Self-learning Anomaly Detection in Industrial Production

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

M.Sc.

Ankush Meshram

aus Brahmapuri, Indien

Tag der mündlichen Prüfung:	04.10.2022
Erster Gutachter:	Prof. Dr.-Ing. habil. Jürgen Beyerer
Zweiter Gutachter:	Prof. Dr. Veit Hagenmeyer

Abstract

Industrial production systems of the 21st Century are designed for production cost reduction through efficient control of cyber-physical industrial components. It is realized through the adaptation of Ethernet technology in industrial networking, which blurs the separation between the *office networks* and the *industrial networks*, to make a sensor in the production floor accessible to the personnel in the corporate office. The blurring of network separation led to cybersecurity vulnerabilities to industrial production, as demonstrated by recent cyber incidents from *Stuxnet* to *Triton* in the last decades. An anomaly-based Network-based Intrusion Detection Systems (NIDS) is one of the cybersecurity countermeasures that monitors the network traffic of an industrial production and learns the characteristics from its network infrastructure information to detect the deviations as *anomalies.* However, the network infrastructure information, such as asset inventory and security policies, are not always available for industrial productions that are non-compliant to the technological advancements in security mechanisms. In such a scenario, the configuration or designing the security countermeasures such as NIDS is challenging. An empirical solution is to self-learn the network infrastructure information (topology, assets and communication links) and the *spatio-temporal* behaviour from passive monitoring of industrial network traffic, in conjunction with an anomaly detection system to detect anomalies. In the absence of industrial process information, learning the process behaviour from the network traffic for anomaly detection is a challenge.

In this dissertation, solutions for the different challenges posed by the self-learning anomaly-based NIDS for industrial production are presented. For the development and evaluation of reported solutions, a miniaturized *PROFINET*-based industrial production system, *Festo Demonstrator*, developed at the

IT-Security Laboratory, Fraunhofer Institute of Optronics, System Technologies and Image Exploitation, Karlsruhe (IOSB) is employed. Various cyber attacks on the demonstrator are simulated and the generated network traffic is used for the evaluation. A systematic Python-based framework passively captures the network traffic from *PROFINET*-based system and extracts relevant information for further analysis as the solution to the challenge of making the industrial network transparent. The extracted network infrastructure information are: *network topology*, *assets and their attributes*, *communication links established between master and slave* (referred as *PROFINET Connection*), *time-stamped process data payload bytes.*

The industrial operations of a production system executed in an order create the foundation for process data exchanges realizing the underlying intended process. It is a challenge to enumerate and track the industrial system's operation behaviour from the multiple protocol communications observed in the network traffic. As the solution, the extracted network infrastructure information is utilized to systematically enumerate and track the *validity* of *PROFINET* operations at different granularities of device, process data communication link and system levels via corresponding *Finite State Machines (FSM)* models. The implemented framework, *PROFINET Operations Enumeration and Tracking (POET)*, utilizes the FSM models to successfully detect the network attack on the demonstrator, where an adversary attempts to change the parameters of an industrial component through valid protocol communication but invalid *PROFINET* operation during the process data exchange.

The industrial process behaviour is defined by the *spatio-temporal* characteristics of periodic, deterministic and ordered exchange of process parameters between industrial components. In a self-learning analysis approach, there are no insights into the semantics of process values being exchanged as the information on the programmed industrial process is unavailable. It puts forward the challenges of representing the extracted time-stamped process data payload bytes and learning the *spatio-temporal* characteristics to detect the deviations as anomalies. Two different representations of process information, *String* and *Graph*, are proposed to capture the *spatial* (order of the payloads) and *temporal* (interval between process payload exchanges) characteristics. Consequently, two different approaches of process behaviour analysis for anomaly detection in industrial production are explored. The industrial process is conceptualized as the constituent of multiple sub-processes. Each sub-process represents the process data communication link established between a pair of master and slave devices exchanging a particular process parameter of the overall process.

