there would not be a delay in sync frame communication if either of them is used as an integration technique.

An attacker may decide to completely drop synchronization frames leading to the breach of the maximum tolerable delay threshold. This may be an attacker with a short time intention who is not bothered about making a big splash hence getting picked up by network monitoring systems. In this case, an end system under such an attack gets desynchronized instantly as the maximum tolerable threshold is breached and all subsequent TT traffic is dropped leaving the end system unsynchronized until it rejoins the synchronization operation again. One of the solutions to consider is to automate a trigger whereby an alert message is flagged once the maximum tolerable threshold is breached. Existing solutions would have to be considered to address this type of attack.

Lastly, a calculated delay attack can be instigated by an attacker whereby the maximum tolerable threshold does not get breached. This probably is an attacker who has a long-term interest and intends to stay unnoticed by network monitoring tools. Attackers involved in this type of attack also are most likely to have a good understanding of the network infrastructure and what the maximum tolerable threshold is. So, a protection mechanism to defend against this attack is a bit more difficult than those that breach the maximum latency threshold because the normal monitoring tools are unable to detect this attack as it is designed not to cross the maximum latency threshold. Therefore, the most viable solution at this stage is to identify consistently delayed frames as outliers and devise a resolution specific to those frames. A commonly used security solution against latency in synchronization frames involves anomaly detection techniques by establishing baseline latency patterns and continuously monitoring deviations. Anomaly detection methods can detect abnormal latency in synchronization frames, potentially caused by AI-powered attacks, using machine learning or statistical analysis by cross-referencing

with other metrics and adapting over time or simple rule-based methods to nullify PCFs breaching a set latency threshold. These systems can detect and flag delayed sync frames, ensuring accurate and reliable time synchronization.

It should be noted that these are just some of the attack scenarios that can take place in the form of a delay attack in TTEthernet-based IIoT; thus, mitigation techniques would have to be pertinent to the specific attack type. These are just two of the main security attacks explored to outline the main consideration for secure TTEthernet-based clock synchronization in this research. Thus, anomaly detection among others is hypothetically a fitting solution by flagging abnormal sync frames by setting different layers of latency thresholds.

Chapter 4

Research Formulation

4.1 Aims & Objectives

The ultimate goal of this research project is to provide security guarantees in the clock synchronization for interconnected safety-critical IIoT systems using TTEthernet.

We consider safety-critical real-time industrial applications using DE (Deterministic Ethernet) which is currently being standardized and is in use in several real-time application areas such as industrial automation, energy systems, aerospace systems, buildings and vehicles connected aimed at converging all applications on a single DE backbone network.

The main objectives identified are designed to build up to the main aim of this research project. Thus, they are steps required to achieve the grand aim rather than independent research topics. Below are the main objectives described briefly, to give context to the research.

1. Investigating the TTEthernet-Based IIoT Applications, their Processes and Usage:

TTEthernet is an emerging technology used by the industrial sector, among others, to solve certain communication issues that the standard Ethernet connection can't resolve. It has two main advantages over other contender solutions recommended over time. Firstly, it supports mixed criticality traffic communication where real-time traffic gets the highest priority followed by the less time-sensitive traffic and the normal Ethernet traffic

over the same network backbone. Secondly, clock synchronization is fault tolerant as it supports multiple synchronization and CMs. Hence, investigating the TTEthernet-based IIoT applications, their processes and usage is the first step in understanding the subject matter in a broader sense. Achieving this objective provides a general context as to why achieving the main aim of this research is important.

2. Exploring the Significance of Clock Synchronization and its Vulnerabilities in TTEthernet-Based IIoT Systems:

This objective focuses on the pivotal role of clock synchronization in ensuring secure communication within real-time industrial networks. The integrity of the clock synchronization protocol is paramount in environments where synchronized network devices are crucial for executing specific operations. An adversary targeting this protocol could significantly disrupt organizational processes including but not limited to a company's production capabilities. Therefore, this approach aims to underscore the critical importance of robust clock synchronization in TTEthernet-based IIoT, evaluating its vulnerabilities and security implications. TTEthernet is studied relative to alternative clock synchronization protocols to justify the selection of TTEthernet for clock synchronization in this research. This objective tries to justify the importance of securing the clock synchronization protocol in a TTEthernet-based IIoT; hence, the main research aim.

3. Developing a Comprehensive Security Framework for Clock Synchronization in TTEthernet-Based IIoT:

This objective focuses on the design of a security solution to protect the clock synchronization protocol in TTEthernet-based IIoT. This objective is designed to develop a novel algorithm to achieve the main aim of the research. It involves the use of Python script to write the main phases of the algorithm which includes setting up two layers of maximum latency thresholds to nullify anomalous sync frames including those intentionally delayed to stay under the globally understood maximum latency threshold. The algorithm is set to work for all IIoT deployments where wired and wireless traffic is communicated. Thus, this objective is designed to outline the technicality of the main aim of the research project.

4. Testing and Evaluating the Proposed Security Solution

This pertains to evaluating the proposed security solution using different network models that mimic practical network scenarios. These network models represent different sizes and complexity networks for wired and wireless networks, designed to test if the security solution can stand in different network environments. A simulation tool, VisualSim, is used to test these network models as a way to justify the validity of the security solution offered. A Jython script is designed so the algorithm written in Python can be run on the simulation platform. The simulation platform only integrates with Java or Jython which is an implementation of the Python programming language designed to run on the Java platform. A fault injector is used to examine how the algorithm reacts to different levels of security breaches. The insights obtained from the validation process contribute valuable knowledge to the research whether it is proven valid under the different deployment scenarios or fails to do so.

4.2 Research Design

4.2.1 Type of Research

Researchers have used different research methods for different subject matters. Two commonly referred to methods are the quantitative and qualitative research approaches. Quantitative refers to a research method based on quantity or numbers and statistics while the qualitative approach explores the whys and how of a subject matter. As with most modern research, this research project is based on the combination of both approaches. Mostly, it's based on the qualitative approach as it addresses the research aims and objectives; although, it also involves lab tests which quantify the number and frequency of sync frames exchanged between nodes within the same IIoT network. Developing a monitoring and protection security algorithm as well as evaluating and testing such an algorithm are two of the main objectives of this research project. The first which involves designing a threat model with regard to why, how and when an attacker can potentially strike his/her acts, relates to the qualitative approach; while the latter relates to the quantitative approach as it works out the practical calculation of synchronization

frames and frequency of such frames taken within a certain time frame to understand if a certain synchronization frame is delayed beyond the set maximum latency threshold. Hence, this research uses a mix of qualitative and quantitative approaches.

On a similar note, research methods can also be classified as inductive and deductive approaches. The deductive approach starts from the general statement or hypothesis to the very specific concept through observation and tests; whereas inductive reasoning follows the opposite logic in that it starts from observation and moves up to the generalization of theories.

The deduction approach follows theory→hypothesis→observation→confirmation.

The induction approach follows observation → hypothesis → theory.

In summary, the deductive approach tests a theory to confirm or reject the hypothesis while the inductive approach generates a theory from an observation. Hence, this research has used both approaches. While the project development generally takes a deduction approach trying to prove TTEthernet clock synchronization can be protected using anomaly detection techniques; the design of network models to mimic different deployment environments and the use of a simulation tool to prove the security solution offered is capable of nullifying sync frames breaching the set latency thresholds proves an induction approach.

4.2.2 Benchmark Resources

Research on the security of clock synchronization focused on the delay attack, (Lisova, 2018) is a good foundation for this research. The synchronization protocol employed in their research is the PTP of the IEEE 1588 standard. Hence, research focusing on the TTEthernet-based clock synchronization rather than the PTP protocol is quite a timely topic to be explored. They used game theory to identify and flag anomalous sync frames; while the fault injector technique is used to understand and detect security breaches in the network models designed for this research. Thus, this research is considered a benchmark in some ways as their final aim is to secure the clock synchronization protocol in an IIoT which is similar to our research aim.

Similarly, as pointed out above, the fault injection technique used in (Daniel and

Roman, 2018) is used to detect sync frames breaching the set maximum latency thresholds. They designed a physical fault injector that pushes abnormal frames in the communication channel to see if their detection system can identify the abnormality in a single-hop traffic communication. The same technique which integrates with the modelling and simulation tool is employed in this research. So, this is another benchmark resource used to apply fault injection to mimic real security breaches targeting the clock synchronization protocol.

4.2.3 Research Strategy

This is an important section as it outlines the general strategy used to complete the research and presents the main tools and techniques used at every stage of the research. It is conducted with a clear overall strategy to conduct the research project. An outline of the research strategy is presented below.

• This research starts by identifying the knowledge gap by investigating different viewpoints in the literature review and exploring the main security threats in a TTEthernet clock synchronization as well as potential ways of addressing this gap in knowledge as in the vulnerabilities before they turn into a security risk, in the following Chapter.

• An exploratory case study was conducted to understand potential security threats and hypothetical resolutions to them.

• This is followed by configuring a modelling and simulation tool which is used to design three different network models to mimic different sizes, topologies and complexities of network deployments. They include communications on the wired and wireless channels as the target network deployment is an IIoT.

• A fault injector is designed to push anomalous sync frames that evade the set latency thresholds which is used to test the validity of a rule-based anomaly detection system.

• The main algorithm that offers the main security solution is written in a Python script which is then translated into Jython to be able to integrate with Visualsim, the simulation tool. The algorithm is designed to have two layers of latency thresholds to catch sync frames that breach the set maximum latency threshold and those that don't correct themselves after a certain deviation although they don't breach the maximum latency

threshold. The rule-based anomaly detection system is designed to work on Visualsim; although, it is configured with the practical network deployment in mind so it works in a real TTEthernet setup with minimal modification. The system continuously monitors the PCF communication and flags if a sync frame is found to have breached one of the global or local maximum latency thresholds. In a simulation environment, a fault injector is used to push faults or anomalous sync frames which the anomaly detection system flags depending on which latency threshold it breaches. It is flagged and added to a flagged list if a PCF breaches the global maximum threshold. If it breaches the local maximum latency threshold, however, it is added to a list which is monitored for a period of 10 synchronization cycles before it is released, if it corrects itself or is added to the flagged list if it does not correct its deviation within the 10 synchronization cycles.

• Simulation results are discussed and the concluding remarks are presented by clearly stating how each objective is met.

The research strategy outlined above establishes a solid foundation for addressing the core research aims and objectives of this study. It is outlined to capture the depth and breadth of the subject matter paving a clear understanding of the key issues. The configuration of the network models and fault injectors is a strategy designed to prove the validity of the main algorithm. This, in turn, is designed to provide the all-important resolutions to the raised research objectives. It is carefully designed to be aligned with the research objectives leading to detailed and insightful findings.

4.3 Data Collection & Analysis

Initial thoughts were that data would be collected from and analysed using a physical TTEthernet setup at Cranfield University which was the only TTEthernet setup in the UK, at the time. Nonetheless, it was found to be malfunctioning; hence, it could not be used for this research. Thus, Visualsim is designed and configured as a modelling and simulation tool to collect and analyse mainly the quantitative data. An exploratory analysis is carried out to collect and analyse the qualitative data. CoRE4INET, which is an extension to the INET framework on the OMNET++ simulation software was considered as an alternative to Visualsim, but it's not as well maintained and there is no support. Furthermore, the

library blocks for CORE4INET are not updated as frequently as in Visualsim. Thus, Visualsim is used as the simulation platform for this research to collect and analyse data.

4.3.1 Data Collection

• **Qualitative Data Collection:** In the exploratory phase of the research, data is collected from the comprehensive literature review and use-case scenarios which includes academic articles published in well-established research journals and case studies that explore varieties of security threats within the TTEthernet clock synchronization and existing security solutions. These resources are critically assessed to identify the most suitable approach to determine the dimensions taken to the research. These data helped to establish a more suitable approach to the core algorithm as well as the configuration of the network models.

• **Quantitative data collection:** refers to the data collected through simulation of the different network models. The simulations measure different metrics including the fault injected, latency of sync frames, and flagged anomalies among others. This includes statistical data representing PCFs communicated in wired and wireless environments, in small and more complex network deployments, where fault injector is applied and where it is not applied. Hence, data collected through the simulation is quantitative data

4.3.2 Data Analysis

• **Qualitative Data Analysis:** The qualitative data analysis involves thematic analysis where data is identified and categorized depending on common themes and relevance of content to the main objective being explored. The literature review is divided into three sections by analysing studies related to a specific dimension of the research or specific objectives identified in this research. Closely related studies are grouped into tables to help understand the knowledge gap. Similarly, a threat model is explored to identify potential security threats and provisional solutions to those threats. Comparing and contrasting is another form of thematic analysis for qualitative data. In this research, wired and wireless mediums of communication are compared to help identify the right approach for the modelling and simulation. Thus, thematic analysis is mainly used for the

qualitative data in this research.

• **Quantitative Data Analysis:** This is more of a structured and measurable analysis using the simulations of different network models that are used to validate the security solution that is presented in the form of the core algorithm. The quantitative data collected from the simulations is analysed statistically to validate the chosen approach. The comparison of results from simulations carried out on the network models, with different levels of faults injected is a statistical analysis. Statistical methods, such as comparative analysis, are used to interpret the data and draw conclusions.

4.4 Research Population

This study focuses on Industrial applications as they mostly require real-time data transmission. Although TT traffic gets absolute priority, event-triggered traffic also gets communicated between sensors and controllers; hence, the need for TTEthernet. Therefore, all IIoT deployments which support mixed-criticality data transmission requirements, where real-time traffic gets priority, can benefit from this research. Beneficiaries of this research work include Aerospace, Industrial automation, railway, energy and automotive applications, among others.

4.5 Mechanism to Assure the Quality of the Study

This is a research project started out of the sheer interest of the researcher. Furthermore, the researcher has no allegiance to any organization, business-related or otherwise, which would have an interest in the outcome of the research work. So, it is free from biases and the quality of the study is unaffected by other influencing factors. Moreover, other important vigorous safeguards have been considered to this effect.

The research employs a methodologically rigorous framework that includes a comprehensive review of the literature to counteract any potential bias coming from the theoretical grounding of the project. The literature incorporates multiple perspectives to address and minimize perspective bias. This approach ensures that the research doesn't favour one viewpoint over others. The statistical data and analytical techniques utilized

are documented in detail. This fosters transparency but also allows other researchers to replicate the study; thus verifying the findings independently. Moreover, feedback is sought from the supervisors at every stage of the research to identify and mitigate any unconscious biases and to challenge any research assumptions and conclusions among other supervision duties.

The study aims to maintain the highest standards of academic integrity and freedom from bias by implementing the above strategies. The quality and validity of the research findings are upheld by these robust safeguards, ensuring that conclusions drawn are the result of rigorous research work.

4.6 Resources Required for the Study

The main resources used for this research work are mainly the use of the internet on a home laptop and the precious time the researcher spends doing the research. The digital publications, which are freely available from the University of East London's digital library as well as some hard books relevant to the research subject are the main sources for the literature section of the research. Expenses were incurred travelling to and from Cranfield University twice.

Visualsim has a free license for researchers although it is a proprietary software used internationally by the aerospace and industrial automation sectors among others. Furthermore, attending some relevant conferences and making presentations did not require travel and material financial investment.

Chapter 5

Environmental Setup

5.1 Network Modeling and Simulation

Having discussed the security threat TTEthernet clock synchronization is exposed to, it is imperative that a use case scenario is developed to explore the project implementation in more practical terms. It has been implied that the main attack surface for a TTEthernet-based clock synchronization is the physical layer and/or the network layer. This can be explained in terms of the end systems and the communication link respectively. An attacker would have to gain access to an end system or the medium of communication between two end systems or an end system and a network switch to commit his/her adversary's goals. In an industrial setting, for example, sensor nodes are easily accessible and can be compromised by malicious attackers to communicate their own prefabricated frames to the receiving end system and cause the damage they intend to do. This section covers the modelling and simulation software used for this project as well as the different configurations applied to test different facets of the research including the effect of wired and wireless channels as well as the different levels of security breaches by delaying the synchronization frames. Fig 5.1, shows how the model is set up to help mimic a real-life network deployment in typical industrial automation. It is the simulation version of the diagram presented in Fig 5.5 under Chapter 5.4, below.

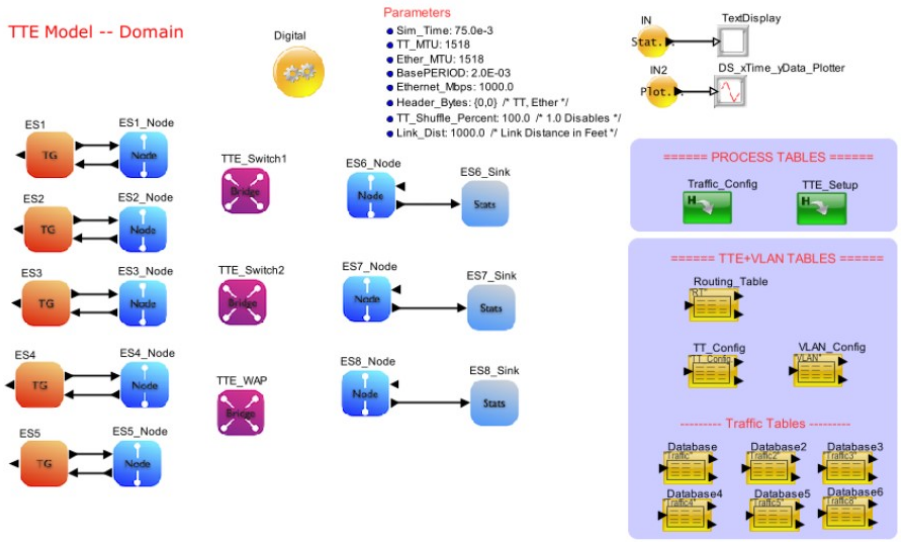

Figure 5.1: shows network model taken to represent a typical real-life IIoT deployment

5.2 End System Configuration for the Model Implementation

TTEthernet Clock synchronization is carried out using three different synchronization performance parties. These include SMs, CMs and SCs. Five of the end systems in the network model are configured to be SMs but the remaining three, including the wireless device and the WAP are not configured for additional synchronization performance; hence, they remain as SCs. It's important to note that most sensor nodes are usually less equipped for advanced configurations to conserve energy among other things. Thus, the sensor node in this case has not been configured for synchronization performance. A networking device that is not configured for synchronization performance is considered an SC; because it has no role in the clock synchronization process other than adjusting its local clock according to the sync frames transmitted to it. Both network switches are configured to be CMs, to be able to aggregate synchronization frames received from the SMs and do their own calculation before they broadcast to all member nodes both the SMs as well as SCs.

A WAP is there mainly to link up the wireless devices to the local area network; hence, it is helpful to configure it as a CM to be able to broadcast synchronization frames to all wireless devices connected to it. In this case, however, it is linked to only one

wireless device and the network switches which are also the CMs in the current model. Hence, there are no SMs linked to it which would feed the initial reference clock to start the synchronization performance. For that matter, the WAP could not be configured as a CM. In essence, the WAP is acting as a transit for traffic between the network switches and the wireless device. Thus, it has assumed an important role as a bridge in multi-hop communication for all traffic including the PCFs to and from the wireless device and the network switches. In most practical TTEthernet clock synchronization scenarios, however, the effect of multi-hop communication is less important as network switches and WAPs are configured as CMs to feed directly connected network devices with synchronization frames.

The permanence Function (Steiner et al., 2009) considers that most network devices have links to more than one network switch, important for redundancy which is a typical characteristic of TTEthernet. However, in practical network deployment, certain network devices may be placed where they cannot easily be connected to more than one switch; thus, there would be no redundancy options in those situations; although by principle, the design of TTEthernet offers options for redundancy. For this project, a fully redundant wired LAN is considered to showcase the benefits of TTEthernet clock synchronization. On the other hand, there is only one WAP; hence, wireless devices cannot be linked to a second WAP, leaving them with a single point of failure. That is, if the WAP fails, all directly connected wireless devices fail to communicate to the wired LAN.

The implementation includes a fault injector that can be applied anywhere in the network. Injected faults represent different forms of security breaches in a practical network including but not limited to frame jamming where an end system or the communication link becomes unresponsive or fails to respond because of DDoS-like attacks; Byte injection where an untimely or unintended message is transmitted among network end systems; and frame delay where the arrival of messages at the receiving end is delayed, (Daniel and Roman, 2018). Frame delay is the approach taken in this research.

For ease of analysis, it is assumed that an adversary manages to place him/herself between the WAP and the wireless device and targets the clock synchronization frames going to the wireless device to then follow it up with more serious attacks on the traffic

communication between the wireless device and the control centre. The target wireless device could be designed to report certain temperature or pressure levels to a control centre on a prescheduled fixed time frame, consistently. Similarly, such a wireless device could also be configured to send an alert message when a certain minimum or maximum level is breached. If the sensor device is out of synchronization or is showing a slightly delayed local clock, then the sensor reports would arrive at a delayed time in the control system according to its local clock. An adversary may decide to hijack an end system and control the flow of synchronization frames or gain access to the communication link to perform selective dropping of synchronization frames or make sure they arrive at a slightly delayed time frame. One way of doing this involves holding synchronization frames for a certain amount of time and re-sending them at a slightly delayed time or they can also drop these frames so that the clock on the wireless device gets drifting continuously affecting functionality gradually.

The fault injector in this case represents a malicious attacker who has gained access to the communication link and assumes two different characters where he/she just keeps hold of synchronization frames for a short time, enough to cause the local clock on the sensor device to slightly drift but stay within the maximum tolerable delay threshold; or completely drops PCFs or delays them enough to breach the maximum tolerable delay threshold. It can be argued that if the security breach is within the maximum delay threshold, then it can be tolerated. The maximum threshold is drawn for a reason, but it is drawn with the assumption that whatever drift there is the next synchronization frame would bring the delay closer to the ideal time. The argument in this scenario, therefore, is that the adversary keeps the delay closer to the maximum delay threshold consistently, gradually degrading the functionality of the wireless device and the reports it sends to the control centre. This way, the attack remains undetected by monitoring techniques and still affects the role of the wireless device.

Both network switches are equipped with 1000 Mbps communication speed. Hence, all network devices can only communicate within this speed limit. Furthermore, the link distance between end systems and switches is configured to be within 1000 feet. Hence, an end system placed beyond the 1000 feet limit may lose its network connection.

This section has outlined the network modelling structure. It also explained the reasons behind the decision to design the model the way it is. A TTEthernet Clock synchronization has many building blocks including the SMs, CMs, and SCs. The sole wireless device configured in this network model can easily be multiplied in a different network model or an actual network deployment. Similarly, the WAP could have been configured for a CM in a different network model that has multiple wireless devices or in a practical network deployment. Hence, the configurations outlined here apply specifically to this network model and for the reasons explained above. The modelling and simulation tool required to implement the configurations discussed above is presented next with more details of how the configuration blocks are set up.

5.3 Modeling & Simulation Software

Visualsim is commonly adopted by the avionics and automotive industry among others. It is used for modelling & simulation, exploration, and collaboration platforms for all network connections and communications types. This section presents the details of the configuration setup used to build the network model.

The developers' team were able to add certain features and modify the existing back-end script to include certain features including a fault injector important for the purpose identified in this project as well as alternative options for the shuffling integration technique that has been used as the only option used as a parameter to avoid traffic congestion; namely the timely-block and pre-emption. Furthermore, the language used at the back end is a scripting language easily available to fiddle and modify configurations to fit one's implementation. Moreover, a rich library of ready-made blocks and parameters is usable for different scenarios. Hence, using existing blocks and parameters one can build a model and simulate by modifying the routing table and traffic tables. The fault injector is designed so that it can be applied anywhere in the network to be able to analyze the effect of the injected fault and mimic real-life deployments in the presence of a malicious attacker. Moreover, options are created so that the level of fault injected can be selected to show the severity of the attack including breaching the maximum delay threshold and staying within the tolerable threshold, as shown below in Fig 5.2. The different levels of

faults injected maliciously by an attacker can be presented as follows.

Injected Fault – **none** - indicates that everything else being constant, no fault is injected or that no malicious attacker is present. Hence, synchronization frames are communicated at a normal speed and accuracy that End systems would need to travel from point A to point B.

Injected Fault – **'Sync Drift Below Threshold'** - implies that a malicious attacker exists, and that fault has been injected but it remains within the maximum tolerable latency threshold. Sync frame delay does not cross the maximum tolerable delay threshold.

Injected Fault – **'Sync Drift Spread'** – In this instance, similar to the 'Sync Drift Below Threshold' explained above, a malicious attacker exists. This is a situation where the fault injected breaches the maximum tolerable delay threshold. This is normally where a security breach is alerted by the security control setup to trigger a security solution.

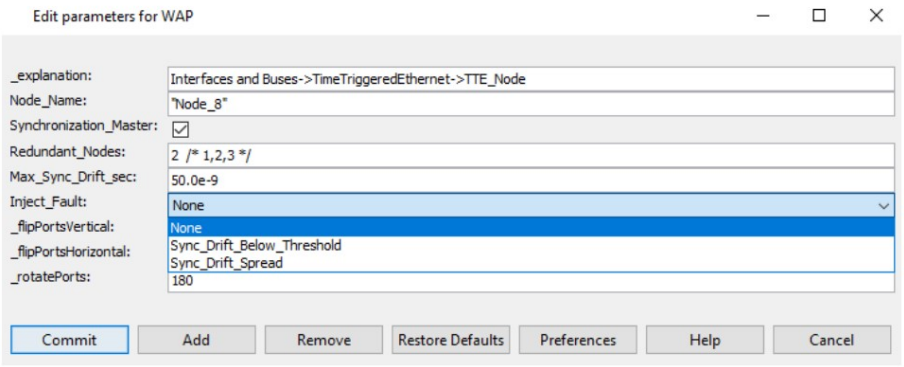

Figure 5.2: Fault Injector parameters displayed for selection for the WAP

TTEthernet is known for its redundancy properties; hence, it is usually referred to as fault tolerant. In this modelling and simulation software, on top of the fact that there are multiple SMs and CMs, every end system has the potential to connect to up to three network switches. Fig 5.3 depicts ES1 with two redundant virtual nodes as there are two switches in the model it can connect to.

The same picture can also be used to show the value set for the maximum tolerable delay threshold to be 50ns (nanosecond). This is usually set between 40-50ns for most TTEthernet setups

Fig 5.4 presents the visual display of the redundant virtual nodes for end systems in visualsim.

Every single End System in the model has similar internal building blocks to enable

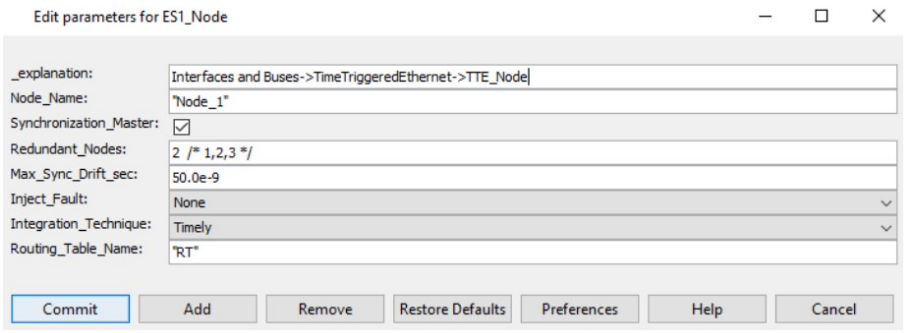

Figure 5.3: Configuration for redundant virtual nodes for ES1 and maximum drift threshold value

them to link to multiple network switches for redundancy purposes. If the connection between 'Node A' and another network device fails, the other two connections can be used to transfer the same traffic. Ultimately, the transaction that gets delivered first is taken and the same frame coming from the other connections gets dropped.

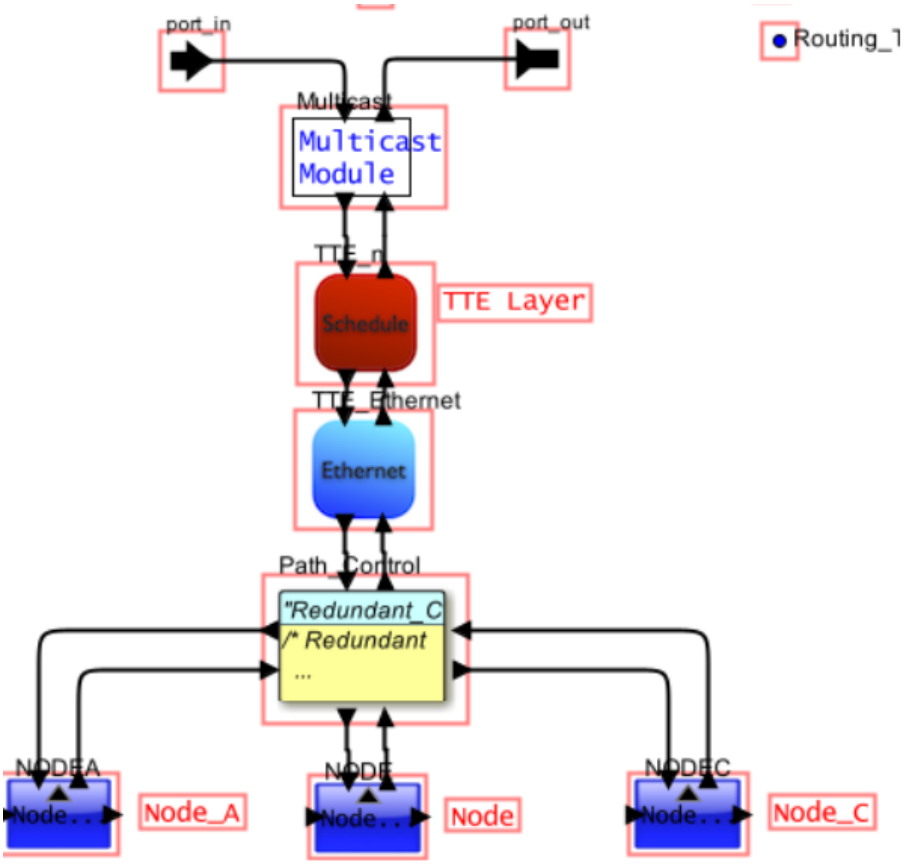

Figure 5.4: Displays redundant virtual nodes of an End System in Visualsim

Hence, Visualsim can be configured to imitate the actual TTEthernet network deployments in a given IIoT. Different network deployments have different resources and configuration requirements. Thus, the adaptability of Visualsim to fit any network

scenario is important. This is further proven, in the next section, where network models can be designed for various network topologies.