In the first approach, the extracted ordered process data payloads are represented as *String*, where the finite set of payload bytes are alphabet-encoded for the industrial production's process and its constituent sub-processes. The temporal information associated with payload bytes is utilized to define the transition intervals between pair of adjacent encoded payload bytes, thus, capturing the *spatio-temporal* characteristics. A systematic framework, *Payload Bytes Profiling (PBP)*, represents the *spatio-temporal* characteristics into spatial (*transition profile*) and temporal (*interval profile*) profiles. A Regular Expression-coupled Suffix Tree algorithm extracts the *transition profiles* at the process and sub-process granularities from the alphabet-encoded process traffic to model the *spatial* characteristics of an industrial process/sub-processes. For *temporal* characteristics modeling, the transition intervals between the transitions of process payload bytes in the *transition profile* of the process/sub-process are modeled through non-parametric Machine Learning (ML) models - Local Outlier Factor (LOF), Isolation Forest (IF) and One-Class Support Vector Machines (OC-SVM), and represents the *temporal* characteristics as the *interval profile* in PBP of a sub-process or the overall process. The PBP for the *Festo Demonstrator* successfully detects the anomalies triggered by the network attack on the demonstrator, where the adversary changes the process values in a sub-process to disrupt the industrial process behaviour.

For the second approach, a *Graph* representation of an industrial network, *graph snapshot*, captures the *spatio-temporal* characteristics of its underlying process with *nodes* representing the industrial components and the *edges* being the communication link between the components. The *graph snapshot* is created from the alphabet-encoded industrial process traffic, and the transition interval information in conjunction with transitioning payload bytes and the differences in their *Cycle Counter* values are encoded on the edges over a fixed time window on the traffic. The anisotropic Graph Neural Networks (GNN) architectures (Message Passing Neural Network (MPNN), Gated Graph Convolutional Network (GatedGCN) and Graph Transformer (GT)) are employed to learn the *spatio-temporal* characteristics of process behaviour with *graph classification* as the ML task to classify *graph snapshots* as *normal* or *anomalous*. The network attack targeting the process values of a sub-process is successfully detected by the anisotropic GNN architectures demonstrating their capability to utilize the edge features encoding the *spatio-temporal* characteristics in their learning.

Kurzfassung

Industrielle Produktionssysteme des 21. Jahrhunderts sind auf eine Senkung der Produktionskosten durch eine effiziente Steuerung von cyber-physischen Industriekomponenten ausgelegt. Realisiert wird dies durch die Anpassung der Ethernet-Technologie in der industriellen Vernetzung, die die Trennung zwischen Büro- und Industrienetzwerken aufhebt, um einen Sensor in der Produktion für das Personal im Büro zugänglich zu machen. Die fehlende strikte Netzwerktrennung führte zu Schwachstellen in der Cybersicherheit der industriellen Produktion, wie die jüngsten Cybervorfälle von *Stuxnet* bis *Triton* in den letzten Jahrzehnten zeigen. Ein auf Anomalien basierendes Network-based Intrusion Detection Systems (NIDS) ist eine der Gegenmaßnahmen zur Cybersicherheit, die den Netzwerkverkehr einer industriellen Produktion überwacht. Es lernt die Merkmale aus den Informationen der Netzwerkinfrastruktur, um die Abweichungen als *Anomalien* zu erkennen. Die Informationen über die Netzwerkinfrastruktur, wie z. B. das Anlageninventar und die Sicherheitsrichtlinien, sind jedoch nicht immer für industrielle Produktionsanlagen verfügbar. In einem solchen Szenario stellt die Konfiguration und der Entwurf von Sicherheitsmaßnahmen, wie der Einsatz eines NIDS, eine Herausforderung dar. Eine empirische Lösung ist das Selbstlernen der Netzinfrastruktur (Topologie, Anlagen und Kommunikationsverbindungen) und des *zeitlich-räumlichen* Verhaltens aus der passiven Überwachung des industriellen Netzverkehrs. In Verbindung mit einem System zur Anomalieerkennung werden so Abweichungen als Anomalien erkannt. Aufgrund der fehlenden Informationen der industriellen Prozesse stellt das Erlernen des Prozessverhaltens mittels Netzwerkdaten eine Herausforderung dar.