5.4 Network Topology

In the model considered for simulation in this research, eight end systems of which one is a wireless device, mainly a sensor node, are connected as clusters of star topologies where two switches and a WAP are interconnected. The wireless sensor node is connected to the LAN through the WAP. All end systems except the wireless end system are connected to both switches for redundancy reasons. Thus, traffic communication has at least two route options to get to its destination. Therefore, it can be said that there is a hybrid of star topologies where all wired end systems are connected to a central point of communication, the switches, and the wireless device connecting to the WAP in the form of a simplified star topology. On the other hand, the two switches and the WAP are interconnected making different routes for redundancy, creating a mesh topology where the three devices are connected to each other.

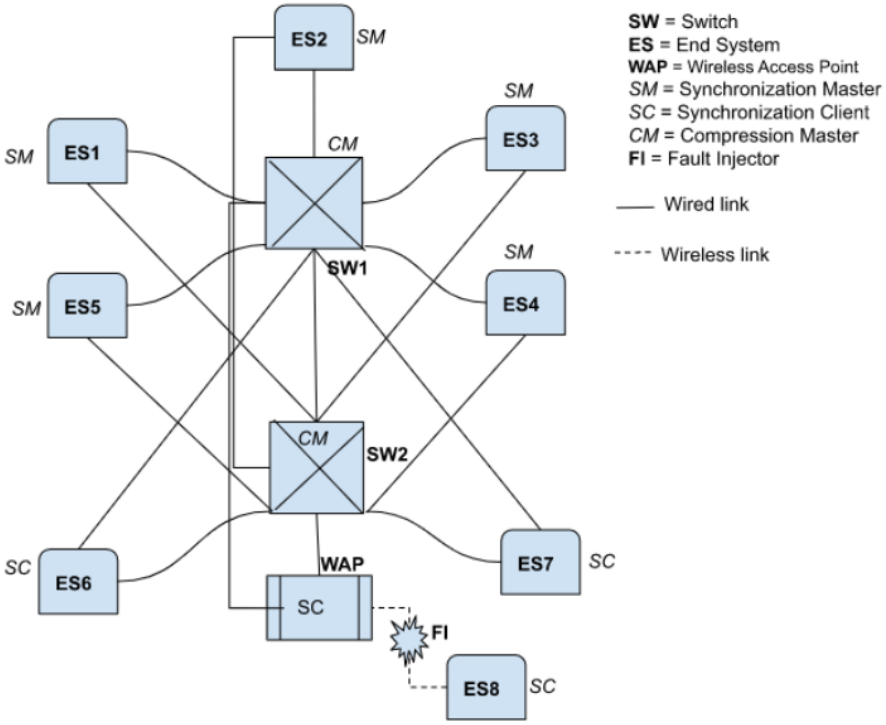

Figure 5.5: Diagram of the network model showing the interconnection of end systems

Each network switch is directly connected to nine network devices including seven

member nodes, another network switch as well as a WAP, as shown in the picture above. Hence, it can be said that there are two clusters of star topologies connected to each other. Furthermore, there is a potential third-star topology where all wireless devices could be connected to the WAP and make a cluster of their own. For this research, however, only one wireless device is considered to complete the Internet of Things side of the research topic and help with data analysis for synchronization frames communicated from the CMs to the SC on the wireless segment of the network model. It should be noted that there is a single point of failure in this segment as the wireless end system is connected to only one WAP which is in turn connected to two network switches. This supports the argument that TTEthernet is primarily designed for a wired connection as, in practical terms, it is not easy to set up a redundant WAP as most wireless devices usually have a single inbuilt network card; hence, they can only connect and communicate with one WAP. So, the actual topology in this case is a one-to-one communication from the WAP to the wireless sensor device and vice versa. Therefore, two clusters of star topologies with a third cluster linking a single wireless sensor to the LAN, are employed to help mimic an actual network deployment. It is important to underline that these topologies are designed to assimilate real-world-like scenarios where the delivery of clock synchronization frames at every member node, in a TTEthernet-based IIoT is affected by an intrusion of a malicious attacker. Fig 5.6 is presented to show a model of 16 port TTEthernet switch setup for this research.

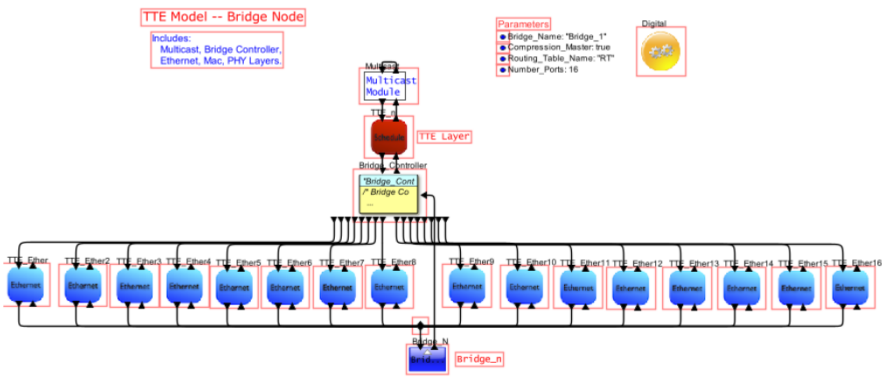

Figure 5.6: TTEthernet switch internal blocks and parameters setup

This chapter has presented the environmental setup required to design the simulation platform necessary to mimic an actual IIoT deployment. It has also shown the detailed

configurations of building blocks to configure the required parameters to set up the required network models. Finally, network models are also designed to have various network topologies. Thus, the required foundation is laid in this chapter important for the simulations and results analysis in the next chapter.

Chapter 6

Discussing Results of Modelling &Simulation

This Chapter presents the modelling and simulation software used in more detail which is followed by the results disclosed after different levels of security breaches are injected in the form of a fault injector. It's shown that security can be breached by violating the maximum tolerable threshold set to trigger a security solution. Similarly, it can be a continuous divergence of sync frames away from the normally accepted delay caused by other factors than a security breach. This can be a skewness closer to the edge of the tolerable latency but never crosses the maximum tolerable threshold; hence, it does not trigger the alarm for a security solution. Such synchronization frames can be seen as outliers but never correct their behaviour because they could well be intentionally designed to stay at the edge. This, in principle, can be argued that if it is tolerable and a security alarm is not triggered for security solution then it should not be considered a security breach. Nonetheless, it should also be understood that outliers are not expected to be outliers for good. Thus, the so-called tolerable behaviour might have been exploited in this case to continually drag or delay PCFs and degrade the reliability and timely delivery of the follow-up traffic. The outcome of sync frames communicated between the network switches and End Systems 6, 7 & 8 are presented to help compare the effect different levels of security breaches can have and how such an effect varies in a wired and wireless medium of communication. A fault injector is applied on certain end systems or the link between end systems and network switches to achieve the results mentioned

above. A comparison is presented between the wired and wireless channels by configuring ES8 as a wireless device receiving wireless communication from the TTE_WAP while ES6 and ES7 are configured as wired devices requiring wired connections to and from TTE_Switch1 & TTE_Switch2. This difference is set up by changing the value in the 'propagation constant' as shown in Fig 6.1.

The different values input for the propagation constant reflect the speed at which traffic travels in different mediums of communication. This is explained in detail in Chapter 7 above.

This section, therefore, presents results for the before and after fault is injected and when different levels of fault are injected. This in turn opens a new spectrum of options for how security breaches, to synchronization frames, can be protected.

6.1 A Comparative Analysis of Integration Techniques for Secure TTEthernet Clock Synchronization

Integration technique is an important aspect in the communication of clock synchronization frames. Different traffic types have different requirements for integration techniques. TTEthernet networks require real-time traffic communication. Hence, TTEthernet clock synchronization benefits from the integration technique which prioritizes PCFs over other traffic types.

As discussed in Chapter 5 above, the three contention resolution techniques offered in TTEthernet, mainly: Timely-block, Preemption and Shuffling are considered, and their pros and cons are analyzed. Nonetheless, Visualsim is designed with the shuffling integration technique which basically is based on a first come first serve basis. This is mainly because clock synchronization was not the focus when it was premeditated at the beginning. Thus, a challenge is taken to work with the developers' team at Mirabilis, the company responsible for Visualsim, to include options for Timely-block and the Preemption resolution techniques as alternatives to shuffling. Timely-block and Preemption are equally good enough for the TTEthernet-based clock synchronization as they give PCFs ultimate priority over all other forms of traffic. However, it is important to

note that the decision to select the integration technique used here does not consider the effect it would have on the follow-up network traffic because this project is only focused on the clock synchronization protocol.

Thus, the choice of an integration technique depends on the type of traffic being monitored. Timely, a short form of Timely-block is used as the main integration technique in all scenarios used in the modelling and simulation software in this project, as shown in Fig 6.1. Hence, as suggested in the literature review section, sync frames are made to access the communication channel as they arrive as the chosen integration technique offers them the utmost priority.

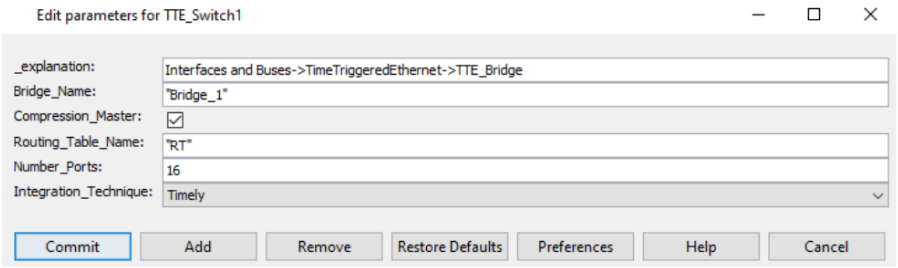

Figure 6.1: showing Timely-block being used as the integration technique

6.2 Latency in TTEthernet Clock Synchronization

Clock synchronization frames among other types of traffic travel at different speed levels depending on different factors. The speed of the communication link, frame size, and the distance between source and destination devices, among others, affect the latency reflected.

- **Speed of communication link** refers to the medium of communication. Traffic travelling on a Fiber Optic cable, twisted copper cable or wireless medium is delivered at different times. Traffic travels faster through Fiber Optic cables than twisted copper cables, however, it is much faster in a wireless channel, sometimes considered as the vacuum. This is reflected in Chapter 7 above in more detail.

- **Frame size** is another factor that affects the speed at which traffic travels between communicating end systems. It normally takes more time to communicate big sizes of frames as compared to smaller ones. The frame size used in the network model is 64 bytes

as shown in Fig 6.2. It can be modified to affect the speed at which traffic is delivered.

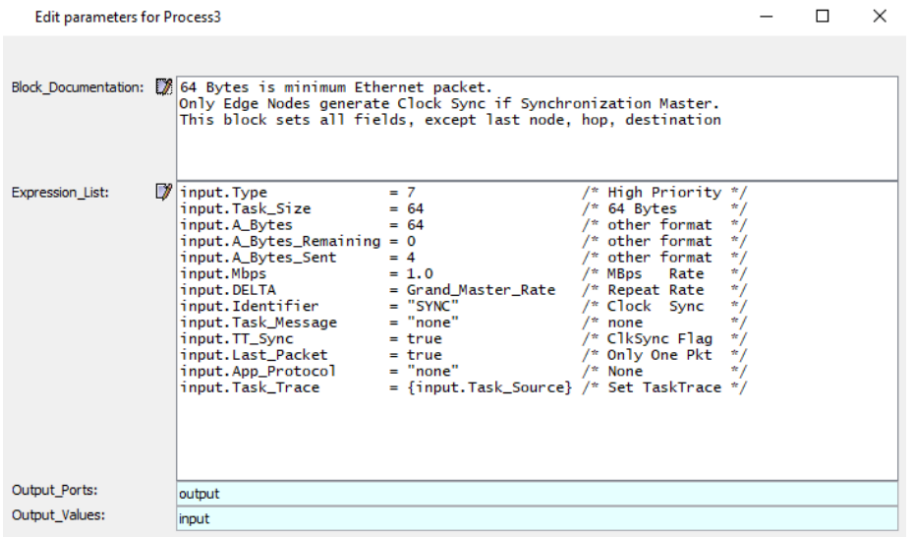

Figure 6.2: shows frame size communicated through the network

Similarly, the maximum frame size as indicated in the parameters shown below is set to 1518 bytes which is the biggest accepted Ethernet frame communicated.

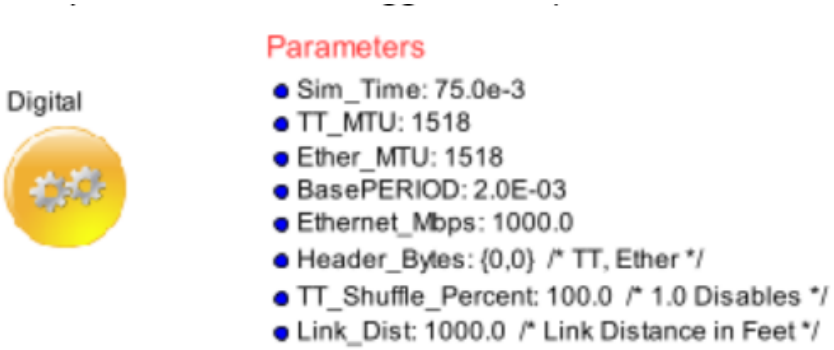

Figure 6.3: parameter set MTU (Maximum Transmission Unit) to 1518 bytes

- **Distance between source and destination nodes** is another factor that affects how fast a sync frame is delivered at the destination end. Sync frames are delivered faster where communicating end systems are closer to each other. The maximum distance in this model is set to 1000 feet, as shown in Fig 6.3. Thus, if the communicating nodes are more than 1000 feet apart then traffic gets dropped unless a relaying device is used.

Therefore, normal latency of traffic in general and TTEthernet clock synchronization in this research, before an unexpected latency as in a security breach, depends on the above points among others. Latency caused by these factors can be considered acceptable when devising a latency threshold. Below are the results of simulations carried out on

the network model, useful to understand the presence of a security breach which causes latency of sync frames.

6.3 Result Before Fault Injector is Applied

Results attained from the simulation where the fault injector is not applied are presented in this section to underscore the benchmark and compare the effect of a compromised clock synchronization after the fault is injected in the following sections. In many practical network deployments, the presence of a security breach is discovered sometime after the effect is felt; this could be months or years after the breach has happened, in some cases. Thus, a protection mechanism must be set up to alert the control centre at a calculated point in their tolerance level to a security breach and react with a security solution if such a point is breached. However, that so-called tolerance level remains subjective; hence, it can be set differently for different deployments. It is set to 50 nanoseconds in this model as the tolerable latency for TTEthernet clock synchronization ranges between 40–50 nanoseconds in most cases. How far a deviation in the local clocks from the normally expected delay is acceptable within the maximum tolerable latency threshold is a topic this research does not cover.

It is important to note the assumption considered in this model is that no other forms of security breaches are present within the test network model. The focus of this research is to explore the changes in the latency of TTEthernet clock synchronization frames after a security breach. Hence, all other latency factors are kept constant. As shown in Fig 6.4, the fault injector is not applied to show the results where no security breach is involved.

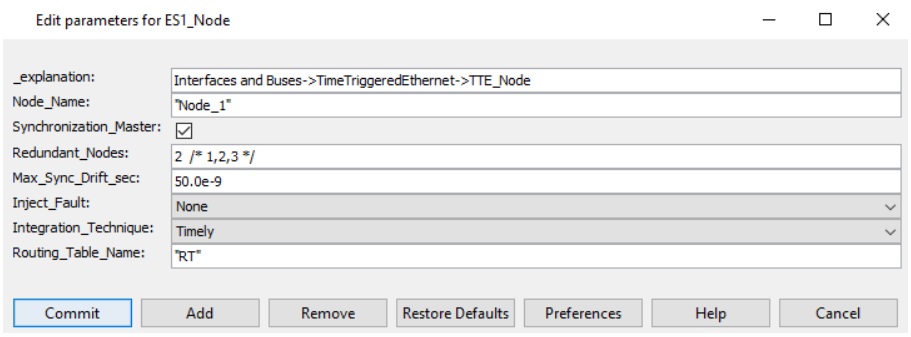

Figure 6.4: Showing network model where fault injector is not applied

The resultant delay of synchronization frames is displayed below. The fact that no fault is injected is justified by showing the maximum and minimum latency values between TTE_Switch2 (Bridge_2) and ES6 (Node_6) is equal. The latency shown at this stage is not a matter of concern for this research as it is nothing but a benchmark to consider when a fault is injected in the coming sections. However, the reason for the latency shown here should be comprehended in relation to the explanation for the reasons given above.

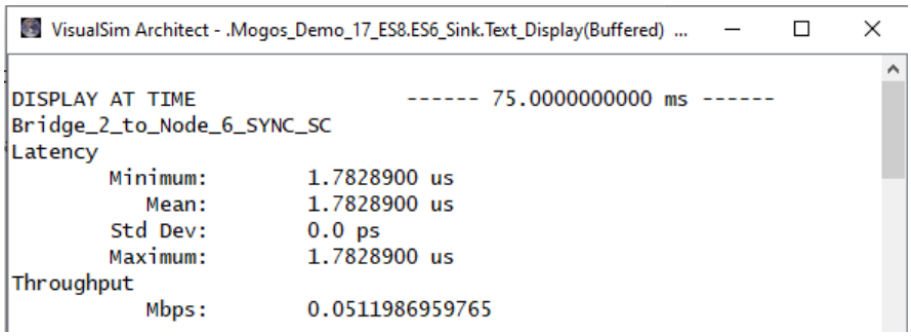

Figure 6.5: Synchronization frames latency in a wired connection between Bridge_2 and Node_6

Three sync client nodes, of which one is a wireless device, are configured in the model to show all types of traffic arriving at these nodes as shown in the figure below; but, most importantly the plotters show that the model mimics an IIoT traffic transaction. Furthermore, it is also important to show the difference in latency caused by the medium of communication where a WAP serves as a relaying device in a multi-hop communication between the network switches and the wireless device.

	Task_Source	Task_Size	Task_Destination	Legend	Latency	Identifier
2	Bridge_2	64	Node_6	Bridge_2_to_Node_6_SYNC_SC	1.74E-06	SYNC
3	Bridge_2	64	Node_7	Bridge_2_to_Node_7_SYNC_SC	1.74E-06	SYNC
4	Bridge_3	64	Node_8	Bridge_3_to_Node_8_SYNC_SC	1.74E-06	SYNC
5	Node_1	64	Node_6	N1_to_N6_TT1_1	1.02E-05	TT1
6	Node_1A	64	Node_6	N1A_to_N6_TT1_1	1.87E-05	TT1
7	Node_1	64	Node_6	N1_to_N6_TT1_1	1.92E-05	TT1
8	Node_1	64	Node_6	N1_to_N6_TT1_1	1.02E-05	TT1
9	Node_1A	64	Node_6	N1A_to_N6_TT1_1	6.19E-05	TT1
10	Node_1A	64	Node_6	N1A_to_N6_TT1_1	1.87E-05	TT1

Figure 6.6: Table showing traffic communicated over wired channels with different latency values

The result in Fig 6.7 shows latency in a wireless medium of communication. Latency naturally is quite smaller in the wireless channel as compared to the wired connection

shown above between Bridge_2 and Node_6. The details of this have been explained in Chapter 7. Nonetheless, this result remains the benchmark; hence, it is not a matter of concern how big or small the latency is at this stage. This research focuses on the effect an injected fault has on the registered benchmark sync frame latency.

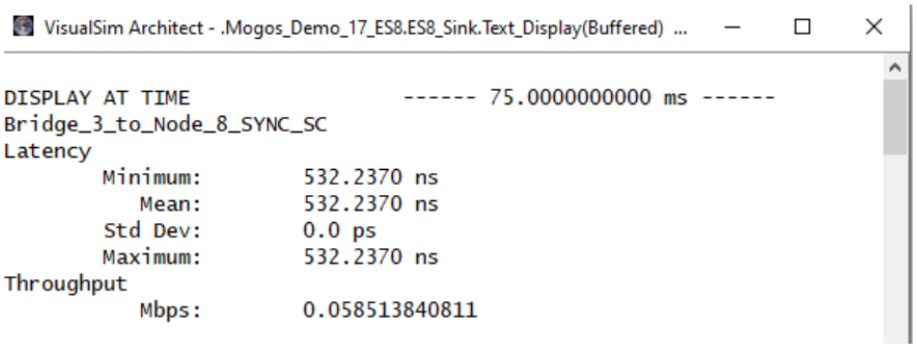

Figure 6.7: Wireless communication between Bridge_3 (WAP) and Node_8 (wireless device)

6.4 Result where Injected Fault Breaches the Maximum Latency Threshold

In this era of technological outbursts, it's naivety to think any network deployment can be fully secure. Security breaches come in all forms, from errors by genuine users and malicious attackers. In this section, a security attack, regardless of the source of the attack or the type of attack, breaches the maximum tolerable delay threshold set for this specific deployment. It should be noted that different acceptable maximum thresholds are set for different network setups, depending on the requirements of the network deployment. The fault injector applied in this case is applied on the link between the WAP and the wireless end system to delay PCFs. This security attack has managed to break the maximum tolerable threshold. The result of such a security breach is shown below.

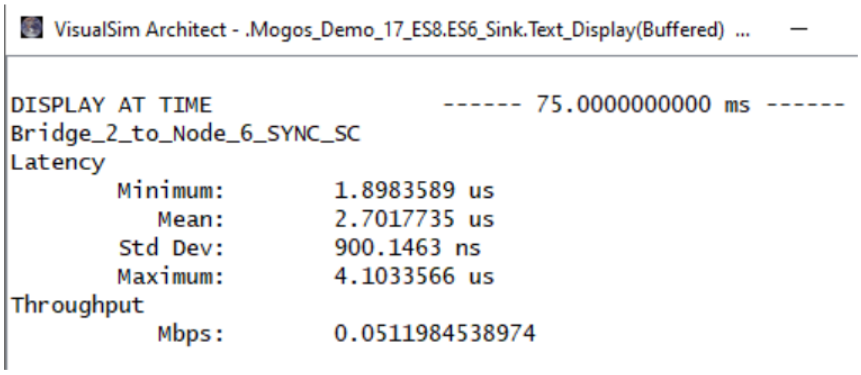

Figure 6.8: Latency shown after a security attack breaches the maximum threshold on a wired channel

Latency, after this type of attack, ranges between 115.4689ns on the minimum latency to 2.3204666us on the maximum latency from the registered benchmark. This shows that the injected fault has breached the 50ns mark which is set to trigger an alarm for a security solution. Any practical network with a security solution set to notify the control centre at the 50ns mark would be able to identify the breach of this type of security attack.

Below is another example of the effect of a similar security attack on the wireless segment of the IIoT.

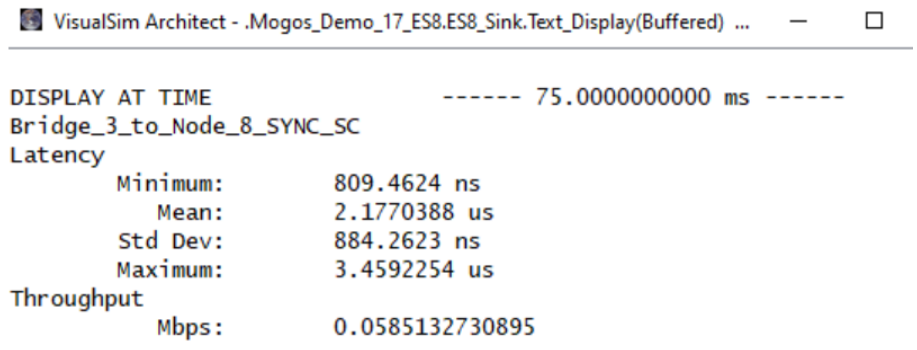

Figure 6.9: Latency shown after a security attack breaches the maximum threshold on a wireless channel

Similarly, the result shows that a security attack of this magnitude causes sync frames to be delayed by 277.2254ns on the minimum latency to 3.6725569us on the maximum latency from the registered benchmark latency. This shows that this type of security attack does breach the maximum tolerable threshold when it gets to the maximum delay threshold. It is evident that the tolerable maximum latency threshold is well breached at

both the minimum and maximum delays registered latencies compared to the benchmark where the fault is not injected. It can also be observed that the effect of injected fault caused a bigger deviation of latency frames in the wireless channel as compared to the wired medium of communication.

6.5 Result where Injected Fault does not Breach the Maximum Latency Threshold

Results in this section show that a security breach can remain within the accepted maximum latency threshold, undetected, for a long time. Clock synchronization is compromised as the fault is injected, but it is calculated so that the maximum delay threshold is not breached. This scenario is designed to analyze the effect of such a security attack on the network model. This is most likely committed by malicious attackers with the intention of not exposing themselves by staying within the accepted threshold.

Below is the result of a simulation after a fault injector is applied on both the wired and wireless network segments to see the effect and compare the latency caused by the registered benchmark.

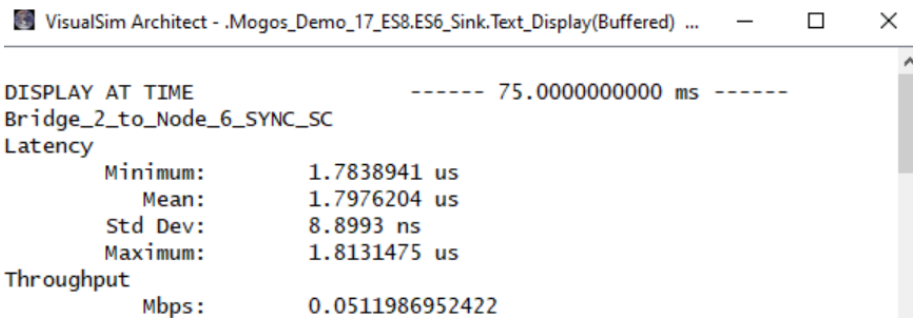

Figure 6.10: Effect of a calculated fault injected on the wired medium of communication

This result shows that the injected fault never crosses the set maximum allowed latency threshold at 50ns on the wired segment of the network model. The increase in latency from the registered benchmark value, on the minimum latency value is 1.0041ns and 30.2575ns on the maximum latency registered for both before the fault is injected and after this specific fault is injected.

Similarly, the same fault is injected on the link between the WAP and the wireless

device to show the effect it would have on the latency of sync frames communicated on the wireless medium. As discussed above, different mediums of communication have their own effect on the latency of all frames communicated; however, this research is determined to show the effect of a security attack on top of the expected delay registered before fault is injected.

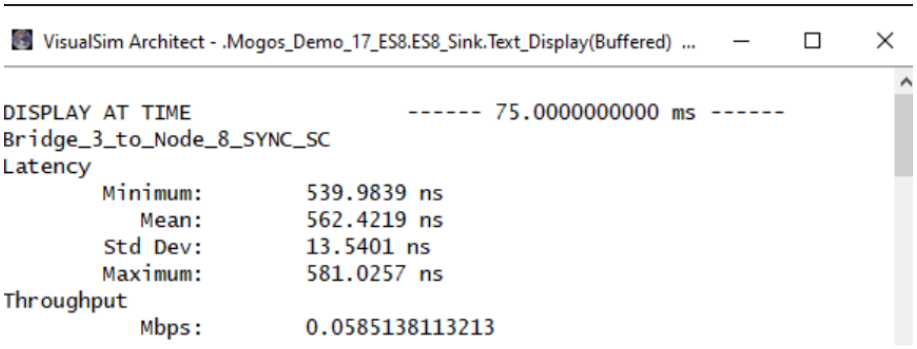

Figure 6.11: Effect of a calculated fault injected on the wireless medium of communication

As discussed above, the result on the wireless segment shows that there is an increase in latency due to the injected fault, but this never crosses the maximum tolerable latency. Compared to the benchmark value registered before a fault is injected on the wireless segment, added latency when the minimum latency is registered is 7.7469ns. When the maximum latency is registered, however, the added latency becomes 48.7887ns. This is close to the maximum allowed threshold, but it did not breach the 50ns benchmark maximum latency threshold.

In conclusion, the results of the simulation carried out at all security breach levels indicate that malicious attackers can design their ill-intentioned security attacks at any level they want and decide if they want to alert the security controls or not. Thus, any security solution needs to come from the analysis of the security breaches and devise a solution fit for the network deployment. Finally, a more detailed analysis to understand the effect a medium of communication has on the latency of sync frames and how those channels are designed for the simulation tool is explored in the next chapter.

Chapter 7

Impact of Communication Medium on Latency of Synchronization Frames: A Comparative Analysis of Wired and Wireless Channels

The network model is set up with the best possible options to represent real-life application scenarios for TTEthernet clock synchronization in an IIoT. One way of proving this is by representing the wired and wireless mediums of communication to study how the delay in synchronization frames is affected by changes in the communication channels. There is a given value, in Visualsim, for the propagation constant which is a measure of the change observed by the phases of propagating waves. It is sometimes referred to as the propagation parameter, propagation coefficient, or transmission parameter and it is set to 1.0 for fibre optic cables while it is 0.8 in twisted pairs and 0.5 in coaxial cables, according to the Visualsim libraries. One major dimension in this research project focuses on IIoT; hence, the propagation constant for a wireless channel which is usually referred to as free space must be considered although Visualsim does not offer this data. Thus, the generic propagation constant formula: $\beta = 2\pi/\lambda$ where λ is the wavelength, is used to find what the propagation constant β is for the wireless medium of communication. It is a generic formula in the sense that there are a host of different factors that change the value of the propagation slightly from the exact value

including but not limited to the distance between communicating end systems and the presence of transmission obstacles such as walls and other factors that can refract, reflect, or block the wireless signal. As shown in the formula given above, the propagation constant is different for different communication channels mainly because the wavelength of different communication channels is different. Therefore, this formula is used to obtain the propagation constant that can be applied to the communication medium between the WAP and the wireless end system in ES8. However, it needs to be declared here that all other factors that can affect the change in wavelength are considered constant to focus on the medium of communication and the differences in latency reflected by the change in the communication channel. Hence, the wired and wireless mediums of communication are analysed below to study the differences reflected in the latency of sync frames caused by the change in the communication channel.

7.1 Wireless Medium of Communication for Sync Frame Latency

The formula given above makes it necessary to find the value of the wavelength to be able to find the value of the propagation constant for the wireless channel. Wavelength refers to the distance between consecutive crests of a wave (Ayu et al., 2021). It is applicable to any form of wave including electromagnetic waves, water waves, elastic waves, and acoustic waves among others. The higher the frequency, the shorter the value of the wavelength.

The formula $\lambda \times F = C$ is used to find the wavelength in free space.λ is the wavelength in free space, 'F' is the frequency and 'C' is the speed of light in a vacuum which is equivalent to $3 \times 10^8 m/s$. Electromagnetic waves generally correspond to the speed of light sent down the free space or wireless medium. Hence, the value of wavelength is directly correlated to the speed of electromagnetic waves or the speed of light and inversely correlated to frequency.