In dieser Dissertation werden Lösungen für die verschiedenen Herausforderungen des selbstlernenden anomalie-basierten NIDS für die industrielle

Produktion vorgestellt. Für die Entwicklung der vorgestellten Lösungen wird ein miniaturisiertes *PROFINET*-basiertes industrielles Produktionssystem, der *Festo-Demonstrator*, verwendet, der am FRAUNHOFER IOSB im Rahmen des Projekts IT-SICHERHEITSLABOR für die industrielle Produktion entwickelt wurde. Auf dem Demonstrator werden verschiedene Cyberangriffe simuliert und der generierte Netzwerkverkehr zur Auswertung herangezogen. Ein Python-basiertes Framework erfasst passiv den Netzwerkverkehr des *PROFINET*-basierten Systems und extrahiert relevante Informationen für die weitere Analyse. Hiermit wird die Herausforderung, das industrielle Netzwerk transparent zu machen, gelöst. Die extrahierten Netzwerkinfrastrukturinformationen sind: *Netztopologie*, *Assets und ihre Attribute*, *Kommunikationsverbindungen zwischen Master und Slave* (bezeichnet als *PROFINET Connection*), *zeitgestempelte Prozessdaten-Nutzdatenbytes*.

Die industriellen Operationen eines Produktionssystems, die in einem Zyklus ausgeführt werden, bilden die Grundlage für den Austausch von Prozessdaten, die den zugrunde liegenden beabsichtigten Prozess realisieren. Es ist eine Herausforderung, das Betriebsverhalten des industriellen Systems anhand der im Netzwerkverkehr beobachteten Protokollkommunikation aufzulisten (*enumerate*) und nachzuverfolgen (*track*). Die Lösung nutzt die extrahierten Netzinfrastrukturinformationen, um die *PROFINET*-Operationen auf unterschiedlichen Granularitäten der Geräte-, Prozessdatenkommunikations- und Systemebene über entsprechende Finite State Machines (FSM)-modelle systematisch aufzulisten und deren Gültigkeit zu tracken. Das implementierte Python-basierte Framework, *PROFINET Operations Enumeration and Tracking (POET)*, nutzt die FSM-Modelle, um erfolgreich den Netzwerkangriff auf den Demonstrator zu erkennen, bei dem ein Angreifer versucht, die Parameter einer industriellen Komponente durch gültige Protokollkommunikation, aber ungültige *PROFINET*-Operationen während des Prozessdatenaustauschs zu ändern.

Das Verhalten des industriellen Prozesses wird durch die *räumlich-zeitlichen* Merkmale des periodischen, deterministischen und geordneten Austauschs von Prozessparametern zwischen industriellen Komponenten definiert. Bei einem selbstlernenden Analyseansatz gibt es keine Einblicke in die Semantik der ausgetauschten Prozesswerte, da die Informationen über den programmierten industriellen Prozess nicht verfügbar sind. Die Herausforderung besteht darin, die extrahierten und mit Zeitstempeln versehenen Bytes der Prozessdaten zu repräsentieren und die *räumlich-zeitlichen* Merkmale zu erlernen, um Abweichungen als Anomalien zu erkennen. Zwei verschiedene Darstellungen von Prozessinformationen, *String* und *Graph*, werden vorgeschlagen, um die *räumliche* (Reihenfolge der Nutzdaten) und *zeitliche* (Intervall zwischen dem Austausch von Prozessnutzdaten) Charakteristik zu erfassen. Folglich werden zwei verschiedene Ansätze zur Analyse des Prozessverhaltens für die Erkennung von Anomalien in der industriellen Produktion untersucht. Der industrielle Prozess wird als Bestandteil mehrerer Teilprozesse modelliert. Jeder Teilprozess stellt die Prozessdaten-Kommunikationsverbindung zwischen einem Paar von Master- und Slave-Geräten dar, die einen bestimmten Prozessparameter des Gesamtprozesses austauschen.

Beim ersten Ansatz werden die extrahierten geordneten Prozessdaten als *String* dargestellt, wobei die endliche Menge von Nutzdatenbytes für den Prozess der industriellen Produktion und die zugehörigen Teilprozesse alphabetisch kodiert werden. Die zeitlichen Informationen, die mit den Nutzdatenbytes verbunden sind, werden verwendet, um die Übergangsintervalle zwischen Paaren benachbarter kodierter Nutzdatenbytes zu definieren und so die räumlich-zeitlichen Merkmale zu erfassen. Ein systematischer Rahmen, *Payload Bytes Profiling (PBP)*, stellt die *räumlich-zeitlichen* Merkmale in räumlichen (*Übergangsprofil*) und zeitlichen (*Intervallprofil*) Profilen dar. Ein mit regulären Ausdrücken gekoppelter Suffixbaum-Algorithmus extrahiert die *Übergangsprofile* auf der Prozess- und Teilprozessgranularität aus dem alphabetisch kodierten Prozessverkehr, um die *räumlichen* Merkmale eines industriellen Prozesses/Teilprozesses zu modellieren. Für die Modellierung der zeitlichen Merkmale werden die Übergangsintervalle zwischen den Übergängen der Prozessnutzlastbytes im Übergangsprofil des Prozesses/-Unterprozesses durch nichtparametrische Machine Learning (ML)-Modelle - Local Outlier Factor (LOF), Isolation Forest (IF) und One-Class Support Vector Machines (OC-SVM) - modelliert und die zeitlichen Merkmale als Intervallprofil im PBP eines Unterprozesses oder des Gesamtprozesses dargestellt. Das PBP für den *Festo Demonstrator* erkennt erfolgreich die Anomalien, die durch den Netzwerkangriff auf den Demonstrator ausgelöst werden, bei dem der Angreifer die Prozesswerte in einem Teilprozess ändert, um das Verhalten des industriellen Prozesses zu stören.