The commonly used radio waves, also called the network bands for Wi-Fi signals, range between 2.4GHz and 6 GHz; although, they can also go as low as 900 MHz or as high as 60 GHz. So, the commonly used frequency of 2.4 GHz is considered for this

scenario. The wavelength of such a wireless transmission would be calculated as follows:

$\lambda = C/F$

where C is the speed of light at $3 \times 10^8 m/s$ and F is given at 2.4 GHz:

$\lambda = \frac{3 \times 10^8 m/s}{2.4 \times 10^9 Hz} = 0.125m$

The next step to find the propagation constant which is the ultimate goal is as follows:

$\beta = 2\pi/\lambda$

λ = 0.125m as is found above and π is approximated to 3.14

$\beta = 2 \times 3.14/0.125$ = 50.24

Therefore, the propagation constant in the wireless medium is set to be 50.24, in this model, to help analyse the sync frame latency on the wireless channel.

The figure below reveals how the propagation constant is configured for the wireless segment of the model between the WAP and ES8, the wireless device. It should also be noted that the value 50.24 is changed to 0.8 when a wired connection is represented.

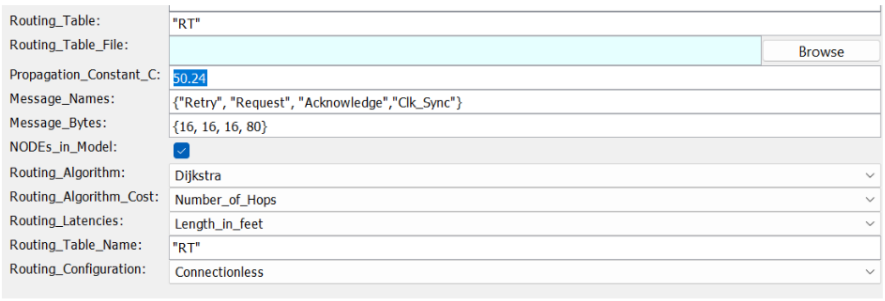

Figure 7.1: Value set in the propagation constant indicates the type of communication medium used

7.2 Wired Communication for Sync Frame Latency

Twisted pair cables are the typical networking cables used for most network infrastructures. Thus, it represents the wired medium of communication in this scenario. Cat5e, again a prominently used cable type among other categories of twisted pair cables is selected for the wired connection for no other reason than the fact that it is typically used for Local Area Networks with data rate of up to 1000 Mbs. For Cat5e, the frequency ranges from 1-100MHZ over a 100 meters distance. Frequency is higher on short distances and fades away as the distance between two end systems increases.

$\lambda = C/F$

Where C is the speed of light at 3×10^8 m/s and F is given at an average of 40 MHz:

$\lambda = \frac{3\times10^8 m/s}{4\times107 Hz} = 7.5m$

The next step to finding the propagation speed which is the ultimate goal is as follows:

$\beta = 2\pi/\lambda$

Where λ = 7.5m as found above

$\beta = \frac{2\times3.14}{7.5}$ = 0.83

The value 0.83 is not too far off from the value given in Visualsim at 0.8 for twisted pair cables.

It's important to note that these calculations are only required in a simulation environment as practical network deployment uses the channels physically. Therefore, the selection of a medium of communication, in this research, affects the latency of synchronization frames as outlined above. The design and simulation of a network model and analysis of results after different levels of faults are injected on the wired and wireless channels pave a platform to introduce the security solution in the form of a rule-based anomaly detection, in the next chapter.

Chapter 8

Comprehensive Analysis and Implementation of Anomaly Detection Solutions

8.1 Introduction

Security attacks on TTEthernet clock synchronization can take different forms. The malicious attacks, analysed in this research, include a security attack designed to stay underneath that so-called 'tolerable maximum latency threshold' as well as a security attack type designed to cause an immediate effect on the target network by breaching the set maximum acceptable latency threshold. From a security solution point of view, any network deployment that is supported with a sound defence mechanism would be expected to defend against a security attack that breaches a maximum tolerable delay threshold; as such a breach would normally trigger the defence mechanism so that a reactive response can be deployed. For the security attack that remains within the maximum tolerable delay threshold, however, it becomes difficult to remediate as this level of attack is deemed tolerable; hence, it does not start the trigger to alarm for a security solution. Moreover, the line where the maximum tolerable delay threshold is drawn is subjective and differs depending on some factors including but not limited to the type of task, industry standards, individual preferences, and organizational goals. Nonetheless, addressing the security breach which remains within the accepted maximum

threshold resolves both types of attacks; thus, the focus, from the security solution point of view, is on how this type of security attack can be defended.

In this chapter, anomaly detection models are explored in more detail. Machine Learning and rule-based anomaly detection methods are investigated followed by a comparative approach to make a case for the main model used in this research which is a rule-based anomaly detection. This chapter concludes by briefly introducing the security solution - rule-based anomaly detection algorithm used in this project before the core algorithm is presented in pseudocode and dissected in detail in this last section of the chapter.

8.2 Comparative Analysis of Anomaly Detection Models

Anomaly detection can be performed on various types of data, such as time-series data, image data, or text data, and can be used in many different applications, such as fraud detection, network intrusion detection, predictive maintenance, and quality control among others. There are several techniques used for anomaly detection, including statistical methods, machine learning techniques, and rule-based systems, (Hilal et al., 2022). Statistical methods, for example, involve comparing the observed data to a statistical model or distribution and identifying data points that fall outside of a certain threshold. This is based on measuring how far a given data point is from the rest of the data distribution. A typical example which used the statistical method is the z-score method which calculates how many standard deviations away a data point is from the mean.

The choice of an anomaly detection technique depends on several factors including but not limited to the type and nature of data, the size of the dataset, the level of expertise of the user, and most importantly the purpose of analysis. Therefore, it is important to carefully evaluate the performance of each method and choose the one that best fits the specific problem at hand. However, a combination of two or more techniques may be used to effectively detect anomalies depending on the requirements set out.

In this section, a comparative analysis of machine learning and rule-based anomaly detection methods are analysed for identifying anomalous sync frames in TTEthernet clock synchronization. While other anomaly detection techniques exist, these two

approaches are preferred for their relevance and applicability to the specific use cases used in this research. This comparative study provides a comprehensive understanding of both methods, ultimately leading to the selection of rule-based anomaly detection for its specific advantages explained below.

8.2.1 Machine Learning Methods for Anomaly Detection

Machine learning techniques involve training a model on a labelled dataset of normal and anomalous data and using the model to classify new data points as normal or anomalous. Deep Learning is a good example of a machine learning technique that uses neural networks to learn complex features and patterns from data. It can handle high-dimensional and unstructured data, such as images, text, audio, etc. It can also generate explanations for anomalies using attention mechanisms and generative models, (Huang et al., 2022).

Machine learning methods are focused on the design and development of algorithms and statistical models so computer systems can automatically learn from data without being explicitly programmed (Nithya and Ilango, 2017). The same authors add that algorithms are designed to learn from examples, adjust to new scenarios, and enhance their efficiency over time without human intervention. Machine learning is used to develop computer programs which can recognize patterns and make precise predictions or decisions based on inputted data. It is used in a wide range of applications, including natural language processing, image and speech recognition, recommendation systems, and predictive analysis (Abada et al., 2022). Some of the commonly used machine learning algorithms include Supervised learning, unsupervised learning, and reinforcement learning, (Jin, 2020).

Deep learning is a commonly used example of machine learning that can be used for a wide variety of tasks including supervised, unsupervised, and reinforcement learning. Deep learning can be used for learning that uses neural networks to learn complex representations of the input data. Typical examples include speech recognition, natural language processing, and image recognition among others (Razvi et al., 2019). It can also be used for unsupervised learning by training neural networks to learn patterns from unlabelled input data, by extracting meaningful features. Similarly, Deep Learning can be

used for Reinforcement learning where an agent learns to take actions in an environment to maximize a reward signal. The goal here is for the deep neural network to get trained to minimize the difference between the predicted and actual reward signals by taking the state of the environment as input and outputs an action or a value.

Therefore, it is important to choose an appropriate model complexity that balances the capacity of the model to capture the underlying patterns in the data, without overfitting to the noise which can be achieved through techniques such as regularization, early stopping, and cross-validation, (Guo et al., 2022).

8.2.2 Rule-Based Anomaly Detection

Rule-based methods involve defining a set of rules that identify anomalous patterns or data points based on domain knowledge. According to (Kiersztyn and Kiersztyn, 2022), rule-based anomaly detection defines a set of rules or conditions for a normal behaviour or patterns in a dataset which makes it easier to put a boundary or a threshold to identify abnormal outliers. Such anomalies can be caused by system errors, human errors or malicious hackers. Regardless of the cause, nonetheless, a fitting anomaly detection method needs to be identified to detect them before harm is done to the system or human practitioner. For example, because this project focuses on the latency of sync frames and the solution outlined targets setting simple latency thresholds to identify anomalous sync frames, rule-based anomaly detection serves to the point. This method uses predefined rules or thresholds to identify data points that deviate from normal patterns. Rule-based methods would be able to flag a temperature reading as anomalous if it exceeds a certain value or threshold. Rule-based systems have some advantages as well as limitations. They are easy to implement and interpret but are limited in the sense that they would not be able to detect unknown anomalies that do not match predefined rules.

Rule-based methods are based on expert knowledge and existing data; hence, they do not require training. In a one-way road, for example, any car that goes in the opposite direction gets flagged by a roadside camera, for anomalous behaviour. The rule in this case is simple and clear which states, that any car that drives in the opposite direction evades the norm; hence, it gets added to the flagged list for further measures by the traffic

authority. Rule-based anomaly detection techniques are widely used in different sectors. It can be used to detect suspicious activity in cybersecurity, and fraudulent transactions in the banking and financial sector, and to monitor system health in industrial applications, among others. After receiving a security alarm for the breach of threshold-based rules, a network practitioner can decide how to react as different network deployments have different defence mechanisms and priorities for their respective scenarios.

Therefore, a rule-based anomaly detection model is a fitting detection technique which uses a set of predefined rules to identify anomalies in data. Different levels of latency thresholds can be defined to trigger a security solution to catch abnormally delayed synchronization frames

8.2.3 The Case for Rule-Based Anomaly Detection

This project started with the aim of securing the clock synchronization protocol in the TTEthernet-based IIoT. The security of network communication, in many cases, starts with the security of clock synchronization. Latency is one of the major security threats facing the clock synchronization protocol as explained in Chapter 4. Thus, one of the easiest and most efficient ways of securing clock synchronization against latency is to set latency thresholds and flag sync frames that breach the latency thresholds. Hence, simple rule-based anomaly detection is preferred in this research to the more complex machine learning methods.

Latency thresholds may need to be dynamically updated as the nature of security attacks is continuously evolving; hence, incorporating machine learning methods could make it a robust system. Nonetheless, for the models designed in this research and the solution sought, it is way simpler to use a rule-based anomaly detection model than to train a machine learning model every time there is new data for simple and stable traffic communication. Advanced machine learning models can even infer the type of security attack based on the nature and pattern of anomaly. For example, they may be able to suggest that a certain security attack is a DoS attack if the anomaly patterns resemble the characteristics of a DoS attack. Nonetheless, as much as this is helpful, the solution sought in this research is the ability to detect and flag a security attack on the TTEthernet

clock synchronization for the security practitioners to analyze and respond accordingly afterwards. Hence, simplicity and ease of interpretation of the outcome is not the only reason a rule-based anomaly detection is preferred for this project; but the fact it does exactly what is sought for in this research - detecting and flagging anomalous sync frames based on predefined latency thresholds. There are two simple thresholds set up to detect anomalous sync frames. If a sync frame is flagged, it simply means it has breached the set latency threshold. This proves another attractive quality which is the predictability of outcome. In machine learning models, however, outcomes are not easily predictable and interpreted, at least not as straightforward as is explained above for rule-based anomaly detection. Similarly, rule-based anomaly detection is more stable in the sense that the same input almost always produces the same output. In machine learning models, on the other hand, especially if an algorithm is trained in a different environment, the result can be slightly different. Another obvious difference is that rule-based anomaly detection does not require training which takes time and computational resources while machine learning models do. To sum this up, this is by no means degrading the power of machine learning methods for anomaly detection. For this specific research, however, rule-based anomaly detection is a good fit and way more effective to achieve the desired outcome.

The local latency threshold, in this research, is used to add another layer to the filtering steps started by defining the global latency threshold. Adding more layers of security can always help refine the detection processes. Like any other anomaly detection system, however, the rule-based anomaly detection method can be improved for more accurate results. In a dynamically evolving network where the behaviour of traffic is unpredictable, however, thresholds need to be dynamically updated to reflect the changes in traffic communication. Sync frames have relatively more stable frequencies or patterns as compared to other traffic types. Nonetheless, external factors or even changes within the network including software upgrades and changes in protocols can change the latency behaviours; hence, the predefined latency thresholds need to dynamically evolve and adapt to the changes. This can be adjusted, in the rule-based method used in this research, by changing the 'latency before fault' or the latency thresholds slightly to reflect the changing behaviour of synchronization frames. Similarly, feedback loops can

be incorporated where network administrators report false positives and false negatives. This can also be used to adjust the values mentioned above so the thresholds can be more focused and effective. Similarly, context-aware models can be devised to reflect the requirement for different latency thresholds in different situations. For example, traffic communicated during the daytime can be treated differently from those transmitted during the nighttime.

Finally, an integration of machine learning and rule-based anomaly detection techniques can be considered to further enhance the effectiveness of the model. Rule-based anomaly detection is a good choice for sync frame anomaly detection. Nonetheless, machine learning methods could also be used to update the set thresholds considering unforeseen situations that might change the pattern or latency behaviour including but not limited to external factors and network or software-related changes. Therefore, although rule-based anomaly detection is a clear choice for sync frames anomaly detection, given its simplicity, the use of machine learning methods can help update the latency thresholds in dynamically evolving networks like the IIoT.

8.3 Introducing The Security Solution: A Presentation Perspective

The security solution sought in this research follows an investigative analysis of data extracted from the simulations carried out with different levels of fault injected in the results discussion section Chapter 6. A dataset containing data from simulations where a fault injector is not applied and where faults of different levels are injected is assimilated and presented to a rule-based anomaly detection model to execute the set rules. This then nullifies sync frames that breach the global and local latency thresholds defined in the model. This relates to anomalous behaviour where a clear rule is set, and the culprit defies the set rules. All sync frames communicated after fault is injected are not supposed to correct themselves because they are designed to stay within the universal maximum tolerable threshold; whereas sync frames communicated without fault injector being applied should correct themselves of any delay. In practical terms, sync frames may

be delayed and not correct themselves for different reasons including a malicious security breach. End systems failing to correct delays in their local clocks after a certain number of synchronization cycles, regardless of the reason, are causing similar effects; hence, they should all be treated as anomalous.

A rule-based anomaly detection model is written in Python. The script has steps where it imports a CSV file, detects anomalies according to the rules defined and adds flagged sync frames into the flagged list. In the first step, it reads data from a CSV file and filters rows with sync frames from among all the different types of traffic generated using the network models simulated in the previous chapter. Having identified the relevant data for analysis, it checks for anomalies in the latency values based on the provided rules. It identifies sync frames which defy the set rules for maximum delay threshold. Finally, it outputs the identifiers of the sync frames that meet the criteria, flags them, and adds them to the flagged lists. The database used for this analysis includes entry from sync frames communicated when no fault is injected as well as when fault is injected to initiate latency where the latency does not breach the 50ns maximum latency threshold but the end systems in question don't correct their local clock to less than 10ns; hence, flagged for breach of the rule and finally, there is entry into the dataset of fault injected where it breaches the 50ns latency instantly; hence, gets flagged for breaching the globally accepted maximum latency threshold.

As mentioned above, two maximum latency thresholds are declared: the global maximum latency threshold at 50ns and the local maximum latency threshold at 30ns. The first rule states that sync frames that are delayed by more than the global maximum latency threshold are automatically added to the flagged list. With the local maximum latency threshold, however, the model continuously monitors the latency values and triggers a counter to start automatically if the latency value goes above the 30ns local threshold. If the end system fails to correct its local clock to a latency value of less than 30ns within 10 synchronization cycles, the current sync frame is added to the flagged list. It should be noted that both latency thresholds are subjective; hence, they can be adjusted to fit one's deployment requirements. Thus, it is possible to monitor sync frame latency at any level using the model set out in this project. All sync frames added to the flagged list

from either rule are treated the same and a relevant defence mechanism is deployed by the network administrators fit for their requirements.

Two different models have been designed to get to the solution stage. The first step involved designing a network model in Visualsim so simulations could be run to understand the traffic communication as well as produce a CSV file of several types of traffic communicated between end systems and network switches or WAPs. For this, a CSVwriter block was used to extract data and save the data in a specified location. As laid out above, different scenarios are being tested which include using different fault injection levels, and integration techniques as well as changing the maximum tolerable thresholds, among others.

The second program that gets executed is another model initially written in Python and then translated into Jython to integrate the model into the simulation tool. Before getting into the details of the Jython model, the configurations used in the simulation tool were modified so certain important steps could be executed in order. After initiating the Visualsim simulation, the first step is to extract the CSV file and overwrite the previously extracted file. This CSV file is then used as input data for the Jython model. A detailed presentation of the rule-based anomaly detection model is presented in the next section.

8.4 The Core Algorithm: Conceptual Design and Pseudocode Breakdown

A rule-based anomaly detection model is designed in Python and then translated to Jython by modifying certain commands in the Python script, as the simulator, can only be integrated with models written in Jython. Fortunately, Jython is an implementation of the Python programming language that is designed to work on a Java platform. Thus, the modification required was minimal. The Python script has been tested in a PyCharm development platform and works well as expected. However, it would be an isolated, standalone, program that can be executed by using existing data extracted from the simulation tool and saved at a preferred location. Therefore, this model needed to be integrated with the simulation tool. In return, running the simulation tool generates traffic

communicated between all network devices and executes both the CSV writer which extracts a CSV file and prints it to the same location as the simulator and executes another program, the Jython model, which then uses the CSV file and applies the rules set out in the model itself. This ensures that results coming out of the Jython program are the outcome of using the latest possible traffic communicated within the network. The final CSV file that includes anomalous frames identified by the Jython program is written to a preferred location which is specified in the model.

Below is the pseudocode which outlines the steps for importing and processing a dataset to detect latency anomalies in the TTEthernet clock synchronization. The algorithm initiates parameters, reads data from a CSV file, processes the data to calculate added latency, and then detects and flags anomalies based on predefined thresholds as shown in fig 8.1. However, for ease of exposition, it is presented in parts followed by a description of what the pseudocode represents.

The 'Main' class is used to integrate the Jython model into Visualsim. This is the main backbone of the program where the Jython script sits.

The following pseudocode outlines the initialization of the Main class, which sets up initial parameters for processing network latency data.

```
Import CSV module


Class Main:


 Function initialize():
   Set ramp start to 0
   Set ramp stop to 100
   Set ramp slope to 10
   Set ramp time to 0
   Set wired latency to 1.74E-06
   Set wireless latency to 5.32E-07
   Set data file to 'C:/Users/BGSAdmin/Documents/ES8.csv'
   Set output file to 'C:/Users/BGSAdmin/Documents
```

```
/flagged_anomalies.csv'
```

This pseudocode focuses on initializing the variables within the main class, capturing the essential setup steps for further processing. The values used are the default values that did not need to be changed as they don't have much to do with the main script coming below. These are parameters of the ramp function which can be explained as such that the 'ramp' starts at '0', has a slope of '10' and has a maximum value of '100'. Ramp is characterized by the linear increase or decrease over time, starting at '0' until it gets to '100' in this case. There is no specific measurement associated with the ramp other than just the numbers used to show the linear increase or decrease, symbolized by the slope.

To the IIoT aspect of the project, the models designed in this research consider the wired and wireless sides of the network model being used. The lines (self.wired_latency = 1.74E-06) and (self.wireless_latency = 5.32E-07) present the given default values of latency when there is no fault or a security breach. There are two different values for two different communication channels. Calculations carried out to help determine these values can be seen in Chapter 6. Hence, the default latency before fault for the wired communication channels used in this research is set to 1.74E-06; while it is 5.32E-07 for the wireless communication channels.

The last two lines of the pseudocode presented above are self-explanatory. Self.data_file is the path to the CSV file where the program reads the data from, whereas self.Output_file is the path to the CSV file where the program writes the flagged anomalies to.

This part of the model is the main link between the Jython program to the simulator tool. The details of how the main body of the model is constructed are represented in the pseudocode below.

```
Function read csv data(file name):

    Initialize empty list data

    Open file name in read mode as csvfile:
        Read csvfile using DictReader as reader
```

```
        For each row in reader:
            Set identifier to row['Identifier']
            trimmed and in lowercase
            If identifier is not 'sync',
            continue to next row
            Rename 'Latency' field to 'Current Latency'
            Set current latency to float of
            row['Current Latency']
            If row['Task Source'] is 'Bridge 3':
                Set latency before fault to wireless latency
            Else:
                Set latency before fault to wired latency
            Add 'Latency Before Fault' to row with
            value of latency before fault
            Calculate added latency as current
            latency minus latency before fault
            Add 'Added Latency' to row with
            value of added latency
            Append row to data

    Return data
```

The above piece of pseudocode reads the csv file extracted from the simulator. It goes on to create an empty data list. It reads the whole data row by row and it identifies all rows that have their identifier value as 'Sync' and adds them to the blank dataset created. It then starts to perform the following.

It renames the 'Latency' field in the newly created dataset with 'Current Latency'. Then, it determines the 'Latency before fault' value. This value is different for wired and wireless communication channels. In a practical scenario, this can be calculated by aggregating the average value in a controlled network where there is no fault or security breach affecting the normal sync frame latency. This can be subjective, but it mainly is

the average day-to-day delay observed by end systems before unusual factors are involved to change the value of latency. On the simulator, however, it is the same value for all end systems where no fault is injected. It is determined in this model that all sync frames originating from Bridge_3 have the values of a wireless communication channel; because Bridge_3 is configured to serve as a WAP. Thus, any traffic originating from the WAP travels on a wireless channel to the wireless end systems. The visualsim block used for WAP and network switch is called 'bridge_' but it is configured differently to meet the different requirements in both devices.

It calculates the 'Added Latency' value and appends it to the dataset. This is calculated by subtracting the default 'Latency before fault' from the 'Current Latency'. Furthermore, it adds a column to the newly created data and populates it with the values of the added latency for every single sync frame. This is helpful for tracking whether a sync frame is in line with the usual latency or has breached the norm. Finally, this dataset is saved for further processing in the following sections of the model.

The next section of the algorithm which outlines the steps taken to flag anomalies and add them to the output file is presented as follows:

```
Function detect anomalies(data, max_time = 0.01, local_threshold_ns

    Initialize empty list flagged_data
    Initialize empty dictionary end_systems

    For each frame in data with index i:
        Set identifier to frame['Identifier']
        Convert frame['Added Latency'] to nanoseconds and store as

        If added_latency_ns > global_threshold_ns:
            Set frame['Flagged By'] to 'Global'
            Append frame to flagged_data
            Clear end_systems dictionary
```

```
            Continue to next frame

        Update end_systems dictionary with the current frame using

        If all frames in end_systems have added_latency_ns greater
            For each frame in end_systems:
                Set frame['Flagged By'] to 'Local'
                Append frame to flagged_data
            Clear end_systems dictionary

    Return flagged_data
```

The above pseudocode defines the 'detect_anomalies' function that is set up to identify anomalies within the communicating sync frames. It uses the new dataset collected and processed by the 'read_csv.data' function to perform the rules and flag anomalies. Important values used as input to perform the necessary functions include:

- The data collected in the 'read_csv.data function.

- Max_time. Sync frames, breaching the locally accepted maximum latency threshold at 30 nanoseconds, are monitored for a maximum time (Max_time) of 0.01 seconds before they are flagged for an anomaly. So, this is an important piece of the rule that defines an anomaly for a sync frame that breaches the local maximum latency threshold.

- Local_threshold_ns. This is the local threshold which is set at 30 nanoseconds. This can be set to different values depending on the needs of the network deployment.

- Global_threshold_ns. This is the global threshold which is set to 50 nanoseconds. Similarly, this can be modified to meet a deployment's needs.

The 'detect_anomalies' function starts by creating two empty data lists. The first is to store flagged anomalous traffic and the second is to store a temporary dataset used to store data until they are processed.

The sequence of operations performed by the algorithm, starting with data import and

culminating in the detection and printing of anomalous sync frames, is described below.

- Firstly, it converts added latency values into nanoseconds.

- It checks if the value of added latency breaches the global threshold at 50ns. If the sync frame is found to have crossed this threshold, then it is added to the 'flagged data' list as a global anomaly as 'flagged by' and another key is used to indicate the type of breach it is as 'Global' breach. The cycle is ended by clearing the 'end_systems' list, and the loop skips to the next iteration.

- If the value of added latency does not breach the global latency, it is then added to an 'end_systems' list for the next steps.

- It checks if the value of added latency breaches the local threshold at 30ns. If the sync frames in the 'end_systems' list are found to have crossed this threshold, they are added to the 'flagged data' list as local anomalies as 'flagged by' 'local' anomalies. The cycle is ended by clearing the 'end_systems' list and skipping to the next iteration.

- Finally, this function returns the 'flagged data' list as global or local anomalies

- The write_csv_data function is there to write the flagged data into a csv file. This takes file_name, the name of the output file, and the data that is prepared to be written. It follows simple steps to finish the process.

- It checks if there is data to write. If there is no data, it just skips the writing process, and no output file is written.

- If there is data, however, it uses the csv.DctWriter to create a writer object to write dictionaries into a csv file using the first lines of the data list as column names.

- Finally, the column names are published in a csv file as headers, and it adds all dictionaries in 'data' as rows.

The 'ramp' function, as mentioned at the beginning of this chapter, has all the values set by default as it is not part of the main script written in Python. It calculates value based on the 'ramp' function. If the ramp's start time is after time is inputted, the ramp value increases linearly until the ramp stops, according to the ramp slope. If the ramp's start time is before the time input, however, the ramp's value starts value.

Similar to the 'ramp' function, the 'fire' method is not designed as part of the Python model but is essential as it does the main processes the model is designed to do. It initiates

the anomaly detection processes, by reading the csv file and identifying anomalies before it writes flagged anomalies to an output csv file. It also prints the total number of flagged anomalies.

Finally, after defining the main class and its methods, the program creates an instance of the 'main class 'main_obj' = Main (). It then calls the 'fire' method, in this instance, to start the entire process of reading data from the input csv file, detecting anomalies within the data, writing the data containing the anomalous sync frames to the output csv file and printing the total number of flagged anomalies.

It should be noted that the main class is designed to help integrate the Jython model into the visualsim simulator. Other platforms, as in practical scenarios may require an API that is used to integrate the model to their platforms. Nonetheless, it is important to remember that the Jython model can also work as a stand-alone model by using a ready-made csv file.

It is carried out by appending a new column for the flagged data which stores 'local' or 'global' values. This is followed by converting added latency values into nanoseconds. It does this by multiplying the added latency value by $10\hat{9}$. This function mainly focuses on identifying anomalies and labelling them as global if they breach the globally accepted latency threshold and 'local' if they are in breach of the local latency threshold set to identify security breaches designed to stay under the skin of the globally accepted latency threshold. A maximum time which can be set to any value desired, but it is set to 1 second in this case, is set to confirm or discard flagged anomalies caused by breaching the local latency threshold. The global threshold which flags sync frames delayed by more than the globally accepted latency threshold at 50ns, which is true with most TTEthernet clock synchronisation in general, is also defined in this section. The acceptable latency threshold for TTEthernet clock synchronisation depends on the clock synchronization accuracy required for the specific application but it is mostly between 40ns – 50ns. Hence, it is set up to be 50ns for this research, knowing it can be modified to different values as mentioned above. Consequently, any sync frame that crosses the global latency threshold is directly added to the flagged list. Finally, the local latency threshold flags end systems if sync frames destined for the specific end system have an added latency that crosses the

30ns latency threshold which in turn triggers a counter and counts up to 1 second before it confirms the sync frame in question is flagged as anomaly or otherwise released as benign. This section ends by returning the list of flagged data.

This chapter explored potential anomaly detection methods to defend against latency-related security breaches in the TTEthernet clock synchronization in an IIoT. It has further narrowed the best-fit solutions to machine learning and rule-based anomaly detection methods. It accepts that machine learning algorithms could be integrated to better the security solution by dynamically adapting to changes and re-tuning the maximum latency thresholds but a rule-based anomaly detection is an adequate security solution to monitor and flag sync frames if they are found to have breached the set latency thresholds for the scenarios used which are designed to mimic real-life IIoT deployments. Therefore, the core algorithm designed to address the latency of TTEthernet clock synchronization frames is presented using a rule-based anomaly detection technique. The steps used to flag a sync frame as an anomaly are summarized visually in fig 8.1. Multiple network models are designed to test the efficacy of the core algorithm in different deployment environments, in the next chapter.

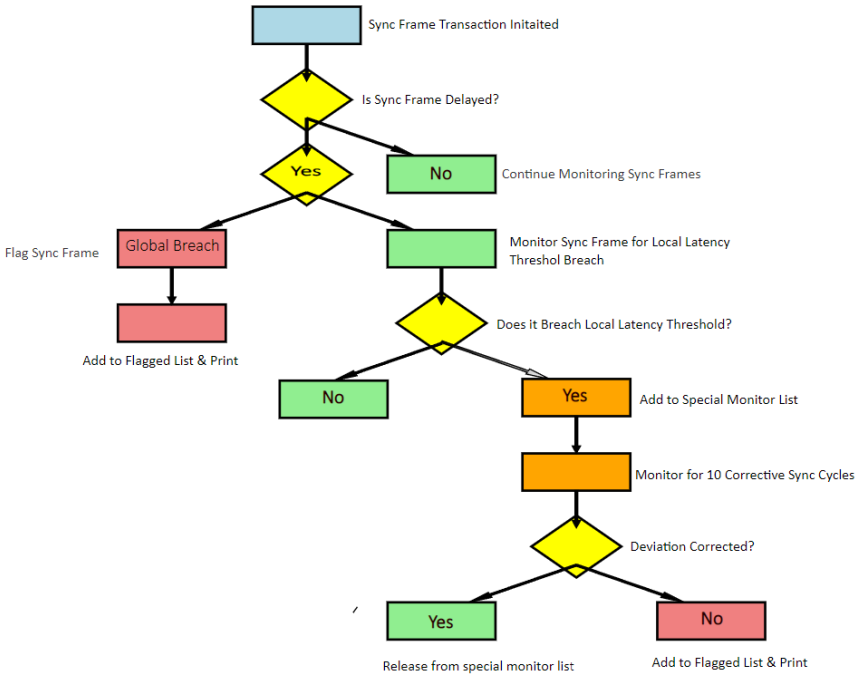

Figure 8.1: Visual aid representation of the security solution

Chapter 9

Evaluating the Core Algorithm for Securing TTEthernet Clock Synchronization Across Diverse IIoT Environments

The evaluation of the security solution starts by designing diverse network models with different sizes and complexities to represent the various network deployment scenarios and running simulations on all network models. The first phase involves the design and configuration of network models. These network models are designed for wired and wireless environments with different sizes and complexities. The second phase focuses on configuring the network models for simulation and analysis. The topologies employed are explained in detail to clarify the network setup used to test the security solution. This is followed by outlining the testing protocol before simulations are run. Finally, the results of simulations run on the network models are analysed in detail. This is designed to test the validity of the security algorithm in all deployment environments.