Beim zweiten Ansatz erfasst eine *Graph*-Darstellung eines industriellen Netzwerks, *graph snapshot*, die räumlich-zeitlichen Merkmale des zugrunde liegenden Prozesses, wobei *Knoten* die industriellen Komponenten und die *Kanten* die Kommunikationsverbindung zwischen den Komponenten darstellen. Der *graph snapshot* wird aus dem alphabetisch kodierten industriellen Prozessverkehr erstellt, und die Übergangsintervallinformationen in Verbindung mit den übergehenden Nutzdatenbytes und den Unterschieden in ihren *Cycle Counter*-Werten werden an den Kanten über ein festes Zeitfenster des Verkehrs kodiert. Anisotropische Graph Neural Networks (GNN) Architekturen (Message Passing Neural Network (MPNN), Gated Graph Convolutional Network (GatedGCN) und Graph Transformer (GT)) werden verwendet, um die räumlich-zeitlichen Merkmale des Prozessverhaltens zu erlernen, wobei die ML-Aufgabe der Graphenklassifizierung die *graph snapshots* als normal oder anomal klassifiziert. Der auf die Prozesswerte eines Teilprozesses abzielende Netzwerkangriff wird von den anisotropen GNN-Architekturen erfolgreich erkannt, was ihre Fähigkeit unter Beweis stellt, die Kantenmerkmale, welche die *räumlich-zeitlichen* Merkmale kodieren, beim Lernen zu nutzen.

Acknowledgements

I am eternally grateful to Prof. Dr.-Ing. Jürgen Beyerer for giving me this opportunity at the Chair Vision and Fusion Laboratory (IES), KIT, funded by the Competence Center for Applied Security Technology (KASTEL). His constant encouragement and insightful discussions over the years have helped me to explore the research problems in the right direction. I am also thankful to Prof. Dr. Veit Hagenmeyer for being the second reviewer and providing constructive suggestions for improving the quality of research. This research work is conducted in cooperation with the Group Industrial Cybersecurity (IC) at Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB) led by Dr.-Ing. Christian Haas. I am grateful to him for patiently and enthusiastically answering all of my novice queries which have helped me to grow as a researcher.

The warmth of my colleagues at the Chair IES and the group IC have made the research work a fun learning experience. Especially Dr. Alexey Pak, Andreas Pfaffenrot, Anne Borcherding, David Meier, Dr.-Ing. Florian Patzer, Jörg Kippe, Markus Karch, Steffen Pfrang and Dr. Tim Zander have been the sounding board to help me develop the research avenues. I am grateful to my friends Julie and Micah Hodgins, Pooja Ravikumar, Russa Biswas, Saptarshi Mitra, and Tim Schoock, who made Germany a home away from home. Also, my heartfelt gratitude to Anne, Christian, Markus, Russa, and Saptarshi, for the countless research discussions that culminated in the presented work.

Last but not the least, this journey wouldn't have been possible without the unconditional love and support of my family - Alka and Prabhu Meshram, Nisha Kotwani, and Sujay Meshram.

Karlsruhe, July 2022 *Ankush Meshram*

Contents

1 Introduction

The *fourth* industrial revolution of cyber-physical systems is the consequence of incremental technological advancement since steam-powered manufacturing mechanization (the *first* industrial revolution) to the *third* industrial revolution of computer-driven process automation. Different control systems, Programmable Logic Controller (PLC), Supervisory Control and Data Acquisition systems (SCADA) and Distributed Control Systems (DCS), in different industrial sectors (chemical, power distribution, manufacturing) have been developed to efficiently achieve desired sector-specific outcomes with lower monetary cost and almost-to-none hazardous incidents. The network of interconnected equipments employed to monitor and control the industrial operations constitutes Industrial Control Systems (ICS) network or industrial network or Operational Technology (OT) network. It differs from traditional Information Technology (IT) network, or office network, in term of connectivity requirements of end users for general purpose transfer of information through Internet. However, the modern industrial communication infrastructure incorporated IT networking strategies and related protocols for shop-floor to enterprise end user access, gaining finer control on the industrial process at the cost of vulnerability to cyber threats. As a result, cybersecurity is needed to protect the industrial assets from malicious intents of the threat actors.

The modern industrial communication network can be classified in different sub-networks, as shown in fig. 1.1. Each sub-network is characterized by its role in industrial process operation, where different sub-network specific industrial components are communicating via underlying networking infrastructure.

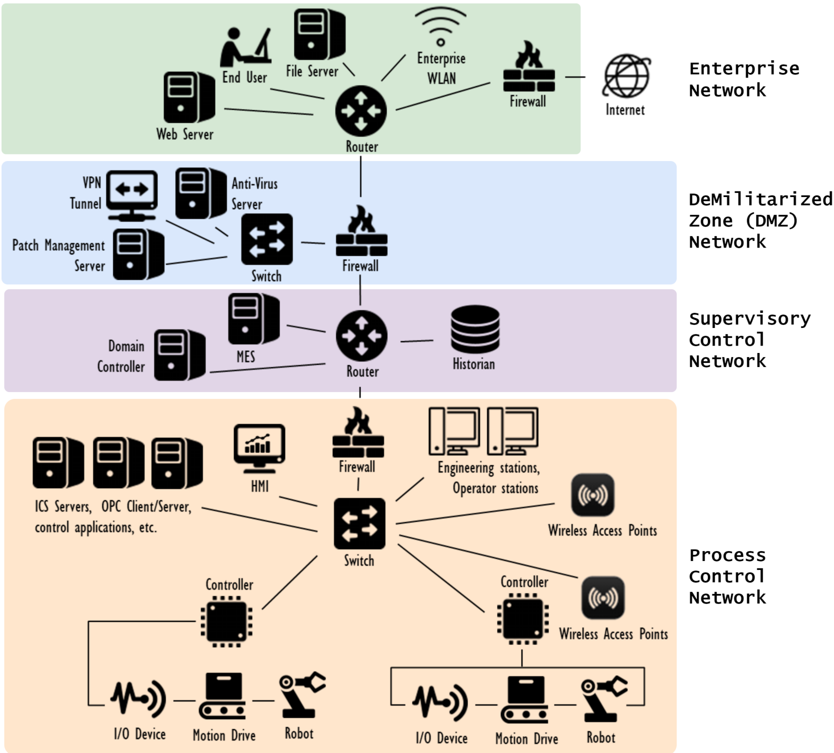

Figure 1.1: Industrial Communication Architecture.

Process Control Network: At the core of industrial operation, sensors, actuators, transmitters and I/O devices are orchestrated to realize the industrial process, and controllers such as PLCs, DCS execute the control logics of the industrial process.

Supervisory Control Network: Maintenance and supervision of industrial process, on-demand change of process control logic, collection and storage of process values are performed through Human-Machine Interfaces (HMI), Manufacturing Execution Systems (MES), Data Historians, Engineering and Operator Workstations.

DeMilitarized Zone (DMZ) Network: Following the industrial cybersecurity standards NIST SP 800-82 [Sto08] and IEC 62443 [COD13], the OT network should be separated from the IT network to limit access to process information and modification of software infrastructure in OT. Patch management servers, anti-virus servers and remote access management servers are DMZ's constituents.

Enterprise Network: The corporate-level business decisions are driven through the data collected from OT networks supported through Enterprise Resource Planning (ERP), user management servers, file servers and IT related functional components. The access to Internet is restricted beyond the network.

Firewalls to filter network traffic across different sub-networks are placed at the sub-network edges. Network switches are used to relay information within the sub-network, and routers for network communication across sub-networks.