9.1 Designing Network Models

The proposed solution is designed to apply to all network sizes and topologies in a TTEthernet-based IIoT, anywhere in the world. The first crucial step is to establish what

'latency before fault' equates to, that is latency before a security breach or fault creeps into a network. There is inevitable latency of sync frames related to the usual factors including but not limited to the processing and transmission of sync frames, the type of communication channel or cables used, the distance between end devices, natural clock drift, as well as environmental factors such as temperature and humidity among others.

'Latency before fault' refers to the latency of sync frames that is accepted as normal after the above factors are considered. However, it should be noted that the bigger and/or more complex the network, the harder it is to have an accurate value for 'latency before fault'. Nonetheless, a clearly defined sync frame latency before a security breach intervenes must be established so that additional latency can be measured against this accepted organic sync frame latency. Different network deployments have different latency requirements. For example, real-time systems that involve critical applications require lower latency than non-critical systems. Thus, the solution offered for different critical networks needs to fit the specific requirements of the network used. Therefore, the solution forwarded in this research is easily modifiable to fit the latency requirement of individual network deployments. Similarly, accepted latency thresholds can be modified depending on the latency requirements of the network deployment used.

Latency before fault can be established by considering historical sync frame latency data. The normal latency that a sync frame would take to communicate between the CM and end systems can be achieved by checking the historical sync frame latency data under normal operating conditions, bearing in mind that no security breach has taken place for the period the historical latency data is collected. In this research, a simulation tool is used where external interference is controlled; hence, the sync frame latency observed before the fault injector is applied could be used as the base latency before fault and is used as a benchmark to measure against the sync frame latency observed after the fault-injector is applied. This makes it easier to differentiate if the latency of sync frames has breached the maximum globally accepted latency threshold or the locally accepted maximum latency threshold set out in this research.

The complexity of IIoT, in general terms, depends on the number and varieties of End Systems employed to set up the network, network topology, security requirements,

data volume communicated, and integration with legacy systems, among others, (Mirani et al., 2022). It's expected that bigger network sizes have more end systems and networking devices compared to smaller networks. Consequently, bigger and more complex networks have more complex network topologies. The network topology used in a small network could be a simple star topology while it is more complex for larger networks as multiple star-topologies, Tree and/or mesh networks or a hybrid of those topologies are interconnected to link all end systems and communicate traffic. Below are the network topologies used for the network models in this research.

Star Topology: This refers to a network topology where all end systems are connected to a central point; switches and WAPs in this research. Traffic communication between end systems takes place through the central point of communication, (Deepak et al., 2022).

Mesh Topology: This is another form of network topology in which all end systems are directly connected to each other. A network topology where all end systems of the same network are directly connected to each other is called full mesh topology. A partial mesh topology is formed in situations where a few end systems cannot be directly connected to every other end system in the network, (Lim, 2016) & (Prehanto et al., 2021).

Other factors that indicate the size and complexity of a TTEthernet network include but are not limited to the number of traffic communicated between end systems as well as available redundancy options. As the number of network switches configured for CM increases, sync frames get more options to communicate between end systems. Thus, bigger TTEthernet networks indicate the presence of multiple networking devices; hence, potentially, a better option for redundancy.

TTEthernet is designed for wired networks to enhance the Ethernet connection by providing deterministic communication in critical, real-time systems which makes it an attractive option for avionics and industrial applications, among others (Lisova et al., 2014). Hence, the fact that IIoT is considered for this research makes the network model used more complex than it would have appeared in a traditional wired LAN. The addition of just one WAP and a single wireless device, as observed in this research, turns a small network into a more complex one. The complexity of a network increases as the network

size increases; thus, the complexity and size of a network are usually positively correlated. However, it is slightly different for a TTEthernet clock synchronization as the addition of a wireless segment makes it more complex compared to adding the same number of wired devices to the network. So, one of the most important results that can be achieved in a TTEthernet-based IIoT is a seamless integration between the wired and wireless segments on top of getting the TTEthernet benefits on the wired LAN.

One of the main advantages of TTEthernet is that it is possible to configure a network with as many SMs and CMs as required for the specific network architecture. The decision of configuring an end system or a networking device as an SM or CM depends on the system requirements (Xu et al., 2017), as it is important that they are configured to generate and distribute appropriate timing signals to all devices in the network while network interfaces are set up to prioritize traffic timing. Network topology is another factor that determines if an end system should be configured as an SM or CM. For example, the wireless segment of an IIoT is difficult to configure for redundancy which is a typical characteristic of a TTEthernet setup as wireless devices are only connected to a single WAP, at a time. Thus, although it is important to realize the limitation as laid out above, the size of a network is not as determinant in TTEthernet as it is in a normal computer network when it comes to synchronization frame latency.

It is difficult to categorically call a specific network deployment as small or big as there is no clear boundary. Nonetheless, it has been a norm to call one large or small depending on some factors as pointed out above. Therefore, on top of the main network model used for the simulation in this research, two bigger and more complex network models are designed and used to test the validity of the solution offered in this paper.

9.1.1 Designing Network Model 1

A TTEthernet network of 8 end systems, of which one is a wireless device, two network switches, and a WAP is utilized for the network model used in this research. Five out of the end systems are configured as SMs. The remaining two end systems, directly connected to the switches are configured as SCs. Similarly, the WAP and the wireless device are configured to be SCs. Both network switches are configured as CMs. This

forms a combination of star and mesh network topologies.

- **Star Topology**: All end systems, except the wireless device are directly connected to a central point of communication (the switches). So, this is the basic example of a star topology.

- **Mesh Topology**: The fact both network switches and the WAP are directly connected means there is an aspect of mesh topology as devices are directly connected to each other.

- **Simplified star topology**: A single wireless device connected to a central point (the WAP) means there is a point-to-point connection which can also be considered as a simplified star topology as the WAP can be considered as a central point where the wireless device is connected to.

It should be noted that although the wireless segment is not set up to make use of the benefits of TTEthernet synchronization, it still benefits from the multiple routes and determinism functionalities the wired section of the network is configured for.

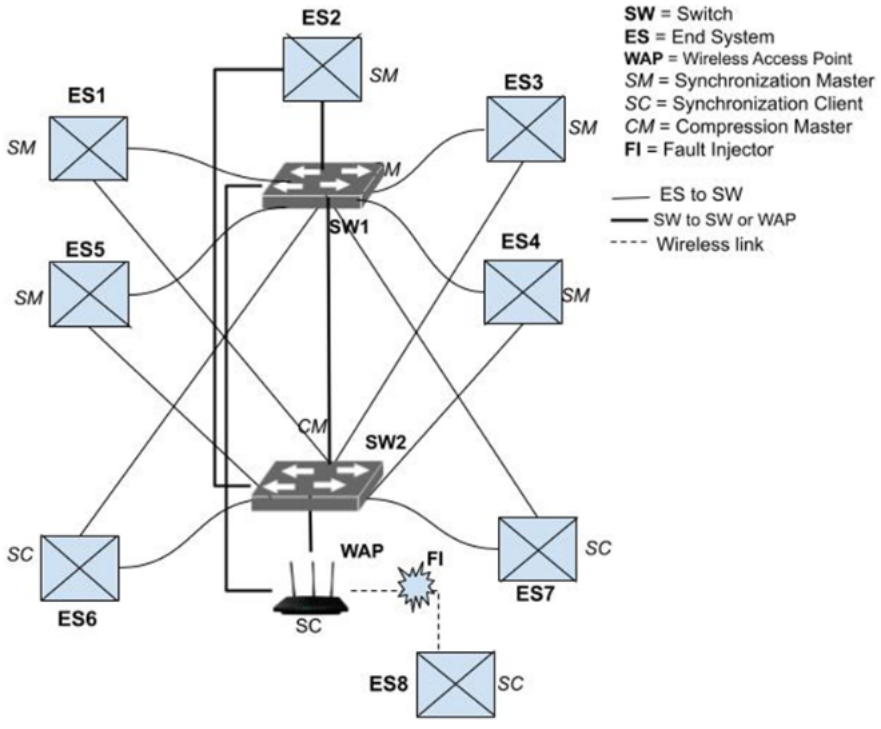

Figure 9.1: The Design of Network Mode 1

9.1.2 Designing Network Model 2

The network model used in this section is an upgrade on the first model in terms of network size. A network of 12 end systems along with 4 network switches is used. Eight of the end systems are configured to be SMs and the remaining 4 are SCs. This model is designed to explore the simplicity of configuring and/or analyzing a fully wired TTEthernet network. Traffic can travel using multiple hops; although sync frames should not need to use multi-hope as intra-cluster synchronization offers an opportunity to synchronize within individual clusters within a network. The network topology used in this model is as follows:

Star Topology: As in network model 1, all end systems are connected to the switches forming two clusters of star topologies.

- **Mesh Topology**: All four switches are directly connected to each other forming a mesh topology.

Therefore, it can be concluded that there is a mix of star and mesh topologies employed in this network model

9.1.3 Designing Network Model 3

This network model takes network model 2 plus two more WAPs which are also connected to 8 wireless devices each.

The configuration of end systems and switches for the synchronization operation on the wired segment of the network remains the same as in network model 2. All wireless devices and WAPs are configured as SCs. Theoretically, the WAP could have been configured as a CM and a few of the wireless devices as SMs. However, in practical terms, most wireless devices lack the resources required due to inherent wireless communication challenges for precise and deterministic timing synchronization, (Seijo et al., 2021) & (Luong et al., 2018). Wireless communications are subject to challenges including signal degradation, interference, and propagation delays, among others. On the contrary, SMs normally require high reliability and low latency communication channels which cannot be wireless channels. Therefore, wireless devices require external correction techniques to make the communication channel more reliable.

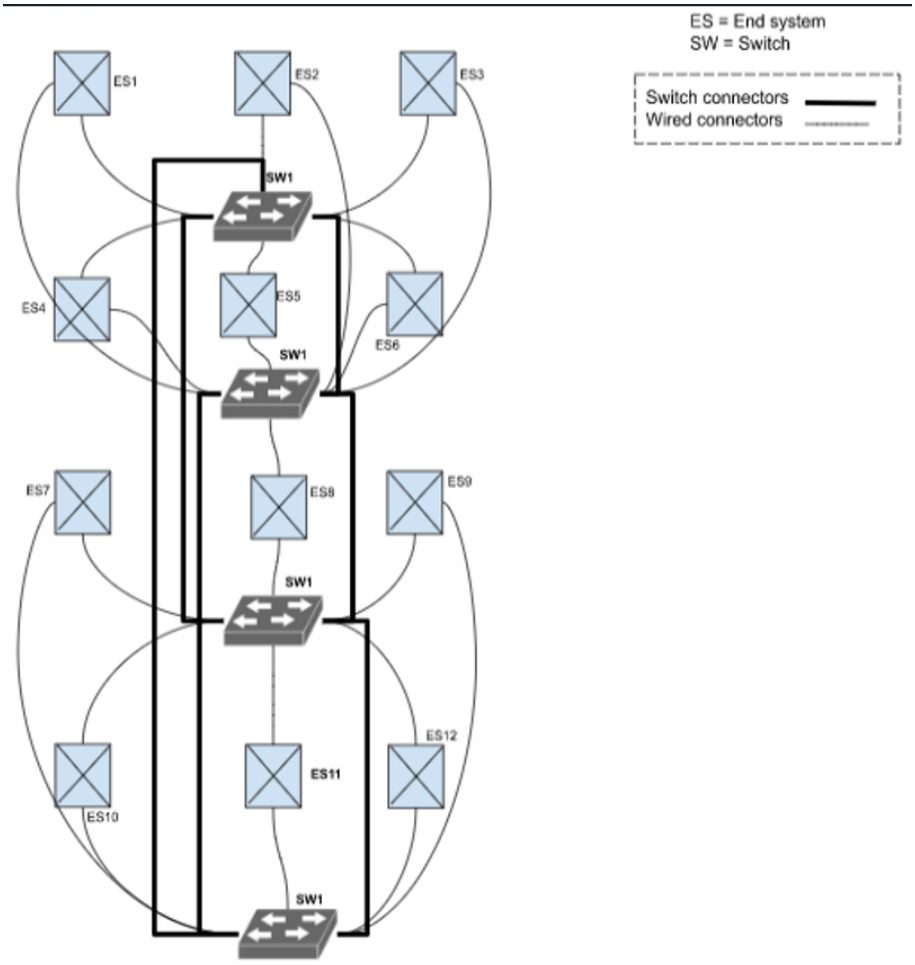

Figure 9.2: The Design of Network Model 2

Network topologies used in this network model are a hybrid of star and mesh topologies.

- **Star topology**: Two clusters of end systems connected directly to the wired switches form star topologies, using the switches as the central point of communication. Similarly, two other clusters of wireless devices directly connected to the two WAPs form star topologies of their own, using the WAPs as the central point of communication.

- **Mesh topology**: All switches and WAPs are directly connected to each other creating multiple routes for traffic transmission. Thus, they form a good example of mesh topology.

Network model 3 is the largest and most complex TTEthernet network used to test the security solution forwarded in this research. Nonetheless, it should also be noted that it is by no means the largest TTEthernet network there is. However, if the security solution is found to stand in all the three network models used here, it can be assumed

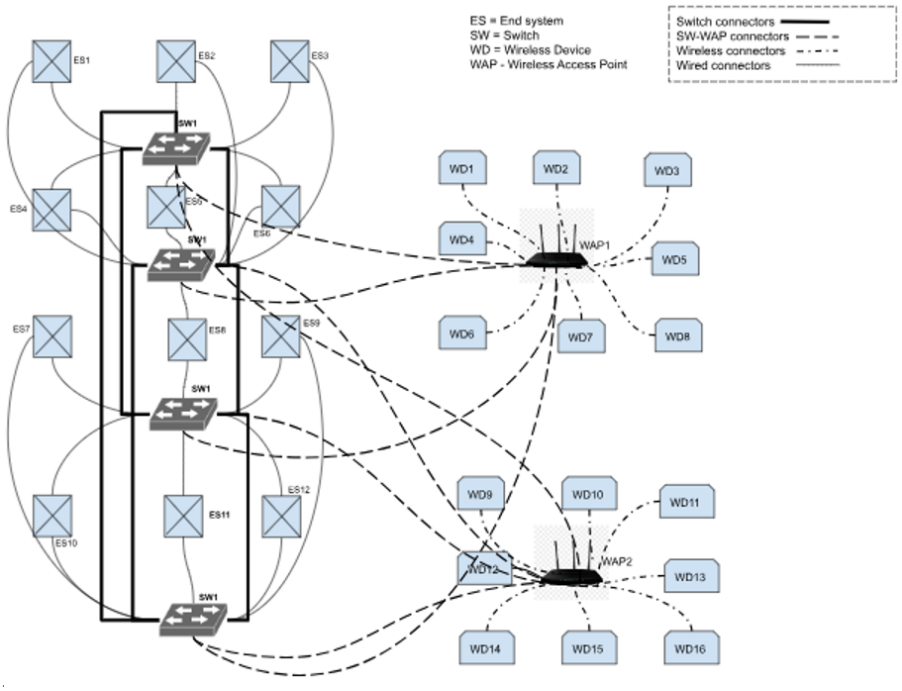

Figure 9.3: The Design of Network Model 3

that it works in all TTEthernet network scenarios. Furthermore, as mentioned in the literature review, Chapter 2, (Tang et al., 2018) specifies an intra-cluster TTEthernet clock synchronization in multi-cluster networks that can be modified without affecting the inter-cluster synchronization. Thus, the size of a network or the number of clusters in a network does not directly affect the TTEthernet synchronization QoS as end systems are always fed with sync frames broadcasted from the closest CM, mostly from within the same cluster. Those CMs are also fed with sync frames from directly connected SMs within the same cluster. Thus, it is expected that the outcome of the test carried out for the validity of the security solution in these three network models would represent all TTEthernet networks.

Thus, three network models are designed, as detailed above, enough to show the complexities in network deployments for TTEthernet clock synchronization. It's designed that testing the core algorithm in these network models confirms its validity if it stands in all network scenarios. The evaluation process of the security solution is carried out in the next chapter.

9.2 Evaluating the Proposed Solution across Different Network Environments

9.2.1 Introduction

The security solution, which is presented in Chapter 8.4, is designed so that it can be applied to any TTEthernet network and improve the security of clock synchronization from latency-related security breaches. Thus, it is set for a test in this chapter using different network sizes and topologies presented in this chapter. It is important to note that this research focuses on TTEthernet clock synchronization; hence, the benefits that come with TTEthernet, including but not limited to redundancy and the use of multiple SMs and CMs could affect the outcome of the test carried out on the different network environments. The capability that TTEthernet offers to configure any switch as a CM and surround it with enough SMs to feed the original clock to the CM means that the size of a network does not matter when it comes to TTEthernet clock synchronization. As stated in the literature review section, the SAE AS6802 standard (Ethernet, 2016) specifies that intra-cluster synchronization can be carried out leaving the inter-cluster synchronization unaffected. This is also backed up in (Tang et al., 2018). Therefore, it is expected that the security solution that is being offered in this research should be effective in all sizes of TTEthernet networks.

9.2.2 Configuring Network Models for Simulation Execution

Consideration has been taken that the proposed solution is effective in all TTEthernet network environments. In doing so, three network models, designed are configured with varying sizes and complexities. In visualsim, every node block as well as their configuration blocks are manually added to the development platform and modified to fit the requirements of the network model desired for simulation. This has limited the ability to design a network model with hundreds of end systems and networking devices. Nonetheless, the researcher believes that any three networks with varying sizes and complexities should be enough to prove this point as size is not a big factor in the validity

of the proposed solution. Network models set up for the simulation, are presented below before the offered solution is tested as a security solution for all network environments where TTEthernet clock synchronization is used.

9.2.2.1 Configuring Network Model 1

The first network model represents most small TTEthernet networks that include wireless devices as part of the network. This model was designed as a small IIoT. It has 11 devices where two are network switches, seven wired end systems, one WAP and one wireless device. This looks small considering the number of devices involved, but in the TTEthernet paradigm, it is more than that as it has the complexity of combining the wired and wireless segments which are completely different as beneficiaries of the TTEthernet services. TTEthernet is designed for the wired network. So, this network model is designed to show that the proposed security solution can still be used seamlessly in an IIoT where wireless devices are a major part of the network setup.

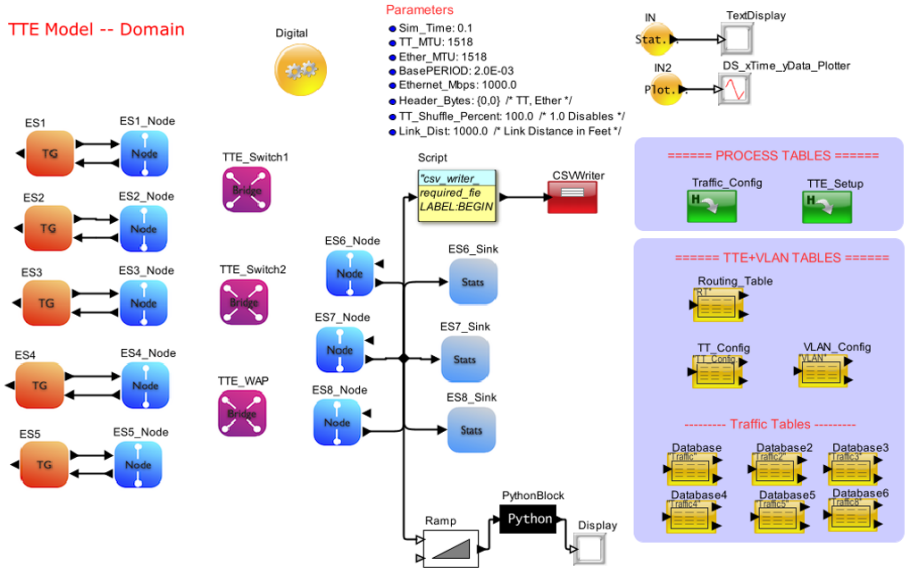

Figure 9.4: Network Model 1

9.2.2.2 Configuring Network Model 2

The second network model is designed to represent the wired network environment. It consists of 16 devices in a completely wired network. Devices used to set up this network model, as shown in the picture below, include four network switches and twelve end

systems connected in two clusters. six end systems are directly connected to the first two switches and the other six end systems are connected to the next two switches. The reason two switches in a cluster are required is that TTEthernet is designed for redundancy where end systems always maintain two connections; one of them is a physical connection and the second is virtual. Hence, traffic travels in multiple routes. All four network switches are directly connected to each other so all twelve end systems can communicate with each other using multiple routes. The three layers of visualsim blocks on the right side of the picture are used to configure the interconnections and traffic routes in the network. This network model is designed to show the security solution can be used in a strictly wired TTEthernet network which has a small to medium network size.

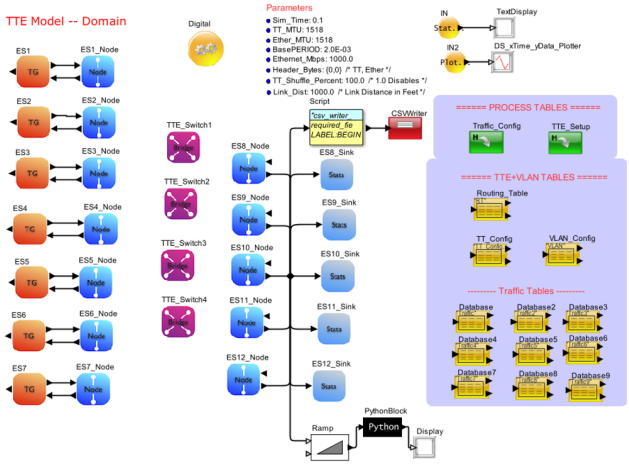

Figure 9.5: Network Model 2

9.2.2.3 Configuring Network Model 3

The third network model is the largest and most complex of them all. It is designed to test the viability of using the security solution in a complex IIoT setting. The network model includes wired and wireless connections. It is designed with thirty-four devices to set up the network model. The wired section of the network is network model two which is made larger and more complex by adding another two clusters of wireless devices. Thus, four clusters are set up using network model two and two other wireless clusters which consist of eight wireless devices each connected to two WAPs. As detailed above, wireless devices are only connected to the one WAP that links them to the local LAN. Hence, there is no redundancy option or alternative route for traffic travelling between

the WAP and the wireless devices. Hence, all wireless devices and WAPs are configured as SCs. Fig 9.7 does not show the fault injector and the Python block just as there is no end system in display. This is because the development platform is too small to display thirty-four devices and their traffic configuration blocks. Therefore, they have been added to containers called compositeActors. So, the three compositeActors displayed in the picture are hosting the wired end systems and one cluster each for the wireless devices connecting to the WAPs.

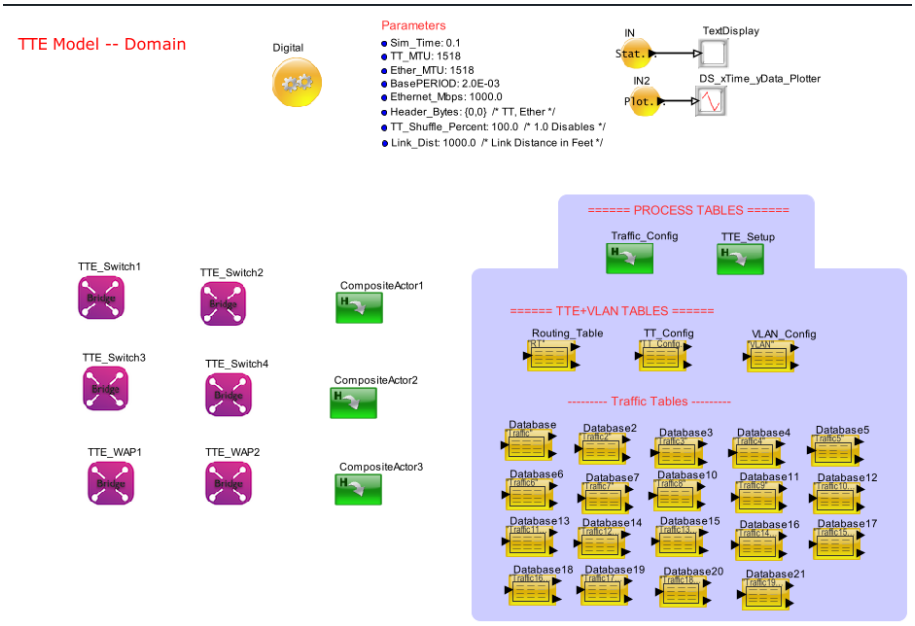

Figure 9.6: Network Model 3

The end systems where the csv writer and the rule-based anomaly detection model are contained in the first container. Below is a screenshot to show the setup.

9.2.3 Testing Protocols

The protocols employed to test the security solution outline the detailed considerations taken. The three network models are configured in the same way. Their only differences stem from the increase in the number of end systems and networking devices included. Below are the main testing protocols that explore the performance metrics relevant to evaluating the performance of the proposed security solution.

- Simulations are run on all three network models before the fault injector is applied and produce a latency value. That value is used as the default 'latency before fault' to calculate the added latency after the fault injector is applied.

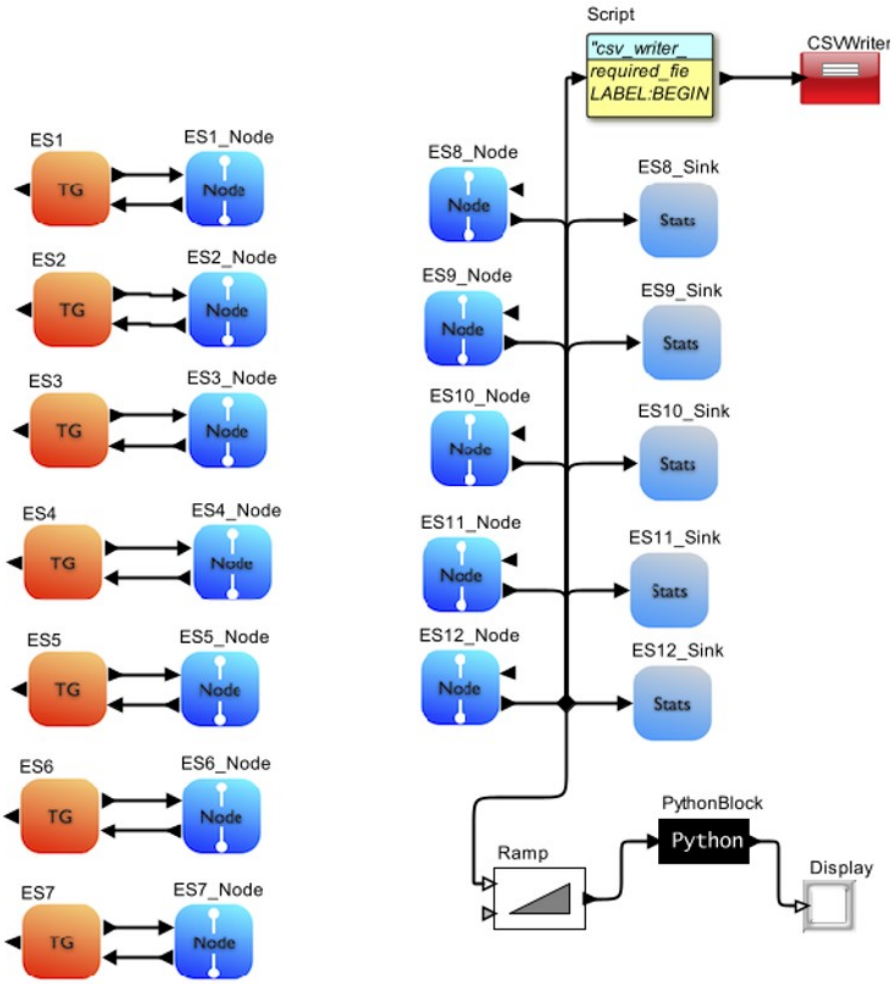

Figure 9.7: Network Model 3, CompositActor1

- They are all set to the same value of 0.1 seconds for sim time. This produces enough traffic to use as input for the anomaly detection model to execute the set rules. 0.1 seconds simulation time is not as small as it sounds, and it can always be modified to different values as required. Besides it takes 10 milliseconds for all end systems to receive a new synchronization frame to correct their clocks. Increasing this value increases the time it takes to produce a csv file and it invariably takes too many processing resources as was evident from the laptop originally used to run the simulation which struggled to run a longer simulation period. For the general picture, the laptop was an Intel i5-2450M CPU @ 2.50GHz with 4GB of RAM and 300GB HDD. Nonetheless, this should not be too much of a challenge in a practical network scenario as the model would be used on a stronger machine that is already used to analyze real-time traffic communicated within a network and pull real-time data to execute rules set up in the model. For efficiency purposes, however, this can be optimized by removing irrelevant traffic from the data

pulled by the rule-based anomaly detection system to work with lighter and more relevant data. A point to note is that the simulation in visualsim is used to generate all types of traffic which is then used as an input for the anomaly detection model. However, when running the rule-based anomaly detection system in a practical deployment, data circulating within the network is used to execute the rules set out in the anomaly detection model from a designated workstation that hosts the database. Thus, a computer struggling to run the program for lack of computational resources is not likely to be an issue.

- All models are configured to use the same integration technique which is the Timely-block. This technique stops other traffic from communicating unless there is enough time for the full frame transaction before the next scheduled communication of a TT frame. This is not the most efficient considering the time wasted to wait for the TT frame to start but all three integration techniques have their own deficiencies.

- The same level of fault is injected in all network models to examine changes in the outcome. This is applied on all systems configured as SCs mainly for consistency reasons. It should be noted, however, that all end systems receive the same sync frame which is broadcasted by the CMs to adjust their clock. Thus, the fault injector could be applied to any or all end systems.

- A propagation constant value which determines whether the medium of communication is wired, or wireless is applied for wired at 0.83 and for wireless at 50.24 in another round of simulation as it was not possible to have a configuration for both the wireless and wired medium of communication.

9.2.4 Simulation Results and Analysis

Simulation, using Visualsim, is run on all three network models. It is run multiple times to explore all outcomes and limitations when a fault injector is not applied as well as when it is applied to replicate security breaches, delaying synchronization frames on their route to the destination end system. Simulations are also carried out to inspect the proposed solution's effectiveness on the wired and wireless sections of an IIoT. The latency breaches inducted by the fault injector are intentionally designed to stay within the accepted maximum latency threshold at 50 ns or breach this threshold. Another latency

threshold is set at 30 nanoseconds designed to monitor sync frames delayed by more than 30 ns but less than the maximum acceptable latency threshold at 50 ns. Below is the outcome of simulations run on all the network models and befitting analysis to the specific simulation outcome.

9.2.4.1 Validating the Proposed Security Solution on the Wired Section of an IIoT - No Fault Injected

This section focuses on validating the proposed solution by running simulations on different network environments. The main factor is, however, that no fault injector is applied for all the simulations to see if any sync frames are flagged for anomalous latency where there is no fault injector. The expectation is that no sync frame should be flagged as an anomaly to validate the proposed security solution.

Network model 1

Network model 1 is the main model designed for this research until two larger and more complex network models are designed to validate the credence of the proposed security solution. Below is the first simulation run on Network Model 1. This is done where no fault injector is applied to the end systems or communication channels.

	Task_Source	Task_Size	Task_Destination	Legend	Latency	Identifier
2	Bridge_2	64	Node_6	Bridge_2_to_Node_6_SYNC_SC	1.74E-06	SYNC
3	Bridge_2	64	Node_7	Bridge_2_to_Node_7_SYNC_SC	1.74E-06	SYNC
4	Bridge_3	64	Node_8	Bridge_3_to_Node_8_SYNC_SC	1.74E-06	SYNC
5	Node_1	64	Node_6	N1_to_N6_TT1_1	1.02E-05	TT1
6	Node_1A	64	Node_6	N1A_to_N6_TT1_1	1.87E-05	TT1
7	Node_1	64	Node_6	N1_to_N6_TT1_1	1.92E-05	TT1
8	Node_1	64	Node_6	N1_to_N6_TT1_1	1.02E-05	TT1
9	Node_1A	64	Node_6	N1A_to_N6_TT1_1	6.19E-05	TT1
10	Node_1A	64	Node_6	N1A_to_N6_TT1_1	1.87E-05	TT1

Figure 9.8: Data produced where no fault is added on Network Model 1

The table above shows the first 10 rows from a csv file which contains 14,517 rows, after 0.1 seconds of simulation time on network model 1. This is the raw data that gets communicated between end systems in the network. However, sync frames are broadcasted from the CMs (network switches in this case) to all end systems, the above data is extracted from traffic targeted at the SCs configured to receive sync frames, for ease of modelling and simulation. The headers of the dataset above are self-explanatory as can be seen below:

'Task_Source' is the source of the sync frame being communicated.

'Task_Size' is the size of the frames being communicated.

'Task_Destination' is the destination the frames are being communicated.

'Legend' – describes the frame direction and the frame type.

'Latency' is the latency associated with a frame travelling to reach its destination from the 'Task_Source'.

Identifier - is the type of frame that is being communicated. These frames can be 'Sync', TT, RC, or Ethernet frames.

TTEthernet supports all types of traffic; hence, the sample is taken from a file that contains TT, RC and BE traffic. The csv file produced by simulating the network models is used as an input for the rule-based anomaly detection model to execute the rules set out in the program. It is expected that the program should only produce flagged anomalies and append them to the csv file if the added latency breaches either the local or global latency thresholds. Nonetheless, as no fault is added in this scenario, no csv file with flagged anomalies is produced as expected. Therefore, it can be said that this simulation produced the expected outcome where no anomalies were detected as there was no fault injected. In a practical network deployment, however, it is not possible to expect that no anomalies would be produced as security breaches are not normally expected. Hence, it is important that the program is run as frequently as possible to nullify any latency-related security breaches.

Network model 2

Like Network Model 1, the following extract is the result of a simulation run on Network Model 2, where no fault is injected.

	Task_Source	Task_Size	Task_Destination	Legend	Latency	Identifier
2	Bridge_2	64	Node_8	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	SYNC
3	Bridge_2	64	Node_9	Bridge_2_to_Node_9_SYNC_SC	1.74E-06	SYNC
4	Bridge_4	64	Node_10	Bridge_4_to_Node_10_SYNC_SC	1.74E-06	SYNC
5	Bridge_4	64	Node_11	Bridge_4_to_Node_11_SYNC_SC	1.74E-06	SYNC
6	Bridge_4	64	Node_12	Bridge_4_to_Node_12_SYNC_SC	1.74E-06	SYNC
7	Node_1	64	Node_8	N1_to_N8_TT1_1	1.02E-05	TT1
8	Node_1A	64	Node_8	N1A_to_N8_TT1_1	1.87E-05	TT1
9	Node_1	64	Node_8	N1_to_N8_TT1_1	1.02E-05	TT1
10	Node_1A	64	Node_8	N1A_to_N8_TT1_1	6.19E-05	TT1

Figure 9.9: Data produced where no fault is added on Network Model 2

This table shows a sample of the raw data traffic extracted from the simulation. This is a completely wired network setup. It also shows that Bridge_2 and Bridge_4 are initiating the broadcast of sync frames. The outcome is the same as Network Model 1 where the latency remains the same as there is no fault injected. Thus, the anomaly detection model does not detect any frame breaching the latency thresholds; hence, it does not produce a csv file containing flagged anomalies as no sync frame breaches the global or local maximum latency thresholds.

Network Model 3

Network model 3 is the largest network model designed to test the performance of the proposed security solution. Below is a sample of the traffic generated by running the simulator on Network Model 3 where no fault is injected.

	Task_Source	Task_Size	Task_Destination	Legend	Latency	Identifier
2	Bridge_4	64	Node_10	Bridge_4_to_Node_10_SYNC_SC	1.74E-06	SYNC
3	Bridge_4	64	Node_11	Bridge_4_to_Node_11_SYNC_SC	1.74E-06	SYNC
4	Bridge_4	64	Node_12	Bridge_4_to_Node_12_SYNC_SC	1.74E-06	SYNC
5	Bridge_2	64	Node_8	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	SYNC
6	Bridge_2	64	Node_9	Bridge_2_to_Node_9_SYNC_SC	1.74E-06	SYNC
7	Node_1A	64	Node_8	N1A_to_N8_TT1_1	1.87E-05	TT1
8	Node_1	64	Node_8	N1_to_N8_TT1_1	1.02E-05	TT1
9	Node_1	64	Node_8	N1_to_N8_TT1_1	6.19E-05	TT1
10	Node_1A	64	Node_8	N1A_to_N8_TT1_1	1.87E-05	TT1

Figure 9.10: Data produced from simulation where no fault is injected on Network Model 3

This is the same as the first two simulations run on network models 1 and 2 in that the Latency remains the same since no fault is injected. The table also shows that simulating the network model produces all types of traffic. The rule-based anomaly detection, however, does filter for the sync frames only as one of the rules set out demands that only sync frames are to be extracted and analysed. Thus, as in the first two simulations, no frames are flagged for anomalous latency as the current latency is equivalent to the default 'latency before fault'.

Therefore, it can safely be said that, regardless of the size or complexity of the network model used, no anomalous frames can be detected if a fault injector is not applied to the network. This would be the same in practical network deployment unless there exists a security breach which this program would nullify if it delayed the sync frames by more

Network Model	Sim Time	Latency Before Fault	Added Latency	Integration Technique	Fault Injector	Medium of Communication	Flagged Anomalies
Network Model 1	0.1sec	1.74E-06	0	Timely	None	Wired	None
Network Model 2	0.1sec	1.74E-06	0	Timely	None	Wired	None
Network Mode3 1	0.1sec	1.74E-06	0	Timely	None	Wired	None

Table 9.1: A summary of simulations run where no fault injector is applied

than the set acceptable thresholds.

The three simulations carried out on the three network models where no fault is injected can be summarized in the table below.

This table proves that if no faults are added or there is no security breach targeting the latency of sync frames - in practical network deployment, the rule-based anomaly detection model would not flag sync frames for an anomaly.

9.2.4.2 Running Simulation on the Wired LAN where Fault is Injected for a Local Breach

The next batch of simulations shows the potential where a bad actor can intentionally delay sync frames but within the global maximum latency threshold. The maximum acceptable latency for TTEthernet clock synchronization is usually set in the range of 40 to 50 nanoseconds. Consequently, a malicious attacker would make sure the delay of sync frames does not cross this threshold to avoid getting caught by security measures designed to monitor the latency of sync frames. Therefore, the anomaly detection model has included a rule to counter this scenario by setting another latency threshold at 30 nanoseconds where a breach of this threshold triggers a timer. If the timer counts for 0.01 seconds, which is equivalent to 10 cycles of synchronization, and the end system does not correct its clock's drift to less than 30 nanoseconds, the frames arriving at this end system are flagged and are referred to as anomalies flagged by 'local' breach. It should be noted that this is subjective; hence, the local threshold can be set up to higher or lower than 30 nanoseconds and that counter can also be decreased or increased to reflect the latency requirement of the specific network deployment.

Network Model 1

Network model 1 is a good example to represent a simple but complex enough network deployment as it encompasses both the wired and wireless segments of an IIoT. A limitation, which is presented in the results discussion section below, makes it clear that it is not possible to have a wireless and wired configuration within the same network model in visualsim. Thus, every end system, wired or wireless, is treated as a wired device in this simulation as the traffic configuration is set for a wired medium of communication. Similarly, the WAP is treated as another network switch in this simulation. This research tries to address the importance of the proposed security solution for practical network environments. Thus, the current setup of this network model should work with both the wired and wireless segments of an IIoT in a practical network scenario. Below is a data sample extracted by simulating network model 1 after a fault injector is applied to stay within the maximum latency threshold of 50 nanoseconds.

	Task_Source	Task_Size	Task_Destination	Legend	Latency	Iden
2	Bridge_2	64	Node_6	Bridge_2_to_Node_6_SYNC_SC	1.75E-06	SYNC
3	Bridge_2	64	Node_7	Bridge_2_to_Node_7_SYNC_SC	1.76E-06	SYNC
4	Bridge_3	64	Node_8	Bridge_3_to_Node_8_SYNC_SC	1.79E-06	SYNC
5	Node_1	64	Node_6	N1_to_N6_TT1_1	1.02E-05	TT1
6	Node_1A	64	Node_6	N1A_to_N6_TT1_1	1.87E-05	TT1
7	Node_1	64	Node_6	N1_to_N6_TT1_1	1.92E-05	TT1
8	Node_1	64	Node_6	N1_to_N6_TT1_1	1.02E-05	TT1
9	Node_1A	64	Node_6	N1A_to_N6_TT1_1	6.19E-05	TT1
10	Node_1A	64	Node_6	N1A_to_N6_TT1_1	1.87E-05	TT1

Figure 9.11: Data produced where fault injector is applied on network model 1

Latency is slightly higher than the default value for a wired medium of communication which is set at 1.74E-06 for sync frames. If the difference between the latency that is displayed in the table above and the default latency value is less than the maximum latency threshold of 50 nanoseconds but bigger than 30 nanoseconds, the rule outlined for the local breach is triggered. If it is less than 30 nanoseconds, however, it is considered a benign sync frame. As shown in the previous examples, the dataset includes all traffic types. A point to note is that for practical network deployment, Bridge 3 would be a WAP connecting wireless devices to the local LAN. However, it is being used as another network switch in this example because the communication channel is configured for a wired connection. Similarly, ES8 is one more wired device in this example which would

be a wireless device in practical network deployment. The anomaly detection program is designed with a practical network deployment in mind. In a practical network, there is no need to configure a device for a wireless or wired connection as they have clear 'latency before fault' values used to determine 'added latency' and this anomaly detection model would work seamlessly.

The anomaly detection program uses the above dataset as an input to execute the rules set out in the program. The simulation that produced the above dataset also produces another csv file, as below, which displays the outcome of the rule-based model.

	Legend	Latency Before Fault	Task_Source	Task_Size	Identifier	Task_Destination	Added Latency	Current Latency	Flagged By
2									
3	Bridge_3_to_Node_8_SYNC_SC	1.74E-06	Bridge_3	64	SYNC	Node_8	4.51E-08	1.79E-06	Local
4									
5	Bridge_2_to_Node_7_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_7	3.80E-08	1.78E-06	Local
6									
7	Bridge_1_to_Node_6_SYNC_SC	1.74E-06	Bridge_1	64	SYNC	Node_6	4.10E-08	1.78E-06	Local
8									
9	Bridge_2_to_Node_7_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_7	4.08E-08	1.78E-06	Local
10									
11	Bridge_2_to_Node_7_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_7	4.35E-08	1.78E-06	Local
12									
13	Bridge_2_to_Node_7_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_7	4.33E-08	1.78E-06	Local
14									
15	Bridge_3_to_Node_8_SYNC_SC	1.74E-06	Bridge_3	64	SYNC	Node_8	3.80E-08	1.78E-06	Local

Figure 9.12: Result from the rule-based model on network model 1

The figure above shows that the model adds new columns which have values for 'latency before fault', 'Added Latency' and 'Flagged by'. Furthermore, the term 'Latency' from the input csv file is converted to 'Current latency' by the model to give it more clarity.

'Latency before Fault' is the default latency value which is the expected latency for synchronization frames before a security breach adds the latency value.

'Added latency' is a new value calculated by the rule-based model. It is calculated by subtracting the 'latency before fault' value from the 'current latency' value. The rules related to the maximum threshold are dependent on this value. Looking at the values in the figure above, all the added latency values are between 38 nanoseconds and 45 nanoseconds which lies between the global latency threshold and the local latency threshold; hence, they are labelled as 'local' under the 'flagged by' column.

'Flagged By' is the last column in the dataset added by the rule-based model. It is used to identify if a frame is flagged by breaching the latency threshold at 50 nanoseconds or 30 nanoseconds. The resulting steps taken to resolve such an issue may depend on this

value.

So, it can be said that this simulation has produced an expected result where faults are injected to stay within the maximum acceptable threshold of 50 nanoseconds. Consequently, sync frames that have crossed the 30 nanoseconds threshold but not the 50 nanoseconds threshold are flagged and have been labelled as 'local' breaches; although, some of these frames are not flagged because the value of added latency did not cross the 30 nanoseconds threshold. In a practical scenario, some of these frames would have corrected themselves; hence, they were not flagged as anomalies.

Network Model 2

Data extracted by running Network Model 2 is expected to be similar to Network Model 1 as a dataset because the only difference is the increase in network size. The dataset displayed below is filtered to show only the sync frames. However, the input csv file used by the rule-based model is not filtered and contains all types of traffic because the program itself has a filter of its own.

	Task_Source	Task_Size	Task_Destination	Legend	Latency	Identifier
2	Bridge_2	64	Node_8	Bridge_2_to_Node_8_SYNC_SC	1.77E-06	SYNC
3	Bridge_2	64	Node_9	Bridge_2_to_Node_9_SYNC_SC	1.77E-06	SYNC
4	Bridge_4	64	Node_10	Bridge_4_to_Node_10_SYNC_SC	1.75E-06	SYNC
5	Bridge_4	64	Node_11	Bridge_4_to_Node_11_SYNC_SC	1.74E-06	SYNC
6	Bridge_4	64	Node_12	Bridge_4_to_Node_12_SYNC_SC	1.78E-06	SYNC
862	Bridge_2	64	Node_8	Bridge_2_to_Node_8_SYNC_SC	1.77E-06	SYNC
863	Bridge_2	64	Node_9	Bridge_2_to_Node_9_SYNC_SC	1.78E-06	SYNC
864	Bridge_4	64	Node_10	Bridge_4_to_Node_10_SYNC_SC	1.75E-06	SYNC
865	Bridge_4	64	Node_11	Bridge_4_to_Node_11_SYNC_SC	1.77E-06	SYNC
866	Bridge_4	64	Node_12	Bridge_4_to_Node_12_SYNC_SC	1.77E-06	SYNC

Figure 9.13: Data produced where fault injector is applied on network model 2

Looking at the latency of sync frames, in the figure above, it is evident that not all sync frames breach the local latency threshold because the fault injector is not applied to all end systems. So, the program only detects sync frames and writes them into a csv file if they have crossed the local latency threshold. Another observation is that most of the sync frames are coming from Bridge_4 or Bridge_2. It has been discussed above that TTEthernet is known for redundancy and the use of multiple routes to communicate traffic. As such, nodes 8 and 9 are physically as well as virtually connected to Bridges 1 and 2 while end systems 10-12 are connected to Bridges 3 and 4. Thus, it is the case that traffic that is delivered first is used, dropping all other follow-up duplicate frames. In this scenario, Bridge_2 and Bridge_4 seem to have an advantage over the other bridges in the

cluster as is evident that traffic emanating from them is delivered first at the receiving end systems.

The result of the model which uses the above csv file as data input to execute all the rules relevant to the project is displayed below.

	Legend	Latency Before Fault	Task_Source	Task_Size	Identifier	Task_Destination	Added Latency	Current Latency	Flagged By
2									
3	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_8	3.13E-08	1.77E-06	Local
4									
5	Bridge_4_to_Node_12_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_12	4.01E-08	1.78E-06	Local
6									
7	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_8	3.00E-08	1.77E-06	Local
8									
9	Bridge_2_to_Node_9_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_9	4.25E-08	1.78E-06	Local
10									
11	Bridge_4_to_Node_12_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_12	3.31E-08	1.77E-06	Local
12									
13	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_8	4.58E-08	1.79E-06	Local
14									
15	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_8	4.03E-08	1.78E-06	Local
16									
17	Bridge_4_to_Node_10_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_10	4.64E-08	1.79E-06	Local
18									
19	Bridge_4_to_Node_11_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_11	4.31E-08	1.78E-06	Local
20									
21	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_8	3.95E-08	1.78E-06	Local

Figure 9.14: Result from the rule-based on network model 2

As explained in network model 1 above, the values under the ‘Flagged by’ column say ‘local’ and the values under the ‘added latency’ correspond to that by displaying values between 30 and 50 nanoseconds. So, this outcome is not different to the one collected by simulating network model 1. This in turn explains that the size of a network does not matter in a TTEthernet clock synchronization as all small clusters can also be synchronized within themselves.

Network Model 3

This is the largest network model of the three models used in this report. However, as the results show, there is no difference in the outcome as every cluster is synchronized locally by configuring enough end systems as SMs and the switches as CMs.

	Task_Source	Task_Size	Task_Destination	Legend	Latency	Identifier
2	Bridge_4	64	Node_10	Bridge_4_to_Node_10_SYNC_SC	3.92E-06	SYNC
3	Bridge_4	64	Node_11	Bridge_4_to_Node_11_SYNC_SC	2.05E-06	SYNC
4	Bridge_4	64	Node_12	Bridge_4_to_Node_12_SYNC_SC	2.00E-06	SYNC
5	Bridge_2	64	Node_8	Bridge_2_to_Node_8_SYNC_SC	4.49E-06	SYNC
6	Bridge_2	64	Node_9	Bridge_2_to_Node_9_SYNC_SC	4.16E-06	SYNC
7	Node_1A	64	Node_8	N1A_to_N8_TT1_1	1.87E-05	TT1
8	Node_1	64	Node_8	N1_to_N8_TT1_1	1.02E-05	TT1
9	Node_1	64	Node_8	N1_to_N8_TT1_1	6.19E-05	TT1
10	Node_1A	64	Node_8	N1A_to_N8_TT1_1	1.87E-05	TT1

Figure 9.15: Data extracted by running simulation on Network Model 3

This picture shows the type of data extracted and written into a csv file to serve as input for the main anomaly detection model to manipulate and execute specific rules relevant to the study. The values under the Latency column show that some have crossed the local latency threshold but not the global threshold. Similarly, some sync frames have a higher latency value than the default 'latency before fault' but are not high enough to breach the local latency threshold.

The figure below displays the outcome of running the simulation and using the extracted traffic dataset to execute the model to see how a comparatively larger and more complex network model uses the security solution proposed effectively. As expected, the outcome is similar to the above two outcomes from network model 1 and network model 2.

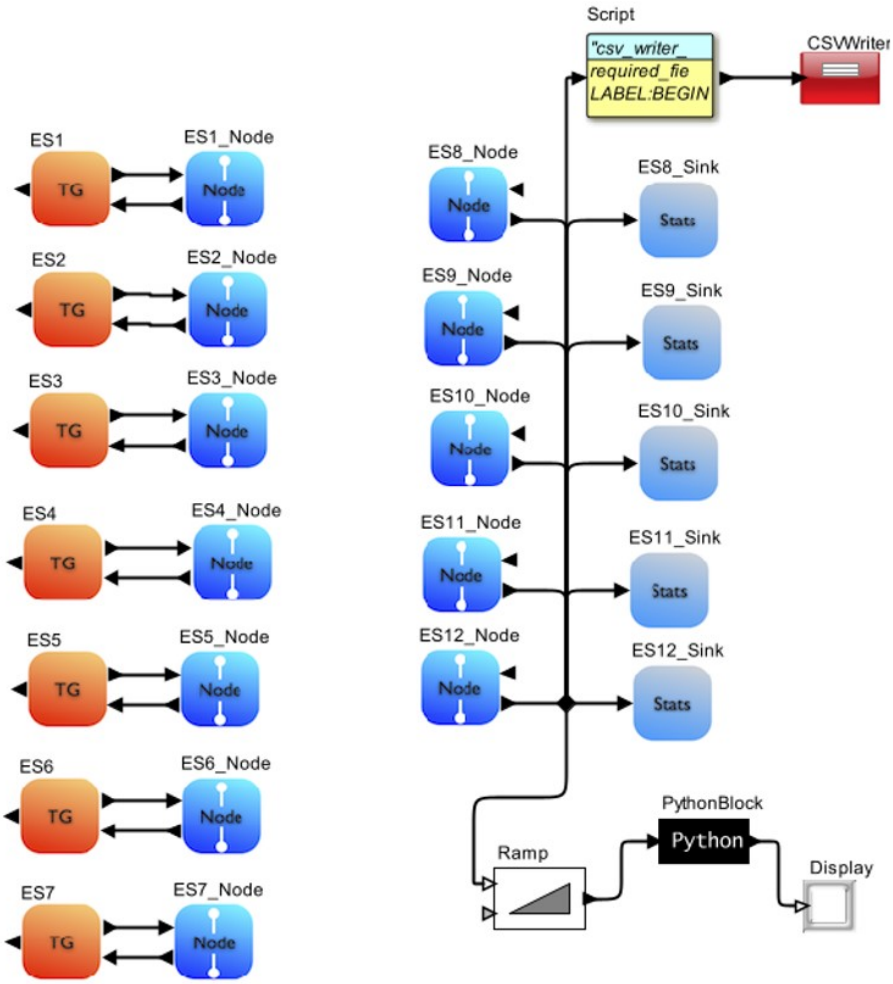

Figure 9.16: Result from the rule-based model on network model 3

The result is expected that all sync frames where the added latency values have breached the 30 nanoseconds, local maximum latency threshold, are added to the 'flagged

Network Model	Sim Time	Latency Before Fault	Added Latency	Integration Technique	Fault Injector	Medium of Communication	Flagged Anomalies
Network Model 1	0.1sec	1.74E-06	30<X<50	Timely	Below MT	Wired	local
Network Model 2	0.1sec	1.74E-06	30<X<50	Timely	Below MT	Wired	local
Network Model 3	0.1sec	1.74E-06	30<X<50	Timely	Below MT	Wired	local

Table 9.2: A summary of the simulations run with local breach

by' column with the value 'local' to suggest that they are anomalies added by breaching the local latency threshold. This has proved that the size of a network is not a factor in TTEthernet clock synchronization.

Finally, the outcome of the above three simulations confirms the proposed solution as they delivered the expected results. If a fault injector is applied to a certain end system, with the intention of delaying sync frames to within the acceptable maximum latency threshold of 30 nanoseconds, it is added to a list and is monitored. If it still doesn't correct its clock within the 10 synchronization cycles, it is added to the list of anomalous sync frames. This has been the case that sync frames are flagged as anomalies by a 'local' breach.

These outcomes can be summarized in table 9.2.

9.2.4.3 Running Simulation on the Wired LAN where Fault is Injected for Global Breach

The global latency threshold is commonly used in some network types, other than TTEthernet networks. Traffic is usually set aside or added to a certain quarantine zone if it crosses the set maximum latency threshold so it can be verified or resoled subjectively by respective latency solutions. It is not different in this study except that there has not been this specific type of security solution in relation to the latency-related security breaches, as of the researcher's current knowledge base, for TTEthernet clock synchronization. The following simulations are representative examples of how the proposed security solution deals with latency breaches that cross the global maximum latency threshold of 50 nanoseconds.

Network Model 1

Network model 1, as mentioned above is designed to validate the security solution in a small IIoT. CSV files extracted from running the simulation on this network model are shown below. Similar to the simulations run above, the simulation on this network model has produced raw data of all types of traffic communicated between end systems in the network. This dataset is normally used as an input for the rule-based anomaly detection model to execute the rules set out for this research.

	Task_Source	Task_Size	Task_Destination	Legend	Latency	Identifier
2	Bridge_2	64	Node_6	Bridge_2_to_Node_6_SYNC_SC	2.18E-06	SYNC
3	Bridge_2	64	Node_7	Bridge_2_to_Node_7_SYNC_SC	2.34E-06	SYNC
4	Bridge_3	64	Node_8	Bridge_3_to_Node_8_SYNC_SC	3.05E-06	SYNC
5	Node_1	64	Node_6	N1_to_N6_TT1_1	1.02E-05	TT1
6	Node_1A	64	Node_6	N1A_to_N6_TT1_1	1.87E-05	TT1
7	Node_1	64	Node_6	N1_to_N6_TT1_1	1.92E-05	TT1
8	Node_1	64	Node_6	N1_to_N6_TT1_1	1.02E-05	TT1
9	Node_1A	64	Node_6	N1A_to_N6_TT1_1	6.19E-05	TT1
10	Node_1A	64	Node_6	N1A_to_N6_TT1_1	1.87E-05	TT1

Figure 9.17: Data extracted by Simulating Network Model 1

This is the first 10 rows of the dataset produced by running the simulation tool on network model 1. The displayed table shows only sync frames and the TT frames because they sit at the top of the csv file. Otherwise, all traffic types including the RC and BE, Ethernet traffic are included in the csv file. It can also be seen that the sync frames are coming from Bridge_2 and Bridge_3. It should be noted that Bridges 1 and 2 are used within the same cluster. Bridge_3, however, is designed as a WAP to connect wireless devices to the local LAN. For this specific simulation, however, Bridge_3 is just another network switch as the traffic configuration is set for wired communication. Hence, end system 8 is also one more wired device for this simulation for the same reason. It should be noted that this model is designed with practical network deployment scenarios in mind.

The values under the ‘Latency’ column are usually way above the maximum latency threshold. This is used to calculate the ‘added latency’ as displayed below which is important for the model to detect a sync frame if it has breached the set maximum thresholds or not.

Legend	Latency Before Fault	Task_Source	Task_Size	Identifier	Task_Destination	Added Latency	Current Latency	Flagged By
Bridge_2_to_Node_6_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_6	4.41E-07	2.18E-06	Global
Bridge_2_to_Node_7_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_7	6.00E-07	2.34E-06	Global
Bridge_3_to_Node_8_SYNC_SC	1.74E-06	Bridge_3	64	SYNC	Node_8	1.31E-06	3.05E-06	Global
Bridge_2_to_Node_7_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_7	1.80E-07	1.92E-06	Global
Bridge_1_to_Node_6_SYNC_SC	1.74E-06	Bridge_1	64	SYNC	Node_6	3.44E-06	5.18E-06	Global
Bridge_3_to_Node_8_SYNC_SC	1.74E-06	Bridge_3	64	SYNC	Node_8	3.49E-07	2.09E-06	Global
Bridge_2_to_Node_6_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_6	7.57E-07	2.50E-06	Global
Bridge_2_to_Node_7_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_7	4.82E-07	2.22E-06	Global
Bridge_3_to_Node_8_SYNC_SC	1.74E-06	Bridge_3	64	SYNC	Node_8	3.54E-07	2.09E-06	Global
Bridge_2_to_Node_6_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_6	1.64E-06	3.38E-06	Global

Figure 9.18: Result from the rule-based model on network model 1

The default value for sync frame latency before a security breach, malicious or otherwise, is set to 1.74E-06 as mentioned above. This value and the 'current latency' are used to calculate the added latency. The table above shows that 'added latency' is above the 50 nanoseconds threshold; hence, they are labelled as being 'flagged by' global breaches, which is another way of saying that these sync frames have breached the global maximum latency threshold which is 50 nanoseconds. The values under the 'added latency' which shows that they are way above the 50 nanoseconds are also confirmed by another rule which states that if a sync frame is flagged by breaching the 'global' threshold the value' global' is entered under the 'flagged by' header.

Network Model 2

This is a network model designed for a complete wired connection as it sometimes is the case in a practical network deployment. It has been mentioned throughout the paper that TTEthernet is designed for wired c communication. So, this network model is a typical TTEthernet network which uses a wired medium of communication, although there is no issue getting seamless transportation of traffic to and from the wireless segment in an IIoT environment. Below is a table showing a few rows from a big dataset produced by running a simulation tool on network model 1, where an injected fault breaches the maximum acceptable latency threshold.

	Task_Source	Task_Size	Task_Destination	Legend	Latency	Identifier
2	Bridge_2	64	Node_8	Bridge_2_to_Node_8_SYNC_SC	1.26E-06	SYNC
3	Bridge_2	64	Node_9	Bridge_2_to_Node_9_SYNC_SC	1.85E-06	SYNC
4	Bridge_4	64	Node_10	Bridge_4_to_Node_10_SYNC_SC	1.15E-06	SYNC
5	Bridge_4	64	Node_11	Bridge_4_to_Node_11_SYNC_SC	9.96E-07	SYNC
6	Bridge_4	64	Node_12	Bridge_4_to_Node_12_SYNC_SC	1.01E-06	SYNC
7	Node_1	64	Node_8	N1_to_N8_TT1_1	9.03E-06	TT1
8	Node_1A	64	Node_8	N1A_to_N8_TT1_1	1.75E-05	TT1
9	Node_1	64	Node_8	N1_to_N8_TT1_1	1.80E-05	TT1
10	Node_1	64	Node_8	N1_to_N8_TT1_1	9.26E-06	TT1

Figure 9.19: Data extracted by simulating Network Model 2

The figure above shows that synchronization traffic is broadcasted from Bridge_2 and Bridge_4. As explained above, these bridges have proved to be the best routes to get traffic to the destination end systems. This dataset is then used as an input to produce the below dataset by applying the rule-based model.

	Legend	Latency Before Fault	Task_Source	Task_Size	Identifier	Task_Destination	Added Latency	Current Latency	Flagged By
2									
3	Bridge_2_to_Node_9_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_9	1.05E-07	1.85E-06	Global
4									
5	Bridge_4_to_Node_12_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_12	2.97E-06	4.71E-06	Global
6									
7	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_8	1.62E-07	1.90E-06	Global
8									
9	Bridge_2_to_Node_9_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_9	4.72E-07	2.21E-06	Global
10									
11	Bridge_4_to_Node_12_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_12	1.43E-06	3.17E-06	Global
12									
13	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_8	1.88E-06	3.62E-06	Global
14									
15	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_8	1.33E-06	3.07E-06	Global
16									
17	Bridge_4_to_Node_11_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_11	1.47E-06	3.21E-06	Global
18									
19	Bridge_4_to_Node_12_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_12	2.37E-06	4.11E-06	Global
20									
21	Bridge_2_to_Node_9_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_9	1.36E-07	1.88E-06	Global

Figure 9.20: Result from the rule-based model on network model 2

It is clear to see that ‘added latency’ has a big range of values all above the maximum latency threshold. The rule says that if the latency of a sync frame is more than the maximum acceptable threshold, it gets flagged and is labelled that it is flagged for breaching the global threshold. So, however big or small the latency, if it breaches the acceptable threshold, it is flagged and then it is dealt with by a fitting correction measure. As expected, only sync frames are added to the dataset and all entries are flagged by global security breach as is backed up by the values under the ‘added latency’ column.

Network Model 3

Finally, network model 3 is used to test the validity of the proposed security solution by running the rule-based anomaly detection model on a relatively large IIoT network, by

applying a fault injector on the receiving end systems. In a practical network deployment, this would represent many wired and wireless end systems connected using a host of network switches and WAPs. For this simulation, however, every device is configured to communicate on a wired medium of communication. Thus, data extracted from this simulation is used for the model to execute rules.

	Task_Source	Task_Size	Task_Destination	Legend	Latency	Identifier
2	Bridge_4	64	Node_10	Bridge_4_to_Node_10_SYNC_SC	3.92E-06	SYNC
3	Bridge_4	64	Node_11	Bridge_4_to_Node_11_SYNC_SC	2.05E-06	SYNC
4	Bridge_4	64	Node_12	Bridge_4_to_Node_12_SYNC_SC	2.00E-06	SYNC
5	Bridge_2	64	Node_8	Bridge_2_to_Node_8_SYNC_SC	4.49E-06	SYNC
6	Bridge_2	64	Node_9	Bridge_2_to_Node_9_SYNC_SC	4.16E-06	SYNC
7	Node_1A	64	Node_8	N1A_to_N8_TT1_1	1.87E-05	TT1
8	Node_1	64	Node_8	N1_to_N8_TT1_1	1.02E-05	TT1
9	Node_1	64	Node_8	N1_to_N8_TT1_1	6.19E-05	TT1
10	Node_1A	64	Node_8	N1A_to_N8_TT1_1	1.87E-05	TT1

Figure 9.21: Data extracted by running simulation on Network Model 3

Fig 9.21 shows the first few lines of a big dataset produced by running the visualsim tool on network model 3. A fault injector is used to increase the latency of sync frames to breach the global maximum latency threshold of 50 nanoseconds. It can also be observed that different traffic types are contained within the network. There are still RC and Ethernet traffic included in the dataset. So, this is the raw data traffic communicated within the network; which is also used as an input for the anomaly detection model to exploit.

	Legend	Latency Before Fault	Task_Source	Task_Size	Identifier	Task_Destination	Added Latency	Current Latency	Flagged By
2									
3	Bridge_4_to_Node_10_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_10	2.18E-06	3.92E-06	Global
4									
5	Bridge_4_to_Node_11_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_11	3.10E-07	2.05E-06	Global
6									
7	Bridge_4_to_Node_12_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_12	2.63E-07	2.00E-06	Global
8									
9	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_8	2.75E-06	4.49E-06	Global
10									
11	Bridge_2_to_Node_9_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_9	2.42E-06	4.16E-06	Global
12									
13	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_8	1.60E-06	3.34E-06	Global
14									
15	Bridge_2_to_Node_9_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_9	8.91E-07	2.63E-06	Global
16									
17	Bridge_4_to_Node_10_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_10	4.47E-07	2.19E-06	Global
18									
19	Bridge_4_to_Node_11_SYNC_SC	1.74E-06	Bridge_4	64	SYNC	Node_11	9.82E-07	2.72E-06	Global
20									
21	Bridge_2_to_Node_8_SYNC_SC	1.74E-06	Bridge_2	64	SYNC	Node_8	2.85E-07	2.03E-06	Global

Figure 9.22: Result from the rule-based model on network model 3

It is shown that ‘latency before fault’ is imported as the ideal value for a wired connection at 1.74E-06 for the sync frames. The current latency, which helps to calculate

Network Model	Sim Time	Latency Before Fault	Added Latency	Integration Technique	Fault Injector	Medium of Communication	Flagged Anomalies
Network Model 1	0.1sec	1.74E-06	X>50 Timely	Timely	Above MT	Wired	Global
Network Model 2	0.1sec	1.74E-06	X>50	Timely	Above MT	Wired	Global
Network Model 3	0.1sec	1.74E-06	X>50	Timely	Above MT	Wired	Global

Table 9.3: A summary of simulations carried out on all network models

the added latency, reflects the latency a sync frame takes from the 'task source' to the 'task destination'. It is also shown that values under the 'added latency' exceed the maximum latency threshold at 50 nanoseconds. This is confirmed by the 'global' values populated under the 'flagged by' header.

So, it is clear to see that regardless of the network size, the proposed solution works effectively in all TTEthernet clock synchronization deployments. This is expected to work effectively in a practical network environment as well.

A summary of the simulations carried out on three of the network models above, where injected fault breaches the global maximum acceptable latency threshold, is presented in the table below.

9.2.4.4 Validating the Proposed Security Solution on the Wireless Section of an IIoT

It has been discussed above that TTEthernet is designed for the wired network connection although it is still used in an IIoT environment. Hence, it is imperative that there is a seamless flow of traffic between the wired end system and the wireless section of the IIoT network. The main reason the wireless segment is not beneficiary of the advantages TTEthernet offers is that there is no option for redundancy. Wireless devices can only connect to a single WAP at a time. Hence, there is a single point of failure. If a WAP fails, all wireless devices connected to it also fail. Nonetheless, the anomaly detection model proposed here as a solution also works on the wireless segment of an IIoT. The rule-based security solution mainly focuses on anomaly detection by setting rules that define an acceptable latency threshold for synchronization traffic. Thus, traffic arriving at

individual wireless devices can also be interrogated for latency and categorized as benign or anomalous depending on respecting the acceptable maximum latency threshold. The screenshots below are examples of how valid the proposed solution is in wireless systems.

Network Model 1

Network model 1 is a simple form of IIoT. It has just one wireless device connected to the LAN using a WAP. In this section, simulations are run where fault injector is not applied; where fault injector is applied to remain under the maximum latency threshold; and where fault injector is applied to breach the maximum acceptable latency threshold at 50 nanoseconds. The fault injector is applied on the communication channel that leads to end system 8, which is the only wireless device in the network. Below is a sample of the first few lines of the raw data of all types of traffic being communicated within the network by simulating network model 1.

	Task_Source	Task_Size	Task_Destination	Legend	Latency	Identifier
2	Bridge_3	64	Node_8	Bridge_3_to_Node_8_SYNC_SC	2.79E-06	SYNC
3	Bridge_3	64	Node_8	Bridge_3_to_Node_8_SYNC_SC	2.81E-06	SYNC
4	Node_2	128	Node_8	N2_to_N8_TT2_1	9.07E-05	TT2
5	Node_2A	128	Node_8	N2A_to_N8_TT2_1	1.59E-04	TT2
6	Node_2	128	Node_8	N2_to_N8_TT2_1	5.76E-05	TT2
7	Node_2A	128	Node_8	N2A_to_N8_TT2_1	1.25E-04	TT2
8	Node_2	128	Node_8	N2_to_N8_TT2_1	9.07E-05	TT2
9	Node_2A	128	Node_8	N2A_to_N8_TT2_1	1.59E-04	TT2
10	Node_2	128	Node_8	N2_to_N8_TT2_1	5.76E-05	TT2

Figure 9.23: Data extracted by running simulation on Network Model 1

Fig 9.23 shows data produced by running a simulation on network model 1 where a fault injector is applied on end system 8 to breach the global maximum latency threshold. Referring to the sync frames at the top of the table, there is only one ‘task source’ and one ‘task destination’. This is because there is only one wireless device in ES8 which is linked up to the local LAN using the only WAP present in the network which is Bridge_3. This dataset is an input for the rule-based anomaly detection model to execute some important rules designed to maximize the efficiency of the proposed security solution. Below is an example of the resulting outcome from running the model on the above csv file.

	Legend	Latency Before Fault	Task_Source	Task_Size	Identifier	Task_Destination	Added Latency	Current Latency	Flagged By
2									
3	Bridge_3_to_Node_8_SYNC_SC	5.32E-07	Bridge_3	64	SYNC	Node_8	2.26E-06	2.79E-06	Global
4									
5	Bridge_3_to_Node_8_SYNC_SC	5.32E-07	Bridge_3	64	SYNC	Node_8	2.28E-06	2.81E-06	Global
6									
7	Bridge_3_to_Node_8_SYNC_SC	5.32E-07	Bridge_3	64	SYNC	Node_8	2.97E-06	3.50E-06	Global
8									
9	Bridge_3_to_Node_8_SYNC_SC	5.32E-07	Bridge_3	64	SYNC	Node_8	4.55E-07	9.87E-07	Global
10									
11	Bridge_3_to_Node_8_SYNC_SC	5.32E-07	Bridge_3	64	SYNC	Node_8	4.17E-07	9.49E-07	Global

Figure 9.24: Result from the rule-based model where the fault injector breaches the maximum latency threshold.

Fig 9.24 shows that 'latency before fault' is set to 5.32E-07 for sync frames on a wireless medium of communication. This value is produced by changing the propagation constant which is 0.83 for the wired twisted pair cable connection to 50.24 to reflect the communication on a wireless medium as shown in the picture below.

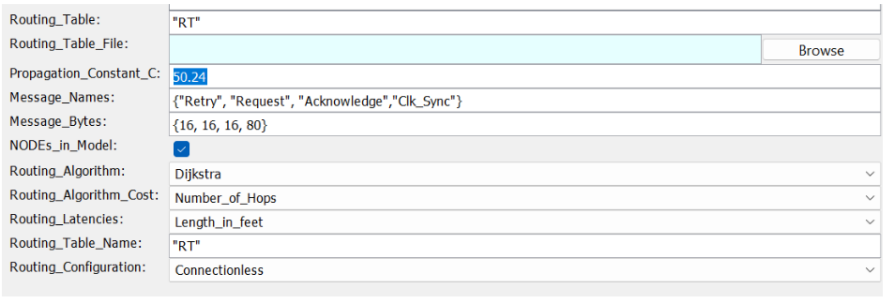

Figure 9.25: Propagation constant set to 50.24 for wireless communication

The propagation constant is where the value is adjusted to reflect the communication medium in visualsim. Note that this would not be needed in a practical network where data is produced from physical devices communicating using networking devices. Steps taken to determine these values are presented in Chapter 7. Figure 9.24 is the result of a simulation run on network model 1. It shows that values under the 'added latency' go beyond the maximum latency threshold and it is backed up by the entries in the 'flagged by' column where they all show global breach. Network model 1 is designed with one wireless device connected to the LAN using a WAP to show it is an IIoT. The csv writer and rule-based anomaly detection model are both applied to Node_8, which is also called end system 8 (ES8), to exploit sync traffic arriving at the wireless end system which is fed by the only WAP in the network. This does not mean that there needs to be one WAP

for every wireless device but if two wireless devices are too far apart from each other that a single WAP cannot cover, it becomes important that another WAP is deployed to connect all wireless devices to the LAN. The WAP is configured as a synchronization client; hence, it cannot broadcast sync frames but can relay one. Thus, the actual 'task source' cannot be Bridge_3. It has to be one of the network switches which are configured as CMs. In visualsim, the route a sync frame took including all hops used can be seen by using the 'listen to port' functionality.

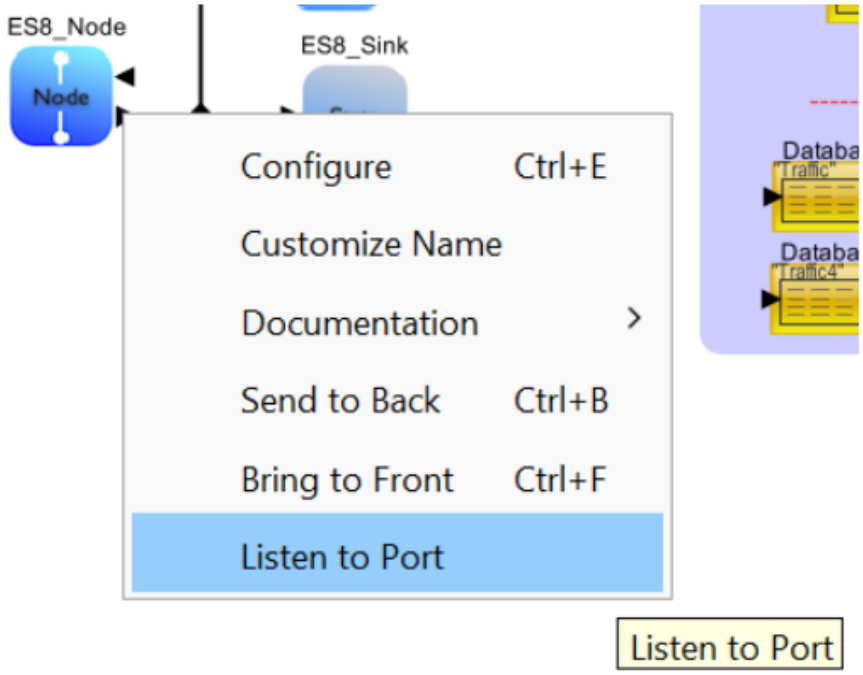

Figure 9.26: Shows how the 'listen to port' functions

Fig 9.26 produces traffic including how they are routed, among other details and it can be found by opening the 'listening to port while running the simulation. This helps to investigate the specific culprit in a chain of hops any traffic travels across a network. It should be noted that the security solution proposed in this research focuses on detecting anomalous sync frames and flagging them for the attention of the security team who can then make use of the 'listen to port' to further explore what might have caused the unexpected latency.

The time a sync frame takes until it gets to the destination node can also be found by listening to the same port. It's displayed in the dataset produced after running the

model above that all entries under the header ‘added latency’ cross the maximum latency threshold because a fault injector is applied to the wireless device. These values are confirmed by entries under the ‘flagged by’ column, which are all labelled as ‘global’ to signify that these specific sync frames are delayed by more than the maximum latency threshold of 50 nanoseconds.

Network Model 3

Network model 3 is the second network model designed to carry out the performance analysis of the proposed solution on the wireless segment of an IIoT. For ease of analysis, both clusters, containing the wireless devices, are put together so the csv writer and rule-based anomaly detection model can be applied to all the receiving wireless devices at the same time and collect data in one csv file to feed the model for further application of important rules, as follows.

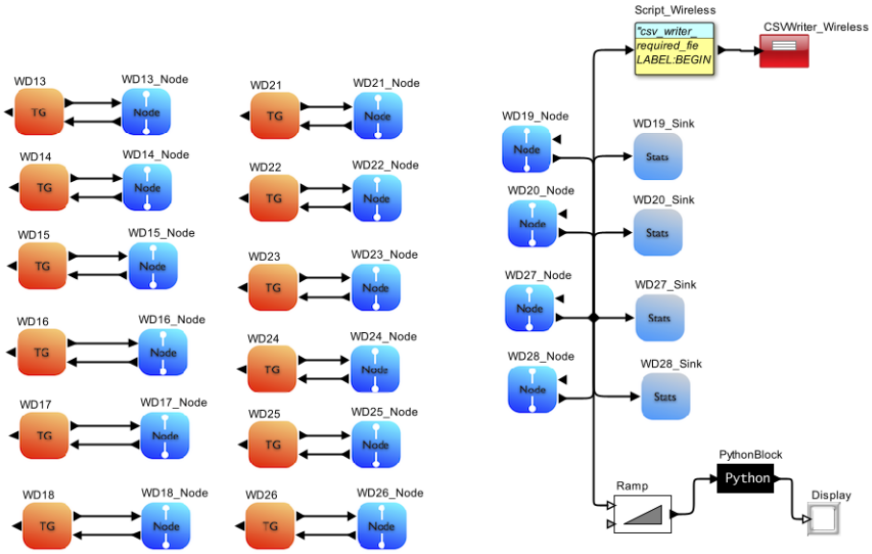

Figure 9.27: Wireless segment of an IIoT setup for simulation

Network model 3, as defined above, is designed to represent an IIoT with three clusters, including one cluster of wired devices connected using network switches and two clusters of wireless connected to the LAN using a WAP each. For this simulation, nonetheless, both wireless clusters are combined so a simulation can be run to analyze traffic arriving at the four receiving end systems. Node 19 is configured with no fault injector applied. Node 20 and Node 27 are configured with faults injected to the effect of a local breach. This is where the latency of sync frames does not exceed 50

nanoseconds but are flagged for breaching the 30 nanoseconds latency threshold and not correcting themselves after 0.01 seconds of synchronization period which is equivalent to 10 synchronization cycles. Lastly, Node 28 is configured with fault injected to the effect of breaching the maximum latency threshold of 50 nanoseconds; hence, flagged for global breach. These different configurations of fault injectors are performed in one simulation to avoid repetition of analysis.

Simulating the above network design has produced a csv file which contains all types of traffic communicated within the network as follows.

	Task_Source	Task_Size	Task_Destination	Legend	Latency	Identifie
2	Bridge_5	64	Node_19	Bridge_5_to_Node_19_SYNC_SC	5.32E-07	SYNC
3	Bridge_5	64	Node_20	Bridge_5_to_Node_20_SYNC_SC	5.75E-07	SYNC
4	Bridge_6	64	Node_27	Bridge_6_to_Node_27_SYNC_SC	5.52E-07	SYNC
5	Bridge_6	64	Node_28	Bridge_6_to_Node_28_SYNC_SC	8.74E-07	SYNC
6	Bridge_6	64	Node_27	Bridge_6_to_Node_27_SYNC_SC	5.40E-07	SYNC
7	Bridge_6	64	Node_28	Bridge_6_to_Node_28_SYNC_SC	8.59E-07	SYNC
8	Bridge_5	64	Node_19	Bridge_5_to_Node_19_SYNC_SC	5.32E-07	SYNC
9	Bridge_5	64	Node_20	Bridge_5_to_Node_20_SYNC_SC	5.46E-07	SYNC
10	Node_13	64	Node_19	N13_to_N19_RC13_2	3.06E-06	RC13
11	Node_14	64	Node_20	N14_to_N20_RC14_2	3.06E-06	RC14
12	Node_16	64	Node_19	N16_to_N19_RC16_2	4.06E-06	RC16

Figure 9.28: Data extracted by simulating the wireless segment of network model 3

Referring to traffic routing, it shows that there are two WAPs used as ‘task sources’ Bridge_5 and Bridge_6. The two WAPs are designed to connect two groups of wireless devices to the LAN. Latency shows values that reflect the configuration of the fault injector on the receiving wireless devices. Node_19 shows the default latency value for a wireless medium of communication because no fault was injected. Node_20 and Node_27 show bigger latencies than the ideal default value. It also shows that both wireless devices have different latencies but are in the same range, nonetheless. Node_28 has a much bigger latency as compared to the other nodes. This dataset is used as an input for the model to produce the results as shown below.

	Legend	Latency Before Fault	Task_Source	Task_Size	Identifier	Task_Destination	Added Latency	Current Latency	Flagged By
1									
2	Bridge_5_to_Node_20_SYNC_SC	5.32E-07	Bridge_5	64	SYNC	Node_20	4.32E-08	5.75E-07	Local
3									
4	Bridge_6_to_Node_28_SYNC_SC	5.32E-07	Bridge_6	64	SYNC	Node_28	3.42E-07	8.74E-07	Global
5									
6	Bridge_6_to_Node_28_SYNC_SC	5.32E-07	Bridge_6	64	SYNC	Node_28	3.27E-07	8.59E-07	Global
7									
8	Bridge_6_to_Node_28_SYNC_SC	5.32E-07	Bridge_6	64	SYNC	Node_28	2.04E-06	2.57E-06	Global
9									
10	Bridge_6_to_Node_28_SYNC_SC	5.32E-07	Bridge_6	64	SYNC	Node_28	7.68E-08	6.09E-07	Global
11									
12	Bridge_6_to_Node_27_SYNC_SC	5.32E-07	Bridge_6	64	SYNC	Node_27	4.72E-08	5.79E-07	Local
13									
14	Bridge_6_to_Node_28_SYNC_SC	5.32E-07	Bridge_6	64	SYNC	Node_28	1.53E-06	2.06E-06	Global
15									
16	Bridge_6_to_Node_27_SYNC_SC	5.32E-07	Bridge_6	64	SYNC	Node_27	4.14E-08	5.73E-07	Local
17									
18	Bridge_6_to_Node_28_SYNC_SC	5.32E-07	Bridge_6	64	SYNC	Node_28	2.44E-06	2.97E-06	Global
19									

Figure 9.29: Result from the rule-based anomaly detection model where the network setup includes various levels of faults injected

Figure 9.29 shows that the default 'latency before fault' value is set to 5.32E-07. Looking at the 'task destination', no anomalous frames are flagged for Node_19. This can be explained by the fact that there is no fault injected at that specific end system. It is also evident that M there are more global security breaches than local breaches although only Node_28 was configured for the global breach as compared to Node_20 and Node_27 being configured for local breaches. This is because not all faults injected for local breaches cross the 30-second threshold. Furthermore, it is shown that the 'added latency' values correspond to the entries under the 'flagged by' column.

This chapter started by designing the network models required to evaluate the security solution. Those network models are configured for the simulation tool and testing protocols are clearly explained in detail to help analyse the results of the simulation. Finally, simulations are run on all the network models and prove that the security solution works as expected. It can be concluded that the proposed solution can be deployed in any TTEthernet-based IIoT and help resolve latency-related issues for the TTEthernet clock synchronization. It has been proved that although the wireless segment of an IIoT is not equipped with options for redundancy and other TTEthernet-related benefits, the proposed solution can still be deployed in an IIoT and both the wired and wireless sections can be monitored for latency-related security breaches the same way. Further interpreting and conceptualizing of the simulation results are presented in the next chapter.

Chapter 10

Interpreting and Contextualizing the Proposed Security Solution

10.1 Overview of the Proposed Security Solution

This research started with clear aims and detailed objectives focused on addressing specific problem scenarios. So, the main aim and objectives must be briefly restated here so the proposed security solutions can be seen relative to the objectives designed.

The ultimate goal of this research is to provide security guarantees for clock synchronization in the interconnected safety-critical IIoT systems using TTEthernet. It is important to note that the objectives of this research are designed to build up to the grand aim of the study. Hence, they can be considered building blocks of the main aim rather than independent research objectives. Below is a summary of how these objectives have been achieved and how those steps have helped realize the grand aim of the research project.

1. **Investigating the TTEthernet-based IIoT applications, their processes and usage:**

This is a bit generic as it is designed to build the basics of what TTEthernet is and how it is used in IIoT applications, to make use of the benefits it offers. It should be noted that all the objectives are seen from the security point of view as this research is about the security of TTEthernet clock synchronization. The background to TTEthernet and applications that make use of its services are discussed in Chapter 1 and it is further

developed in the subsequent chapters; mainly Chapter 3. These chapters covered the background of TTEthernet and applications benefiting from it. Furthermore, the security issues associated with TTEthernet clock synchronization and the provisional solution to those vulnerabilities are discussed.

Therefore, this objective is met as expected as it aspired to investigate the basics of what and how TTEthernet clock synchronization works. This contributes to the design of the core security algorithm as it is important to understand the background knowledge base to be able to design a fitting security solution.

2. **Exploring the importance of clock synchronization and its security breaches in TTEthernet-based IIoT:**

This objective is focused on exploring the importance of clock synchronization and security breaches related to it in a TTEthernet-based IIoT environment. This is a continuation of the first objective with the focus shifted to the security of synchronization frames communicated within a given network deployment. Different clock synchronization protocols are compared to establish the importance of TTEthernet clock synchronization. The study proved that no other clock synchronization protocol is better that satisfies most if not all of the benefits including but not limited to determinism, an ideal solution for real-time applications by employing static offline scheduling technique to guarantee the delivery of messages on a specific time frame; fault tolerance which is a typical character because TTEthernet uses multiple SMs and CMs which consequently offers the option for multiple routes for traffic communication; Integration with Ethernet which is the main trait as it is fully compatible with Ethernet with the capacity of transmitting real-time and non-real-time traffic on the same network backbone; and Scalability which implies the fact that TTEthernet clock synchronization can be adaptable on complex wired and wireless segments of a network as proved in this research.

The major security threats in TTEthernet clock synchronization are explored and summarized as node insertion, node compromising, message deletion, message insertion, and message manipulation. These are the techniques a malicious attacker could use to compromise the security of a TTEthernet clock synchronization and cause further security

threats including latency of synchronization frames, among others.

Thus, it can be said that the second objective was successfully achieved as it explored the importance of clock synchronization and security threats within the TTEthernet clock synchronization. This objective is important as it outlines the security vulnerabilities necessary to understand and design a security solution for.

3. **Developing a security algorithm for securing clock synchronization in TTEthernet-based IIoT:**

This objective promises the provision of a security solution to vulnerabilities identified within the TTEthernet clock synchronization. It has already been established that one of the main security threats in the TTEthernet clock synchronization in the IIoT is the latency of synchronization frames. The problem identification process is carried out in the preceding chapters 2 and 3. A rule-based anomaly detection model has been developed to address the latency-related security breaches within the TTEthernet clock synchronization. The algorithm sets some rules important to identify synchronization frames delayed by more than an accepted maximum latency threshold and flags them so network practitioners can take necessary security measures to resolve the issue. The model includes two acceptable maximum latency thresholds designed to nullify sync frames breaching latency thresholds at both levels. The first latency threshold refers to a sync frame delayed by more than the 'global' latency threshold at 50 nanoseconds. The second latency threshold is set to monitor a malicious attacker who may decide to send a calculated latency that would delay sync frames but within the widely known acceptable global latency threshold of 50 nanoseconds, within the TTEthernet paradigm, for TTEthernet clock synchronization. The rule relating to the second latency threshold adds sync frames that breach the set latency threshold at 30 nanoseconds to a separate list to monitor and keeps them for 0.01 seconds which is equivalent to 10 synchronization cycles. If the sync frames added to the list do not correct their latency to be under the second latency threshold, they are flagged for anomaly and added to another list containing sync frames that have breached any of the rules set in the fault detection model.

Therefore, developing a security algorithm for securing clock synchronization in TTEthernet-based IIoT, which is equivalent to the main aim of the study has successfully

been accomplished. It should be noted that it was tested in a simulation environment which is designed to mimic practical network deployments of IIoT nature. The core algorithm is designed with all sizes and complexities of practical network deployments in mind; hence, it is expected to equally work in any network environment perpetually, and internationally.

4. **Evaluating and testing the proposed approach on a realistic use case:**

As pointed out in the research methodology, chapter 4, evaluation and testing of the proposed approach was designed for a realistic, practical TTEthernet-based network deployment. Nonetheless, an alternative simulation and modelling tool was used as the practical lab was found to be malfunctioning. Therefore, Visualsim which is widely used across the aviation and industrial automation sectors, among others, is used to validate the proposed security solution. The proposed security solution is directly linked to the test network models. It pulls data from the network in real-time, during the simulation, analyses the data, and prepares a database of its own with the right format before it applies the rules to identify sync frames which have breached the acceptable latency thresholds. Finally, it flags sync frames found to have breached any of the latency thresholds.

Thus, the evaluation and testing steps of the proposed solution proved that the security solution forwarded can resolve latency-related security issues within the TTEthernet clock synchronization in the IIoT.

10.2 Interpretation of Findings

Every step of the proposed security solution is interpreted here in detail about how it addresses the research objectives. Note that the last two research objectives target the design of the security solution and the steps required to verify it as a security solution as the first two build the essential knowledge base required to advance the research. It should also be noted that the test results are interpreted from a practical network deployment point of view as well as the designed network models. The results of the tests carried out to validate the proposed solution have shown that whatever the network size, the results remained similar both on the wired and wireless segments of the IIoT. Thus, it is not essential to discuss the results for every single simulation carried out on all the network

scenarios. It is, however, important to differentiate the results on the wired and wireless clusters carried out on the test environments.

10.2.1 Test Results from Simulations Run on the Wired LAN

Simulations carried out on all network scenarios (wired and wireless), where a fault injector is not applied, proved that no sync frame is flagged for an anomaly. This is the basis of the validation process where sync frames are not expected to be flagged for anomalies in normal traffic communication where they are not delayed by more than the set latency thresholds. Note that the proposed solution is set with a default 'latency before fault' value, from which 'added latency' is derived. However, this value may be different among different network deployments as they have varying latencies observed. Furthermore, network practitioners may apply different techniques to establish the 'latency before fault' value which, in turn, produces different values. Generally, it is obtained by taking the average latency from a controlled network where no added latency is observed from malicious or benign factors over an extended period.

Thus, the proposed solution stood tall at this early stage of the validation process as results showed the expected outcome, producing no flagged anomalies as the fault injector was not applied.

The second batch of simulations focused on the addition of a fault injector with the effect of breaching the global maximum latency threshold. The fault injector is designed so that the simulations can produce sync frames with latency higher than the set maximum latency threshold at 50 nanoseconds. Hence, the rule-based anomaly detection model rightly flagged all frames as having breached the rule set out for the maximum latency threshold. As explained in the model, the added latency is calculated by subtracting the 'latency before fault' from the current latency. 'Latency before fault' is set to different values for wired and wireless communication mediums, as outlined in Chapter 7, during simulation. 'Latency before fault' for sync frames is set to 1.74E-06 on the wired and 5.32E-07 on the wireless communication mediums. The steps used to find these values are presented in Chapter 7 above. A limitation present in visualsim, which is presented below, in this chapter, has made it difficult to add different values for the wired and

wireless mediums of communication in one configuration block. Nonetheless, the model is designed with practical network deployments in mind as it includes a rule which differentiates wired and wireless traffic. The rule specifies that if sync frames come from a WAP, the 'latency before fault' value that is used to calculate the added latency is 5.32E-07 and if the sync frame comes from a wired device, the 'Latency before fault' would be 1.74-06. So, it can be argued that this model is suited better for the practical network environment as compared to the simulation environment.

This batch of simulations proved the viability of the proposed security solution further. Sync frames breaching the maximum latency threshold are flagged in real-time. For this simulation, it was possible to link the model to the simulation tool so sync frames could be monitored in real-time and flagged if they were found to have breached the latency thresholds. Furthermore, it has come across as if it would work better on the practical network deployment as it is easier to set different values of 'latency before fault' depending on the 'source of traffic' or the networking device sending the sync frame that is not a possibility in the simulation network scenarios.

The third batch of simulations targeted to monitor sync frames that do not cross the 50 nanoseconds maximum global latency threshold. It is not documented anywhere but talking to researchers in the field, it could be gathered that the generally accepted maximum latency threshold for TTEthernet clock synchronization is in the range of 40-50 nanoseconds. This is because no research focused on setting maximum thresholds for TTEthernet clock synchronization which is also the driving force for this research work. It should be noted that this could also be knowledge shared with a malicious attacker who may want to use it for his or her bad acts. This may not cause instant damage as breaching the global latency threshold would, but it is possible to degrade the network performance and reliability among others, as stated in Chapter 2. The latency threshold used to monitor sync frame latency that manages to stay under 50 nanoseconds is set to 30 nanoseconds. Synchronization frames flagged for an anomaly in these batches of simulations breach two rules although they remain within the global maximum latency threshold. For ease of analysis, this is named as 'local' latency threshold. The first rule is that sync frames crossing the local latency threshold are added to a list and kept for

0.01 seconds, equivalent to 10 synchronization cycles. If the sync frame's latency does not re-align to less than the local threshold, it is added to the flagged sync frames list for breaching the local latency threshold. The outcome of the simulations run on the three network models has proved that the model still works as expected. Hence, it has proved its viability once again.

In a practical network scenario, the values used for the global and local latency thresholds can be modified depending on the network's latency requirement. Otherwise, the model should work as it is and produce similar results.

10.2.2 Implications of the Wired and Wireless Communication

The proposed solution is designed to work on both the wired and wireless segments of the IIoT setup. It has been re-iterated in this research multiple times that TTEthernet is designed for the wired medium of communication. This means that the wireless section of an IIoT would not get the benefits TTEthernet offers; mainly, redundancy, fault tolerance, and other advantages that come with it. Nonetheless, TTEthernet is still predominantly used in Industrial environments where wireless sensors are the main components of the network. Therefore, the proposed security solution must address seamless communication in the IIoT.

Simulation has been carried out on the different wireless network scenarios and the result showed that the model works in the wireless segment in the same way it did in the wired medium of communication.

10.3 Proposed Solution Against Existing Literature

Different viewpoints have been analysed in the literature review section. This has led to understanding the main gap in knowledge leading to outlining the research methodology and working to achieve the required security solution. Hence, this section highlights how the security solution offered in this research resonates with the arguments analysed in the literature review and resolves the knowledge gap identified.

The main requirement of a TTEthernet synchronization stems from the lack of a synchronous and asynchronous transmission of traffic on the same network backbone;

where strict scheduling of traffic is upheld. The closest solution that has been offered where all three types of traffic; mainly the TT, RC, and BE traffic can be communicated, with a more flexible scheduling algorithm for the efficient use of bandwidth and other resources, is the Time Sensitive Networking (TSN). However, it depends on a single Grandmaster for clock synchronization, with a single point of failure. TTEthernet synchronization, however, offers multiple SMs and CMs to address the issue of redundancy and fault tolerance and related benefits associated with it. Thus, TTEthernet clock synchronization has an advantage over the other solutions offered for clock synchronization in the IIoT.

The intra-cluster clock synchronization method is specified in the SAE AS6802 standard which this research is based on. It is an important term that has been mentioned multiple times in this research to indicate that the size of a network is not a hindrance in TTEthernet clock synchronization as intra-cluster synchronization is possible leaving the inter-cluster synchronization unaffected which is also stressed in (Tang et al., 2018).

As an extension to Ethernet communication, TTEthernet was designed to enhance traffic transmission on the wired LAN. Thus, it was suggested in (Peón et al., 2014) that an extension to the wireless link of the IIoT needs to be addressed. This project accepts this as a challenge for future research and makes sure the security solution proposed here addresses the wireless as well as the wired sections of the IIoT.

(Choi et al., 2018) accepts that previous studies focused on the general security requirements and standardization of security policies and procedures instead of technical solutions and not many have been on clock synchronization. In a TTEthernet clock synchronization, the CM broadcasts sync frames to all member nodes using the UDP protocol which is less reliable for the delivery on the receiving end. The use of fault-tolerant, redundant TTEthernet clock synchronization helps to partly address the reliability of sync frame delivery. This research takes this approach on board and considers different latency thresholds and sending error signals where sync frames are not delivered in the acceptable time frame. Sync frames delayed by more than the set acceptable latency thresholds are flagged for breaching the rules.

Different attack types have been studied about the TTEthernet clock synchronization;

among which, latency is a popular concern. Some of the attacks targeting the clock synchronization in IIoT include but are not limited to the ASN (Absolute Slot Number) which targets the specific time frame allotted to individual nodes, (SeqNum) sequence number attacks, node replication attacks, global time attacks and traitor attacks which all need addressing with a security solution. In one way or another, all these attack types contribute to the delay of sync frames, which probably is the main aim of some of those attacks. Practical examples of security attacks on the timing signals have been presented (Shepard et al., 2012) & (Lévesque and Tipper, 2016). Synchronization attacks are practical and severe and can bypass security measures if no detection method is in place, (Smache et al., 2019). They proposed a machine learning algorithm to detect clock synchronization attacks. Nonetheless, they did not disclose their algorithm, nor did they mention the specific type of attack the machine learning algorithm would detect. Thus, the importance of having a specific latency-focused sync frame detection algorithm and flagging those anomalous frames is designed in this research. The damage delayed sync frames can cause in a network has also been discussed in the literature review. These include the damage to the reliability and network performance which includes the interruption of operations and unintended behaviour of industrial applications, (Ullmann and Vögeler, 2009) as well as performance degradation and high risks in safety-critical applications, (Lei et al., 2022).

This project considered all the pros and cons of the articles explored on the subject matter. It identified the latency of synchronization frames, instigated by malicious attackers or otherwise, as a security threat. A fitting security solution is designed in the form of a rule-based anomaly detection system that identifies sync frames breaching the set maximum latency thresholds. So, the security solution has addressed one of the many gaps in knowledge in the subject and has offered a new dimension for further research.

10.4 Implications of the Proposed Security Solution

The fault detection model proposed in this research is novel and far-reaching. It is accepted that it only addresses the latency-related security threat, however majorly important it is. The implications of the solution proposed can be seen from the theoretical

and practical points of view.

The theoretical aspect is that it lays the precedence for a more comprehensive security solution that addresses all security threats on the TTEthernet clock synchronization as well as the IIoT in general. The model can also be considered for setting threshold-related algorithms for various scenarios. Thus, it is an exciting step into the rule-based security solutions to the TTEthernet clock synchronization and beyond.

In practical terms, this model can be used without too many modifications to the code. There are specific attributes subjective to particular deployment environments. For example, the maximum latency threshold values can be adjusted depending on the latency requirement of the network deployment used. Names of nodes and network switches used can also be modified. Generally, certain attributes need to be adjusted to represent a specific network deployed in a particular environment. Nonetheless, the general purpose of the model can be used by a practitioner with just a few changes.

10.5 Conclusion

This is an essential chapter in the project as it relates the proposed security solution to the research objectives and the build-up of the project. It has shown the coherence of the research work from the start. It started by re-iterating the research aim and objectives before it briefly presented the proposed security solution relative to the objectives. It has also recapped an important aspect of the research methods to show how a practical TTEthernet lab could not be used because the setup was found to be malfunctioning at the time of enquiry. All the simulations run to test the proposed solution are briefly explained and justify the viability of the security solution proposed both on the wired and wireless segments of IIoT TTEthernet networks.

The proposed solution has also been explored in the arguments presented in the literature review chapter. This has helped place the knowledge gained through this research work among the existing knowledge base on the subject matter. Finally, it is also interrogated for its viability in the theoretical research area and how practical practitioners could make use of the proposed solution in practical terms. Practical limitations imposed and future works relevant to this chapter are presented in the next chapter 11.2 as part of

the overall limitations and future work identified for the project.

Thus, this innovative method effectively proves that the security solution tackles the vulnerability associated with sync frame latency, a primary technique used in communication channel attacks. Moreover, test results across various network environments underscore its validity as a security solution, aligning with the specified aims and research objectives.

Chapter 11

Overall Conclusion

This section concludes the research work by summarizing the highlights of the steps taken to get to this point. The limitations hindering smooth progress as well as some important topics identified for future work are also covered. Finally, the final reflection is presented in a short statement to give the main takeaway of the research work.

11.1 Summary

The research work started with an ambitious but attainable goal. TTEthernet being a relatively emerging technology, some questions need addressing, regarding the security of traffic communicated within the TTEthernet clock synchronization. However, not enough research has been done to answer many of these questions. The main aim of this research is centred around the security guarantees required for the TTEthernet clock synchronization in the safety-critical IIoT environment. This is further broken down into smaller, more manageable, research objectives which include: exploring the TTEthernet protocol including the IIoT applications that use it; investigating the importance of TTEthernet clock synchronization and the security vulnerabilities associated with the TTEthernet-based IIoT; developing security solution algorithms to secure the TTEthernet clock synchronization in the IIoT; and, finally, validating the proposed security solution using network models designed using the simulation tool.

The initial steps involved laying down the basic exploration of IIoT in general terms. The safety and security implications in the IIoT applications and the integration of OT/IT

and security vulnerabilities associated with them are investigated. Deep-dive research is carried out on TTEthernet and how different types of traffic can be communicated on the same network backbone with strict scheduling techniques. TT traffic gets priority over event-triggered ones in the TTEthernet scheduling. IIoT, TTEthernet, and clock synchronization are investigated in detail which is followed by the TTEthernet-based IIoT applications where different use cases are discussed. Finally, the flow of investigation led to the main topic of the security of TTEthernet clock synchronization in the IIoT. The security threats that exist within the TTEthernet-based IIoT are critical from the safety and security point of view. A delay in the delivery of traffic by a fraction of a second can cause hazardous situations. Thus, the research focused on the security of TTEthernet clock synchronization. This is important because clock synchronization forms the basis for the secure follow-up of traffic communication between individual network devices.

The choice of TTEthernet for clock synchronization is analysed against alternative clock synchronization protocols and it proved that TTEthernet clock synchronization offers determinism for hard real-time traffic and delivery of sync frames using multiple routes. This is mainly because it supports multiple SMs and CMs helpful for redundancy purposes where a failed end system doesn't significantly affect traffic communication. Therefore, TTEthernet clock synchronization has distinctive benefits that other protocols cannot offer.

Three scheduling techniques mainly Pre-emptive, Timely-block, and Shuffling are explored for their pros and cons as a scheduling method. Pre-emptive and Timely-block use different scheduling algorithms but they give priority for sync frames to be communicated first while shuffling goes against the general essence of TTEthernet as it works on a 'first come first served' basis. Thus, Timely-block which blocks other traffic types until PCFs and other TT traffic are communicated in full, is used as the scheduling technique in this research.

The literature review presented relevant research carried out concerning IIoT, TTEthernet, and most importantly, the security of TTEthernet clock synchronization in an IIoT. This helped to identify a gap in knowledge, about the security vulnerabilities present within the TTEthernet clock synchronization. Alternative clock synchronization

protocols have been suggested but none with the benefits TTEthernet offers which makes it the most appropriate for certain network deployment types whose required QoS includes the highest form of determinism and fault tolerance, among others. Similarly, research has been carried out focusing on the latency of sync frames in a TTEthernet clock synchronization but fell short of using latency thresholds or related artificial intelligence models as a security solution. Hence, this opportunity presented itself for further investigation to find a solution to the latency of sync frames which ultimately affects the smooth communication of the follow-up network traffic.

A threat model was designed to identify the system assets and their attack surface, vulnerabilities within the assets, adversary goals, attacker types, and security objectives. These in turn motivate the need to find a security solution. This is further explored by focusing on two commonly known security breaches that target the TTEthernet clock synchronization, mainly the MitM attack and the delay attack. Different scenarios are investigated to understand how these attack types could breach the security of an IIoT through the TTEthernet clock synchronization and potential security solutions are outlined hypothetically.

This was followed by the experimental work using a network modelling and simulation tool, the visualsim. Some example configurations are displayed to show how the network models are configured and to what effect. A fault injector is used to mimic security breaches in specific situations. Three configuration scenarios, namely: '**None**', where there is no security breach, '**Sync_Drift_Below_Threshold** where injected fault delays sync frames by less than the global maximum latency threshold', and '**Sync_Drift_Spread**', where injected fault breaches the set maximum latency threshold, are set out for the fault injector. These three configuration scenarios are designed to show the differences where a security breach happens and the levels of the breach so a fitting security solution can be outlined.

The network topology used for the network model is described as two-star topologies centred around the two network switches and a potential third with just one wireless end system connected to the WAP. The third topology can also be described as a one-to-one connection but in an actual deployment it is likely that there would be more than one

wireless device, hence, it becomes another star topology.

The network models are designed to represent the wired and wireless segments of a TTEthernet-based IIoT. Visualsim has a default value for the twisted pair cables used in a LAN and some alternative cable types but not for the wireless medium of communication. Nonetheless, the configuration block in Visualsim allows for a value to be added to represent the wired and wireless mediums of communication. Thus, a value for the propagation constant that represents the wireless medium of communication is calculated. It is noted that these calculations would not be needed in a real-life network deployment as practical wireless channels are used to communicate traffic to and from the wireless devices to the control centre.

Simulations are run after the configurations are defined to reflect real-life network deployments. This research is more focused on the added latency than the latency itself. Sync frames like any other traffic can be delayed for different reasons. Nonetheless, this research is focused on establishing the ideal latency and keeping a record of the acceptable latency for every communication channel. A mechanism is then devised to calculate the latency on top of the ideally acceptable latency recorded. This way, added latency is calculated to check the presence of a security breach in a network. Simulation results show differences in latency depending on the level of fault injected if any. The level of added latency differs depending on the level of fault injected. Ideal latency is registered where no fault is injected. Similarly, the global and local latency thresholds are breached when the right level of fault is injected to mimic the security breaches that take place in practical network deployments.

The proposed security solution is presented in pseudocode for clarity and simplicity. It is composed of a set of rules that are executed when a simulation is run on the network models. It is noted that the rules, mainly the latency thresholds, are subjective as they can always be modified according to the latency requirements of a network deployment.

The validation stage involved the design of three network models with various sizes and complexities to test the proposed security solution in different network environments. It has been proved that the security solution works on every network environment, and the results are similar regardless of the size and complexity of the network deployment.

It is also applicable in the wireless segment of an IIoT. The rules for added latency are the same although there are different values for the 'latency before fault' which helps to determine the 'added latency'.

This has addressed one of the main security concerns about the security vulnerabilities within the TTEthernet clock synchronization which is the latency of sync frames. This is a solution that has opened a new window for further research where similar techniques can be used to resolve similar issues. From the practical point of view, with small modifications relevant to each network deployment, network practitioners can deploy the proposed security solution and reap the benefits. It should be noted that the model is designed with practical network deployment in mind; hence, it should work in real-life network deployments with minimal modifications to the model.

Finally, it is pointed out that the rule-based anomaly detection model can be optimized by dynamically updating the latency thresholds which would benefit from incorporating a machine learning method to learn about changes caused by external factors or changes within the network itself and update the latency thresholds accordingly. Nonetheless, it should be stressed that the rule-based anomaly detection model has produced the expected result. So, it is safe to say that the proposed security solution can be used in all network environments where TTEthernet clock synchronization is employed.

11.2 Limitations and Future Research

There are some limitations imposed on the progress of this project. So, the main limitations and the future research work that could be done to address the limitations or add another dimension to the research work are outlined below.

11.2.1 Limitations

The methodology section in this research lays out that the primary plan to test the validity of the model was to use the TTEthernet lab setup at Cranfield University. However, it was not possible to use it at the time of enquiry. An alternative option was considered to go to the University of Siegen, in Germany after a positive discussion with Dr. Daniel Onwuchekwa, who oversaw the TTEthernet lab at the university, but commitment to work

and financial constraints meant that it was not materialized either. Hence, the backup plan where a good modelling and simulation tool in Visualsim had to be used. Thus, the main limitations stem from the fact that the proposed security solution could not be validated using a practical network deployment.

Consequently, limitations that come with the simulation tool have dragged the progress of the research; hence, they had to be accepted as unavoidable limitations. One of the limitations imposed by the simulation tool is the fact that there is only one configuration block in visualsim that can be configured for wired or wireless mediums of communication. Hence, it was not possible to configure a network model for the wired and wireless segments of an IIoT at the same time. Thus, it is not an ideal representation of an IIoT network environment. Nonetheless, the proposed security solution is designed to work in real-world network deployment where TTEthernet clock synchronization is used; so, it should work with some modifications to the script.

The inability to replicate the steps used to set a value for the 'latency before fault' in the practical deployments as it is in the simulation tool is another limitation. 'Latency before fault' is the default latency before traffic is subjected to any kind of security breach that adds to the natural latency. This may be different from the actual value that practical network deployments observe. This value is found by simply running the simulation where the fault injector is not used. Nonetheless, some effort is required in the practical network scenarios to establish the 'latency before fault' value. It is recommended that this is done using a controlled network where no additional factors are involved to increase or decrease the natural latency. It should be noted that different network deployments have different latency requirements. Thus, the value for the 'latency before fault' should be extracted from a network that it is going to be used for. This is an important value to find the added latency which is an important value in the rule-based model to decide if sync frames should be deemed anomalous for breaching the set latency thresholds.

Another important limitation is that all end systems, switches and traffic blocks are manually configured making it difficult to design a larger network model. Hence, the chance to design a TTEthernet-based network model with a bigger size and complexity has been hampered due to the amount of time and resources it consumes.

On a similar note, running simulations on the aforementioned network models proved to be too much for a laptop with small RAM and processing resources; hence, the simulation tool had to be installed on a laptop which had better resources just so the simulations could be run.

Finally, since this is an emerging technology, there are not many publicly available resources to explore existing knowledge bases. This makes it difficult to get different perspectives on the subject matter.

11.2.2 Future research

This research project has opened new possibilities which need to be explored further. A few of them are discussed below.

- It is mentioned that machine learning models can be used to optimize the accuracy of the rule-based anomaly detection model. Hence, integrating machine learning techniques into the security solution offered in this research is an important next step considered for future work.

- The evaluation process proves that the proposed security solution can be applied to any TTEthernet-based IIoT deployment, anywhere, and enhance the security of traffic communication by securing the clock synchronization protocol. Nonetheless, starting from the very beginning of the research work, the ambition was to test the proposed security solution in a practical network deployment. However, it was not possible to do this for practical reasons, as the only TTEthernet lab in the UK, at Cranfield University, was found to be malfunctioning. Thus, testing the security solution in a practical lab would add the credence it deserves; hence, testing the proposed security solution on a practical IIoT deployment remains to be one of the main future work plans.

- The researcher believes that the anomaly detection model used for this research can be modified to work for regular traffic communication other than clock synchronization. Furthermore, this can also be modified to work in non-TTEthernet-based networks. So, this is an opportunity that presents itself for further study; hence, it is considered for future work.

- It is understood that latency is not the only security threat to the TTEthernet clock

synchronization. Thus, further research is considered to understand other vulnerabilities within the clock synchronization and study methods of addressing them. Thus, exploring further for vulnerabilities other than the latency of sync frames and investigating security methods is a research that is considered for future work.

11.3 Final Thoughts

This research has proposed a noble security solution for the TTEthernet clock synchronization in IIoT by addressing security vulnerabilities related to the latency of synchronization frames. This is an option researchers and network practitioners can consider to enhance their knowledge base and practically deploy it to secure their IIoT deployment. It is important to note, however, that latency is only one of many security threats facing TTEthernet clock synchronization and IIoT in general; so, a bigger perspective needs to be considered for end-to-end security. The anomaly detection model used is scalable as it can be adapted to different topologies and network complexities and it can, potentially, be modified to serve similar roles for other clock synchronization protocols or general traffic communication in a given network.

Appendix A

Bridge Controller Configuration

The following is a scripting language presenting the bridge controller configuration. Network switches and WAPs are configured as bridges in this configuration. The configuration sets out the connection between different end systems and other bridges within the network. Bridges are an important component in TTEthernet as most of them are configured as CMs to collect PCFs from SMs before they pass the aggregated PCFs to all devices directly connected to them. Bridge controller configurations also include data routing and filtering, traffic management and error handling among others.

```
1
2
3  /* Bridge Controller          */
4  $
5  setup_ds                                          = {DS_Name = "Node_Setup",Node_
6  None_Hash                     = ("none").hashCode()
7  SEND("Network_Setup",setup_ds)
8  flag_once                     = true
9  syncMasterDS                  = {DS_Name = "Sync_Master_DS"}
10 Warning_Flag                  = ("WARNING").check()
11 Node_Name_Hash                = Node_Name.hashCode()
12 Hop_Arr                       = {}
13 Clock_Sync_Arr                = {}
14 Destination_Arr               = {}
15 Port_Arr                      = {"none","none","none","none","none","none","none"
```

```
16 max_out_ports                              = Port_Arr.length() //16
17 idArrThreshold                             = 20
18 idArrPruneLength                           = 10
19 Node_Flag                        = false
20 if (!Block_Reference.contains("Bridge_")) {
21    GTO  (Generate_Port_Mapping)
22    throwMyException (Block_Reference + " expect type of Node: Bridge_N, \n\nFounc
23 }
24
25 LABEL: Generate_Port_Mapping
26    WAIT (TResolution * 4)
27    ThisBroadcast               = ("Multicast_N").read()
28    ThisBroadcastIdx            = ThisBroadcast.search(Node_Name)
29    if (ThisBroadcastIdx      >= 0)                 {
30       ThisBroadcast            = ThisBroadcast.removeElement(ThisBroadcastIdx)
31    }
32    ThisBroadcast_Len           = ThisBroadcast.length()
33    Port_Arr                                       = newArray(ThisBroadcast_Len,"r
34    Idx                         = 0
35    while (Idx                  < ThisBroadcast_Len) {
36       Destination              = ThisBroadcast(Idx)
37       Hop                      = getRoutingTableHop(RoutingTable, Node_Name, Desti
38       Hop_Arr                  = Hop_Arr.append(Hop)
39       Destination_Arr          = Destination_Arr.append(Destination)
40       ++Idx
41    }
42    Idx                         = 0
43    Idx3                        = 1
44    while (Idx                  < ThisBroadcast_Len) {
45       Hop                      = Hop_Arr(Idx)
46       Port                     = "dn_" + (Idx3)
47       Idx2                     = 0
48       while (Idx2              < ThisBroadcast_Len) {
49          if (Hop              == Hop_Arr(Idx2))     {
50             Port_Arr(Idx2)    = Port
51             if(Idx3                         > max_out_ports)      {
52                throwMyException("Please add additional ports to the Bridge. Broa
```

```
53              }
54           }
55           ++Idx2
56        }
57        Idx                      = Port_Arr.search("none")
58        if (Idx                  < 0)                  {
59           Idx                   = ThisBroadcast_Len
60        }
61        ++Idx3
62     }
63     Idx                         = 0
64     while (Idx                  < Port_Arr.length()) {
65       if (Port_Arr(Idx)        == "none")            {
66          Port_Block_Name        = "Schedule_" + (Idx+1) + "_" + Node_Name
67          ("Credit_Event_Flag").write(Port_Block_Name, newToken (false))
68       }
69       ++Idx
70     }
71  EXIT
72
73
74  LABEL: BEGIN   // **************************  Begin Port Processing  ***********
75
76
77  SWITCH (port_name)            {
78
79     CASE: input                                          // Node down
80     if (!port_token.TT_Flag && Integration_Technique == "Preemption")
81           port_token.Task_Priority       = 1
82     }
83     else if (SYNC_Hash  == (port_token.Identifier).hashCode()) {
84           port_token.Task_Priority       = 10000
85     }
86     port_token.Task_Hop         = getRoutingTableHop(RoutingTable, Node_Name, port_
87     Port                        = Port_Arr(Hop_Arr.search(port_token.Task_Hop))
88     SEND (Port, port_token)
89     EXIT
```

```
90
91    CASE: node_up
92          if (SYNC_Hash  == (port_token.Identifier).hashCode()) {
93                if(port_token.check("Sync_Cur_Time") && Compression_Master)     {
94                      GTO(END)
95                }
96                if(!port_token.check("Bridge_Trace"))   {
97                      port_token.Bridge_Trace = {Node_Name}
98                }
99                else
100                     Bridge_Trace                    = port_token.Bridge_Trace
101                     Bridge_Trace.append(Node_Name)
102                     port_token.Bridge_Trace = Bridge_Trace
103               }
104               SEND (output, port_token)
105               if(flag_once)   {
106                     flag_once       = false
107                     if(Compression_Master)  {
108                           syncMasterArr_  = ("synchronisationMasterArr").re
109                           syncMasterArr   = {}
110                           qdx                             = 0
111                           while(qdx               < syncMasterArr_.length()
112                                 sm                        = syncMasterArr_(
113                                 nxtHop          = getRoutingTableHop(Rout
114                                 if (nxtHop.hashCode()             != None_
115                                       syncMasterArr.append(sm)
116                                 }
117                                 qdx                       = qdx + 1
118                           }
119                     }
120                     else
121                           syncMasterArr   = {}
122                     }
123                     qdx                                   = 0
124                     while(qdx                       < syncMasterArr.length())
125                           syncMasterDS.setField(syncMasterArr(qdx)+"_Cur_Ti
126                           syncMasterDS.setField(syncMasterArr(qdx)+"_Add_Ti
```

```
127                     syncMasterDS.setField(syncMasterArr(qdx)+"_ID",{}
128                     qdx                             = qdx + 1
129             }
130     }
131     src                     = port_token.SM_Name
132     last_char       = src.substring(src.length()-1)
133     if(last_char == "A" || last_char == "C")        {
134             src             = src.substring(0,src.length()-1)
135     }
136     if(!syncMasterDS.check(src+"_Cur_Time"))        {
137             GTO(END)
138     }
139     idArr           = syncMasterDS.get(src+"_ID")
140     if(idArr.search(port_token.SM_ID)>=0)   {
141             GTO(END)
142     }
143     idArr.append(port_token.SM_ID)
144     syncMasterDS.setField(src+"_ID",idArr)
145     timeArr         = syncMasterDS.get(src+"_Cur_Time")
146     curTime         = port_token.Sync_Gen_Time + (TNow - port_token.N
147     timeArr.append(curTime)
148     syncMasterDS.setField(src+"_Cur_Time",timeArr)
149     startTimeArr            = syncMasterDS.get(src+"_Add_Time")
150     startTimeArr.append(TNow)
151     syncMasterDS.setField(src+"_Add_Time",startTimeArr)
152     qdx                     = 0
153     sum                     = 0.0
154     while(qdx       < syncMasterArr.length())       {
155             timeArr         = syncMasterDS.get(syncMasterArr(qdx)+"_C
156             if(timeArr.length()     > 0)    {
157                     startTimeArr            = syncMasterDS.get(syncMa
158                     sum             = sum + (timeArr(0) + (TNow - sta
159             }
160             else            {
161                     GTO(END)
162             }
163             qdx             = qdx + 1
```

```
164                 }
165                 avgTime         = sum/qdx
166                 qdx                     = 0
167                 while(qdx       < syncMasterArr.length())       {
168                         sm              = syncMasterArr(qdx)
169                         timeArr         = syncMasterDS.get(syncMasterArr(qdx)+"_C
170                         startTimeArr            = syncMasterDS.get(syncMasterArr(
171                         idArr           = syncMasterDS.get(syncMasterArr(qdx)+"_I
172                         if(idArr.length()       >= idArrThreshold)      {
173                                 qdy     = 0
174                                 while(qdy       < idArrPruneLength)             {
175                                         idArr.removeHead()
176                                         qdy             = qdy + 1
177                                 }
178                                 syncMasterDS.setField(syncMasterArr(qdx)+"_ID",id
179                         }
180                         if(timeArr.length()     == 1)   {
181                                 syncMasterDS.setField(syncMasterArr(qdx)+"_Cur_Ti
182                                 syncMasterDS.setField(syncMasterArr(qdx)+"_Add_Ti
183                         }
184                         else            {
185                                 timeArr.removeHead()
186                                 startTimeArr.removeHead()
187                                 syncMasterDS.setField(syncMasterArr(qdx)+"_Cur_Ti
188                                 syncMasterDS.setField(syncMasterArr(qdx)+"_Add_Ti
189                         }
190                         qdx             = qdx + 1
191                 }
192                 port_token.Sync_Cur_Time = avgTime
193         }
194         if(Compression_Master)        {
195                   port_token.Network_Start_Time = TNow
196         }
197         port_token.Bridge_Trace               = {Node_Name}
198     SEND (output, port_token)
199 EXIT
200
```

```
201     CASE: virtual                                          // cannot send virtual to
202        throwMyException (Block_Reference + " cannot send virtual to a Bridge Node.
203
204     CASE: DEFAULT
205        throwMyException (Block_Reference + " unknown port (" + port_name + ") =\n\
206  }$
```

Appendix B

CSV Writer configuration

The following script shows the configurations to outline how data is collected and saved. The configurations include the naming and location of data. It also sets out the formatting of relevant data for analysis, including the required number of columns and rows and the type of data collected. Data sampling rate as in the frequency at which data is sampled and written to a csv file and conditional logging as in the conditions set out to only flag PCFs in breach of the set latency thresholds are some of the main parameters defined in this script.

```
1
2  $required_fields        = {"Task_Size","Task_Source","Task_Destination","Identifi
3  LABEL:BEGIN
4  if(port_token.Identifier        == "SYNC")      {
5          Stream_ID               = "SC"
6          Bridge_Trace    = port_token.Bridge_Trace
7          trace_index             = Bridge_Trace.length() - 1
8          SrcN                    = Bridge_Trace(trace_index)
9          DesN            = port_token.Task_Destination
10 }
11 else                     {
12         SrcName                 = port_token.Task_Source
13         DesName                 = port_token.Task_Destination
14         Stream_ID               = port_token.Stream_ID
15         SrcN                    = "N" + SrcName.substring(SrcName.indexOf("_")+1)
```

```
16          DesN                    = "N" + DesName.substring(DesName.indexOf("_")+1)
17  }
18  Identifier           = port_token.Identifier
19  if (Identifier      == "Ethernet")          {
20    Text              = "_Ether_"
21  }
22  else                                        {
23    Text              = "_" + Identifier + "_"
24  }
25  Trace_Text           = SrcN + "_to_" + DesN + Text + Stream_ID
26  port_token.Plot_Trace   = Trace_Text
27  if (port_token.Identifier       == "SYNC") {
28          port_token.Current_Time = port_token.Sync_Cur_Time
29  }
30  else            {
31          port_token.Current_Time = TNow
32  }
33  Latency                         = port_token.Current_Time - port_token.TIME
34  csv_ds                                      = {Latency = Latency,Legend = Tra
35  qdx                                                 = 0
36  while(qdx                                   < required_fields.length())     {
37          csv_ds.setField(required_fields(qdx),port_token.get(required_fields(qdx))
38          qdx                                         = qdx + 1
39  }
40  SEND(output,csv_ds)$
```

Appendix C

End System Node's Configuration

This is another crucial setting where end systems are configured to closely represent typical devices within a practical IIoT deployment. End system configuration relates to defining how individual nodes behave, process data, and interact with other nodes in the network.

```
1
2  /* Redundant Controller.     */
3
4  $
5  setup_ds                                          = {DS_Name = "Node_Setup",Node_
6  SEND("Network_Setup",setup_ds)
7
8  Idx                    = 0
9  My_ID_Arr              = {}
10 Idx                    = Node_Name.substring(Node_Name.lastIndexOf("_")+1)
11 Name_Arr               = {"","A","C"}
12 Src_Arr                = newArray(3,"Node_" + Idx)
13 Port_Arr               = {"out_south","out_west","out_east"}
14 Src_Arr                = Src_Arr + Name_Arr
15 My_ID                  = (1000   * Idx).longValue()
16 syncCount                          = 0
17 if (Redundant_Nodes  <= 0 || Redundant_Nodes > 3) {
18    throwMyException(Block_Name  + " Redundant_Nodes parameter must be 1,2,3.")
19 }
```

```
20 while(true)                                {
21         deltaTime         = -1000.0
22         TIMEQ("GM_Rate",port_token,1,Grand_Master_Rate)
23 }
24
25 LABEL: BEGIN
26
27
28 SWITCH (port_name)
29
30    CASE: input                                              // to node
31       port_token.Unique_ID = My_ID.incr()
32       if (SYNC_Hash  == (port_token.Identifier).hashCode()) {
33          port_token.Task_Priority = 100000
34          port_token.SM_Name     = Node_Name
35          syncCount        = syncCount + 1
36          port_token.SM_ID       = syncCount
37          port_token.SM_Identifier = port_token.SM_Name + "_" + syncCount
38          Idx          = Src_Arr.search(port_token.Task_Source)
39          if (Idx      < 0)                 {
40             throwMyException(Block_Name    + " Clock Sync cannot find Task_Source
41          }
42          cmArr                  = ("compressionMasterArr").read("Network_Setup")
43          if(cmArr.length() > 0) {
44                 qdx                      = 0
45                 while(qdx       < cmArr.length())       {
46                         dst              = cmArr(qdx)
47                         Idx     = 0
48                         while (Idx      < Redundant_Nodes)   {
49                                 Idx2         = 0
50                                 Destination  = dst + Name_Arr(Idx)
51                                 while (Idx2  < Redundant_Nodes)   {
52                                    Source    = Src_Arr(Idx2)
53                                    Hop       = getRoutingTableHop(Routing_Table_Na
54                                    if (Hop.hashCode()            != None_Hash)
55                                       port_token.Task_Hop         = Hop
56                                       port_token.Task_Source      = Source
```

```
57                                    port_token.Task_Destination = Destination
58                                    destPort           = Port_Arr(Idx2)
59                                    SEND (destPort, port_token)
60                                 }
61                                 ++Idx2
62                              }
63                              ++Idx
64                           }
65                        qdx             = qdx + 1
66                  }
67            }
68            else           {
69                    virtual("WARNING","Synchronization Master "+Node_Name+" couldnt
70                    destPort              = Port_Arr(Idx)
71                SEND (destPort, port_token)
72            }
73            EXIT
74         }
75         if (!port_token.TT_Flag && Integration_Technique == "Timely")
76           port_token.Task_Priority        = 1
77         }
78         Idx             = 0
79         Dest_Node       = port_token.Task_Destination
80         while (Idx      < Redundant_Nodes)   {
81            Idx2         = 0
82            Destination  = Dest_Node + Name_Arr(Idx)
83            while (Idx2  < Redundant_Nodes)   {
84               Source    = Src_Arr(Idx2)
85               Hop       = getRoutingTableHop(Routing_Table_Name, Source, Destinatio
86               if (Hop.hashCode()            != None_Hash)     {
87                  port_token.Task_Hop         = Hop
88                  port_token.Task_Source      = Source
89                  port_token.Task_Destination = Destination
90                  destPort         = Port_Arr(Idx2)
91                  SEND (destPort, port_token)
92               }
93               ++Idx2
```

```
94          }
95          ++Idx
96       }
97    EXIT
98
99    CASE: in_south
100   GTO (From_Node)
101
102   CASE: in_east                                          // fm node
103   CASE: in_west
104      Source          = port_token.Task_Source
105      Idx             = Source.length() - 1
106      Source          = Source.substring (0, Idx)
107      //port_token.Task_Source      = Source
108      port_token.Task_Destination = Node_Name
109
110   LABEL: From_Node
111      if (SYNC_Hash  == (port_token.Identifier).hashCode()) {
112         if(port_token.check("Sync_Cur_Time"))  {
113                  newDelta    = abs(TNow - (port_token.Sync_Cur_Time+((TNow - port
114                  if(deltaTime== -1000.0)        {
115                         deltaTime = newDelta
116                  }
117                  else if(newDelta       < deltaTime)    {
118                         deltaTime = newDelta
119                  }
120                  else                                                {
121                         GTO(END)
122                  }
123                  port_token.Sync_Cur_Time       = (port_token.Sync_Cur_Time+((TNc
124                  if(Inject_Fault                          == "Sync_Drift_Below_Thre
125                         port_token.Sync_Cur_Time        = port_token.Sync_Cur_Tim
126                  }
127                  else if(Inject_Fault                     == "Sync_Drift_Spread") {
128                         port_token.Sync_Cur_Time        = port_token.Sync_Cur_Tim
129                  }
130                  if(abs(TNow - port_token.Sync_Cur_Time) > Max_Sync_Drift_sec)  {
```

```
131                         log                                                 = Node_Na
132                         virtual("WARNING",log)
133                   }
134            }
135            SEND (output, port_token)
136            EXIT
137         }
138         if (!port_token.check("Unique_ID"))  {
139            throwMyException (Block_Name + " cannot find Unique_ID field, pls check
140         }
141         //Idx             = My_ID_Arr.search(port_token.Unique_ID)
142         //if (Idx         < 0)  {
143            //My_ID_Arr    = My_ID_Arr.append(port_token.Unique_ID)
144            //if (My_ID_Arr.length() > 32)      {
145               //My_ID_Arr = My_ID_Arr.removeHead()
146            //}
147         if(port_token.Last_Packet)        {
148            port_token.removeField("Unique_ID")
149            SEND (output, port_token)
150         }
151      EXIT
152
153      CASE: DEFAULT
154         throwMyException(Block_Name + " detected an unknown port: " + port_name)
155   }
156   $
```

Appendix D

Jython program

The following script is the core algorithm that presents the security solution forwarded in this research. It was originally written in Python which is converted to Jython to be able to integrate with the modelling and simulation tool. It presents all the latency thresholds and conditions associated to be able to detect and flag an anomalous sync frame before it is added to a csv file which contains all flagged data for the attention of network practitioners to deploy fitting security solutions.

```
1
2  $import csv
3
4  class Main:
5      def __init__(self):
6          self.ramp_start = 0  # Starting value of the ramp
7          self.ramp_stop = 100  # Ending value of the ramp
8          self.ramp_slope = 10  # Slope of the ramp
9          self.ramp_time = 0  # Time at which the ramp starts
10         self.wired_latency = 1.74E-06  # add the 'latency before fault' value for
11         self.wireless_latency = 5.32E-07  # add the 'latency before fault' value
12         self.data_file = 'C:/Users/BGSAdmin/Documents/Working models/Network Mode
13         self.output_file = 'C:/Users/BGSAdmin/Documents/Working models/flagged_ar
14
15     def read_csv_data(self, file_name):
16         data = []
17         with open(file_name, 'rt') as csvfile:
```

```
18              reader = csv.DictReader(csvfile)
19              for row in reader:
20                  identifier = row['Identifier'].strip().lower()
21                  if identifier != 'sync':
22                      continue
23                  # Rename the 'Latency' field to 'Current Latency'
24                  row['Current Latency'] = row.pop('Latency')
25                  # Calculate the added latency and add it to the row
26                  current_latency = float(row['Current Latency'])
27                  # Determine the 'latency before fault' based on the Bridge used
28                  if row['Task_Source'] == 'Bridge_10':
29                      latency_before_fault = self.wireless_latency
30                  else:
31                      latency_before_fault = self.wired_latency
32                  row['Latency Before Fault'] = str(latency_before_fault)  # Add 'L
33                  added_latency = current_latency - latency_before_fault
34                  row['Added Latency'] = str(added_latency)
35
36                  data.append(row)
37          return data
38
39      def detect_anomalies(self, data, max_time=1, local_threshold_ns=30, global_th
40          flagged_data = []
41          end_systems = {}
42
43          for i, frame in enumerate(data):
44              identifier = frame['Identifier']
45              added_latency_ns = float(frame['Added Latency']) * 1e9  # Convert sec
46
47              if added_latency_ns > global_threshold_ns:
48                  frame['Flagged By'] = 'Global'
49                  flagged_data.append(frame)
50                  end_systems = {}
51                  continue
52
53              # Update the end system dictionary with the latest frame
54              end_systems[identifier] = frame
```

```
55
56              # Check if the end system has exceeded the local threshold for max_ti
57              if all(float(f['Added Latency']) * 1e9 > local_threshold_ns for f in
58                  for f in end_systems.values():
59                      f['Flagged By'] = 'Local'
60                      flagged_data.append(f)
61                  end_systems = {}
62
63          return flagged_data
64
65      def write_csv_data(self, file_name, data):
66          if not data:
67              return
68
69          fieldnames = data[0].keys()
70
71          with open(file_name, 'wt') as csvfile:
72              writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
73              writer.writeheader()
74              writer.writerows(data)
75
76      def ramp(self, time):
77          if time >= self.ramp_time:
78              value = self.ramp_start + (time - self.ramp_time) * self.ramp_slope
79              return min(value, self.ramp_stop)
80          else:
81              return self.ramp_start
82
83      def fire(self):
84          data = self.read_csv_data(self.data_file)
85          flagged_data = self.detect_anomalies(data, max_time=1, local_threshold_ns
86
87          # Write the flagged anomalies to the output CSV file
88          self.write_csv_data(self.output_file, flagged_data)
89          print("Flagged anomalies: %s" % len(flagged_data))
90
91  # Initialize the Main class and call the fire method
```

```
92 main_obj = Main()
93 main_obj.fire()
94 $
```

Bibliography

Abada, R., Abubakar, A. M., and Bilal, M. T. (2022). An overview on deep leaning application of big data. *Mesopotamian Journal of Big Data*, 2022:31–35.

Abuteir, M. and Obermaisser, R. (2013). Simulation environment for time-triggered ethernet. In *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, pages 642–648. IEEE.

Abuteir, M. and Obermaisser, R. (2015). Scheduling of rate-constrained and time-triggered traffic in multi-cluster ttethernet systems. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 239–245. IEEE.

Alghamdi, W. and Schukat, M. (2017). Advanced methodologies to deter internal attacks in ptp time synchronization networks. In *2017 28th Irish Signals and Systems Conference (ISSC)*, pages 1–6. IEEE.

Alghamdi, W. and Schukat, M. (2021). Precision time protocol attack strategies and their resistance to existing security extensions. *Cybersecurity*, 4:1–17.

Alzarqawee, A. N. J. and Fritsch, L. (2023). Towards ai-powered cybersecurity attack modeling with simulation tools: Review of attack simulators. In *Advances on P2P, Parallel, Grid, Cloud and Internet Computing Proceedings of the 17th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2022)*. Springer.

Annessi, R., Fabini, J., Iglesias, F., and Zseby, T. (2018). Encryption is futile: Delay attacks on high-precision clock synchronization. *arXiv preprint arXiv:1811.08569*.

Ayu, H., Jufriadi, A., Mustika, S., Kurniawati, M., Pratiwi, H., Sundaygara, C., and

Hudha, M. (2021). How to learn oscillation and wave in samr framework? In *Journal of Physics: Conference Series*, page 012160. IOP Publishing.

Banđur, Đ., Jakšić, B., Banđur, M., and Jović, S. (2019). An analysis of energy efficiency in wireless sensor networks (wsns) applied in smart agriculture. *Computers and electronics in agriculture*, 156:500–507.

Bhoi, S. K., Sahu, P. K., Singh, M., Khilar, P. M., Sahoo, R. R., and Swain, R. R. (2019). Local traffic aware unicast routing scheme for connected car system. *IEEE Transactions on Intelligent Transportation Systems*, 21(6):2360–2375.

Bose, R., Roy, S., Chakraborty, S., and Sarkar, I. (2020). Development of a real-time fuel monitoring system for construction industry using internet of things. *International Journal of Innovative Technology and Exploring Engineering*, 9(3):2011–2017.

Boyes, H., Hallaq, B., Cunningham, J., and Watson, T. (2018). The industrial internet of things (iiot): An analysis framework. *Computers in industry*, 101:1–12.

Burhan, M., Rehman, R. A., Khan, B., and Kim, B.-S. (2018). Iot elements, layered architectures and security issues: A comprehensive survey. *sensors*, 18(9):2796.

Centerholt, V., Kjidderö, F., Saarikko, T., and Grahn, S. (2020). Value chains vs. ecosystems: Current perspectives among swedish smes entering the interconnected world of iot. *SPS2020*, pages 489–500.

Choi, J., Shin, Y., and Cho, S. (2018). Study on information security sharing system among the industrial iot service and product provider. In *2018 International Conference on Information Networking (ICOIN)*, pages 551–555. IEEE.

Conklin, W. A. (2016). It vs. ot security: A time to consider a change in cia to include resilienc. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 2642–2647. IEEE.

Conti, M., Dragoni, N., and Lesyk, V. (2016). A survey of man in the middle attacks. *IEEE communications surveys & tutorials*, 18(3):2027–2051.

Daniel, O. and Roman, O. (2018). Fault injection framework for assessing fault containment of ttethernet against babbling idiot failures. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–6. IEEE.

de las Morenas, J., da Silva, C. M., Funchal, G. S., Melo, V., Vallim, M., and Leitao, P. (2020). Security experiences in iot-based applications for building and factory automation. In *2020 IEEE International Conference on Industrial Technology (ICIT)*, pages 322–327. IEEE.

De Vincenzi, M., Costantino, G., Matteucci, I., Fenzl, F., Plappert, C., Rieke, R., and Zelle, D. (2024). A systematic review on security attacks and countermeasures in automotive ethernet. *ACM Computing Surveys*, 56(6):1–38.

Deepak, S., Anandakumar, H., Pavithra, S., Keerthika, V., and Nandhini, K. (2022). Performance analysis of star topology for small networks using riverbed. In *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 1, pages 2108–2111. IEEE.

Dutertre, B., Easwaran, A., Hall, B., and Steiner, W. (2012). Model-based analysis of timed-triggered ethernet. In *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, pages 9D2–1. IEEE.

EneoSigma (2023). It/ot integration platforms. URL: `http://www.eneosigma.com/solutions.html/`. Accessed: 20-08-2023.

Ethernet, T.-T. (2016). As6802™.

Evans, M., He, Y., Luo, C., Yevseyeva, I., Janicke, H., and Maglaras, L. A. (2019). Employee perspective on information security related human error in healthcare: Proactive use of is-chec in questionnaire form. *IEEE Access*, 7:102087–102101.

Falliere, N., Murchu, L. O., and Chien, E. (2011). W32. stuxnet dossier. *White paper, Symantec corp., security response*, 5(6):29.

Fan, K., Ren, Y., Yan, Z., Wang, S., Li, H., and Yang, Y. (2018). Secure time synchronization scheme in iot based on blockchain. In *2018 IEEE international*

conference on Internet of Things (IThings) and IEEE green computing and Communications (GreenCom) and IEEE Cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData), pages 1063–1068. IEEE.

Garimella, P. K. (2018). It-ot integration challenges in utilities. In *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, pages 199–204. IEEE.

Guo, L., Wan, B., Li, C., Zhou, K., and Li, X. (2018). A flow-grained end-to-end delay analysis for rc traffic in ttethernet. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 904–913. IEEE.

Guo, Z., Tian, R., Xu, W., Yip, D., Radyk, M., Santos, F. B., Yip, A., Chen, T., and Tang, X. S. (2022). Highly accurate heart failure classification using carbon nanotube thin film biosensors and machine learning assisted data analysis. *Biosensors and Bioelectronics: X*, 12:100187.

Hals, A. (2015). Well integrity assessment: Challenges related to human and organizational factors-the case study of veslefrikk. Master's thesis, NTNU.

Herrera, H. A., Rivas, W. R., and Kumar, S. (2018). Evaluation of internet connectivity under distributed denial of service attacks from botnets of varying magnitudes. In *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, pages 123–126. IEEE.

Hilal, W., Gadsden, S. A., and Yawney, J. (2022). Financial fraud: a review of anomaly detection techniques and recent advances. *Expert systems With applications*, 193:116429.

Huang, C., Wen, J., Xu, Y., Jiang, Q., Yang, J., Wang, Y., and Zhang, D. (2022). Self-supervised attentive generative adversarial networks for video anomaly detection. *IEEE Transactions on neural networks and learning systems*.

Huang, D.-J., Teng, W.-C., and Yang, K.-T. (2013). Secured flooding time synchronization protocol with moderator. *International Journal of Communication Systems*, 26(9):1092–1115.

Jayabharathi, S. and Ilango, V. (2022). Anomaly detection using machine learning techniques: A systematic review. In *International Conference on Advances in Data-driven Computing and Intelligent Systems*, pages 553–572. Springer.

Jin, W. (2020). Research on machine learning and its algorithms and development. In *Journal of Physics: Conference Series*, page 012003. IOP Publishing.

Kawamura, T., Fukushi, M., Hirano, Y., Fujita, Y., and Hamamoto, Y. (2017). An ntp-based detection module for ddos attacks on iot. In *2017 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pages 15–16. IEEE.

Kiersztyn, K. and Kiersztyn, A. (2022). Fuzzy rule-based outlier detector. In *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–7. IEEE.

Kirk, R. (2015). Cars of the future: the internet of things in the automotive industry. *Network Security*, 2015(9):16–18.

Kolias, C., Kambourakis, G., Stavrou, A., and Voas, J. (2017). Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84.

Koulamas, C. and Kalogeras, A. (2018). Cyber-physical systems and digital twins in the industrial internet of things [cyber-physical systems]. *Computer*, 51(11):95–98.

Kyriakakis, E., Lund, M., Pezzarossa, L., Sparsø, J., and Schoeberl, M. (2020). A time-predictable open-source ttethernet end-system. *Journal of Systems Architecture*, 108:101744.

Lei, Z., Ren, S., Hu, Y., Zhang, W., and Chen, S. (2022). Latency-aware collaborative perception. In *European Conference on Computer Vision*, pages 316–332. Springer.

Lévesque, M. and Tipper, D. (2016). A survey of clock synchronization over packet-switched networks. *IEEE Communications Surveys & Tutorials*, 18(4):2926–2947.

Li, R., Fu, B., Xie, G., Peng, F., and Li, R. (2023). Efficient holistic timing analysis with low pessimism for rate-constrained traffic in ttethernet. *Journal of Systems Architecture*, 134:102785.

Liang, G., Weller, S. R., Zhao, J., Luo, F., and Dong, Z. Y. (2016). The 2015 ukraine blackout: Implications for false data injection attacks. *IEEE Transactions on power systems*, 32(4):3317–3318.

Lichtsinder, B. Y. (2022). Ethernet networks with deterministic delays. *Vestnik of Samara State Technical University. Technical Sciences Series*, 30(3):81–97.

Lim, F. P. (2016). A review-analysis of network topologies for microenterprises. *Small*, 3:15–000.

Lisova, E. (2018). *Monitoring for securing clock synchronization*. PhD thesis, Mälardalen University.

Lisova, E., Gutiérrez, M., Steiner, W., Uhlemann, E., Åkerberg, J., Dobrin, R., Björkman, M., et al. (2016a). Protecting clock synchronization: Adversary detection through network monitoring. *Journal of Electrical and Computer Engineering*, 2016.

Lisova, E., Uhlemann, E., Åkerberg, J., and Björkman, M. (2014). Towards secure wireless ttethernet for industrial process automation applications. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–4. IEEE.

Lisova, E., Uhlemann, E., Steiner, W., Åkerberg, J., and Björkman, M. (2015). A survey of security frameworks suitable for distributed control systems. In *2015 International Conference on Computing and Network Communications (CoCoNet)*, pages 205–211. IEEE.

Lisova, E., Uhlemann, E., Steiner, W., Åkerberg, J., and Björkman, M. (2016b). Risk evaluation of an arp poisoning attack on clock synchronization for industrial applications. In *2016 IEEE international conference on industrial technology (ICIT)*, pages 872–878. IEEE.

Luong, A., Hillyard, P., Abrar, A., Che, C., Rowe, A., Schmid, T., and Patwari, N. (2018). A stitch in time and frequency synchronization saves bandwidth. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 96–107. IEEE.

Maksutov, A. A., Cherepanov, I. A., and Alekseev, M. S. (2017). Detection and prevention of dns spoofing attacks. In *2017 Siberian Symposium on Data Science and Engineering (SSDSE)*, pages 84–87. IEEE.

Martins, G., Bhattacharjee, A., Dubey, A., and Koutsoukos, X. D. (2014). Performance evaluation of an authentication mechanism in time-triggered networked control systems. In *2014 7th International Symposium on Resilient Control Systems (ISRCS)*, pages 1–6. IEEE.

Mazur, D. C., Entzminger, R. A., Kay, J. A., and Morell, P. A. (2016). Time synchronization mechanisms for the industrial marketplace. *IEEE Transactions on Industry Applications*, 53(1):39–46.

Mirani, A. A., Velasco-Hernandez, G., Awasthi, A., and Walsh, J. (2022). Key challenges and emerging technologies in industrial iot architectures: A review. *Sensors*, 22(15):5836.

Mizrahi, T. (2014). Security requirements of time protocols in packet switched networks. *RFC*, 7384:1–36.

Nithya, B. and Ilango, V. (2017). Predictive analytics in health care using machine learning tools and techniques. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 492–499. IEEE.

Paes, R., Mazur, D. C., Venne, B. K., and Ostrzenski, J. (2019). A guide to securing industrial control networks: Integrating it and ot systems. *IEEE Industry Applications Magazine*, 26(2):47–53.

Peón, P. G., Kopetz, H., and Steiner, W. (2014). Towards a reliable and high-speed wireless complement to ttethernet. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–4. IEEE.

Peón, P. G., Uhlemann, E., Steiner, W., and Björkman, M. (2015). A wireless mac method with support for heterogeneous data traffic. In *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*, pages 003869–003874. IEEE.

Poór, P., Basl, J., and Zenisek, D. (2019). Predictive maintenance 4.0 as next evolution step in industrial maintenance development. In *2019 international research conference on smart computing and systems engineering (SCSE)*, pages 245–253. IEEE.

Prehanto, D. R., Indriyanti, A. D., and Permadi, G. S. (2021). Performance analysis routing protocol between ripv2 and eigrp with termination test on full mesh topology. *Indones. J. Electr. Eng. Comput. Sci*, 23(1):354–361.

Przybylski, T., Sugunaraj, N., and Ranganathan, P. (2023). Aircraft communication systems-topologies, protocols, and vulnerabilities.

Qiu, T., Zhang, Y., Qiao, D., Zhang, X., Wymore, M. L., and Sangaiah, A. K. (2017). A robust time synchronization scheme for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 14(8):3570–3580.

Razvi, S. S., Feng, S., Narayanan, A., Lee, Y.-T. T., and Witherell, P. (2019). A review of machine learning applications in additive manufacturing. In *International design engineering technical conferences and computers and information in engineering conference*, volume 59179, page V001T02A040. American Society of Mechanical Engineers.

Rehman, S. U., Singh, P., Manickam, S., and Praptodiyono, S. (2020). Towards sustainable iot ecosystem. In *2020 2nd International Conference on industrial electrical and electronics (ICIEE)*, pages 135–138. IEEE.

Ruiz-Garcia, L., Lunadei, L., Barreiro, P., and Robla, J. I. (2009). A review of wireless sensor technologies and applications in agriculture and food industry: state of the art and current trends. *sensors*, 9(6):4728–4750.

SA, I. (2003). Standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 3: Carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications - maintenance 7. *IEEE Std 802.3aj-2003 (Amendment to IEEE Std 802.3-2002)*, pages 1–78.

Seijo, Ó., Val, I., Luvisotto, M., and Pang, Z. (2021). Clock synchronization for wireless time-sensitive networking: A march from microsecond to nanosecond. *IEEE Industrial Electronics Magazine*, 16(2):35–43.

Sethi, P., Sarangi, S. R., et al. (2017). Internet of things: architectures, protocols, and applications. *Journal of Electrical and computer engineering*, 2017.

Shad, R., Broklyn, P., and Egon, A. (2024). The evolving thread landscape pf ai-powered cyberattacks: a multi-faceted approach to defense and mitigate. Technical report, EasyChair.

Shepard, D. P., Humphreys, T. E., and Fansler, A. A. (2012). Evaluation of the vulnerability of phasor measurement units to gps spoofing attacks. *International Journal of Critical Infrastructure Protection*, 5(3-4):146–153.

Smache, M., Olivereau, A., Franco-Rondisson, T., and Assia, T. (2019). Autonomous detection of synchronization attacks in the industrial internet of things. In *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pages 1–9. IEEE.

Song, J., Han, S., Mok, A., Chen, D., Lucas, M., Nixon, M., and Pratt, W. (2008). Wirelesshart: Applying wireless technology in real-time industrial process control. In *2008 IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 377–386. IEEE.

Steiner, W. (2013). Candidate security solutions for ttethernet. In *2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC)*, pages 4A5–1. IEEE.

Steiner, W., Bauer, G., Hall, B., Paulitsch, M., and Varadarajan, S. (2009). Ttethernet dataflow concept. In *2009 Eighth IEEE International Symposium on Network Computing and Applications*, pages 319–322. IEEE.

Stouffer, K., Falco, J., Scarfone, K., et al. (2011). Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16–16.

Suethanuwong, E. (2012). Scheduling time-triggered traffic in ttethernet systems. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, pages 1–4. IEEE.

Tabassum, I., Bazai, S. U., Zaland, Z., Marjan, S., Khan, M. Z., and Ghafoor, M. I. (2022). Cyber security's silver bullet-a systematic literature review of ai-powered security. In *2022 3rd International Informatics and Software Engineering Conference (IISEC)*, pages 1–7. IEEE.

Tămaş-Selicean, D. and Pop, P. (2014). Optimization of ttethernet networks to support best-effort traffic. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–4. IEEE.

Tang, X., Li, Q., Lu, G., and Xiong, H. (2018). Safe clock synchronization mechanism for multi-cluster ttethernet networks. In *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6. IEEE.

Tariq, N., Petrunin, I., Tsourdos, A., and Al-Rubaye, S. (2020). External synchronisation in time-triggered networks. In *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–7. IEEE.

Tawalbeh, L., Muheidat, F., Tawalbeh, M., and Quwaider, M. (2020). Iot privacy and security: Challenges and solutions. *Applied Sciences*, 10(12):4102.

Toghuj, W. and Turab, N. (2023). Automotive ethernet architecture and security: challenges and technologies. *International Journal of Electrical & Computer Engineering (2088-8708)*, 13(5).

Tsiknas, K., Taketzis, D., Demertzis, K., and Skianis, C. (2021). Cyber threats to industrial iot: a survey on attacks and countermeasures. *IoT*, 2(1):163–186.

Tămaş-Selicean, D., Marinescu, S. O., and Pop, P. (2012). Analysis and optimization of mixed-criticality applications on partitioned distributed architectures. In *7th IET International Conference on System Safety, incorporating the Cyber Security Conference 2012*, pages 1–6.

Ullmann, M. and Vögeler, M. (2009). Delay attacks—implication on ntp and ptp time synchronization. In *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, pages 1–6. IEEE.

Varga, P., Plosz, S., Soos, G., and Hegedus, C. (2017). Security threats and issues in automation iot. In *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, pages 1–6. IEEE.

Wang, J., Ding, P., Wang, Y., and Yan, G. (2018). Back-to-back optimization of schedules for time-triggered ethernet. In *2018 37th Chinese Control Conference (CCC)*, pages 6398–6403. IEEE.

Wu, H.-T., Hu, W.-C., and Jiang, B.-W. (2019). General-purpose intelligent management system of logistics fleet. In *2019 International Conference on Intelligent Computing and its Emerging Applications (ICEA)*, pages 92–96. IEEE.

Xu, C., Zhang, L., Ling, Z., Xu, M., and Wang, D. (2017). Ttethernet for launch vehicle communication network. In *2017 29th Chinese Control And Decision Conference (CCDC)*, pages 5159–5163. IEEE.

Yang, W., Wang, Q., Wan, Y., and He, J. (2016a). Security vulnerabilities and countermeasures for time synchronization in ieee802. 15.4 e networks. In *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 102–107. IEEE.

Yang, X., Zhang, S., and Li, H. (2016b). Research and implementation of precise time synchronization system of microgrid based on ieee 1588. In *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pages 258–261. IEEE.

Yin, D., Zhang, L., and Yang, K. (2018). A ddos attack detection and mitigation with software-defined internet of things framework. *IEEE Access*, 6:24694–24705.

Zhang, Y., He, F., Lu, G., and Xiong, H. (2016). Clock synchronization compensation of time-triggered ethernet based on least squares algorithm. In *2016 IEEE/CIC*

International Conference on Communications in China (ICCC Workshops), pages 1–5. IEEE.

Zhao, L., He, F., Li, E., and Lu, J. (2018). Comparison of time-sensitive networking (tsn) and ttethernet. In *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pages 1–7. IEEE.

Zhao, L., Matsuo, I. B. M., Zhou, Y., and Lee, W.-J. (2019a). Design of an industrial iot-based monitoring system for power substations. *IEEE Transactions on Industry Applications*, 55(6):5666–5674.

Zhao, R., Qin, G., Lyu, Y., and Yan, J. (2019b). Security-aware scheduling for ttethernet-based real-time automotive systems. *IEEE Access*, 7:85971–85984.