Introduction of IT technology into the OT networks has started to blur distinction between office and industrial networks. However, there are fundamental differences in requirements and characteristics of office and industrial networks [Gal12], summarized in table 1.1 and expanded as follows:

Primary Function. Industrial networks are employed to control physical equipments which are directly affecting the physical world. On the other hand, office networks are transferring data for processing digital information such as files, audio-visual information, logistical data, etc.

Application Domain. Wherever there is machinery that requires monitoring and controlling, an industrial network is commissioned. The industrial domains where such a network is utilized are discrete manufacturing (automotive industry), process control (electricity generation), utility distribution (water distribution), transportation (railways) and embedded system (car's control network). IT networks find their end users in corporate and home environments.

Table 1.1: Characteristic differences of Industrial and Office networks.

	Industrial Networks	Office Networks
Primary Function	*Control of physical equipment*	*Data processing and transfer*
Application Domain	*Manufacturing, utility distribution*	*Corporate and home environments*
Failure Severity	*High, industrial disasters*	*Low, information unavailable*
Real Time Availability	*High, response time 250 μs - 10 ms*	*Moderate, response time 50+ ms*
Determinism	*High, low jitter*	*Low, high jitters dropped*
Traffic	*Periodic and aperiodic*	*Aperiodic, „best effort“*
Spatio-Temporal Consistency	*Required*	*Not required*
Network Protocols	*PROFIBUS, PROFINET,..*	*TCP/IP, UDP, FTP,..*

Failure Severity. The impact of failure in industrial networks is high leading to Human, Societal and Environmental (HSE) hazards. For example, the industrial disaster of Fukashima Daiichi in 2011. When an IT network is down, the unavailability of information on a webpage or web services has relatively less severity to the industrial disaster.

Real Time Availability. Industrial applications such as motion control require data to be transmitted and received within the duration of $250\ \mu s - 1\ ms$, and less stringent processes require $1\ ms\ -\ 10\ ms$ of response time. Office networks have moderate response time requirement of $\geq 50\ ms$.

Determinism. The variance in the response time of a signal is referred as *jitter*. Industrial networks require low jitters for real-time availability, implying data is sent and received in highly deterministic manner. Office network applications are not usually impacted by jitters. However, Voice over Internet

Protocol applications are exceptions that require low jitter, and higher jitter packets are dropped without affecting considerable its intended purpose.

Traffic. Process data is exchanged periodically in industrial networks, and diagnostic data (such as alarm notifications) are transmitted in aperiodic manner. The IT networks transmit data with "*best effort*" delivery service.

Spatio-Temporal Consistency. In industrial networks, the data needs to be transmitted with temporal consistency in an ordered manner. No such guarantee is required in IT networks.

Network Protocols. Industrial protocols satisfying above mentioned requirements have been in development since 1980s. Serial-based fieldbus protocols such as *Process Field Bus (PROFIBUS)* and *Modbus* were developed in parallel to replace traditional two-wire signalling of control loops. Ethernet-based industrial protocols such as *Process Field Net (PROFINET)*, *Modbus Transmission Control Protocol (Modbus/TCP)* and *Ethernet Industrial Protocol (EtherNet/IP)* are the result of incorporating Ethernet technology for fast real-time industrial communication. The *Internet protocol suite* protocols (*Transmission Control Protocol (TCP)* and *Internet Protocol (IP)*), *User Datagram Protocol (UDP)* and *File Transfer Protocol (FTP)* are some of IT protocols used for transmitting network application-specific data between end users.

1.1 Cybersecurity in Industrial Networks

The security objective for IT networks revolve around *confidentiality*, *integrity* and *availability* of information, the *CIA* triad. The IT protocols were developed while considering these security objectives resulting in secured protocols such as *Transport Layer Security (TLS)*, *Secure Shell Protocol (SSH)*, etc. For industrial networks, the *availability* of information in real-time is of the highest priority. The *CIA* triad didn't suffice to the requirements of industrial network security, hence, its security objectives are categorized as basic foundational requirements in IEC 62443 [COD13]: *Access Control (AC)*, *Use Control (UC)*, *Data Integrity (DI)*, *Data Confidentiality (DC)*, *Restrict Data Flow (RDF)*, *Timely Response to Event (TRE)* and *Resource Availability (RA)*.

Table 1.2: Summary of cyber incidents targeted at industrial systems in last decades.

Threat	Target	Exploit
STUXNET (2010)	Uranium enrichment facility in Iran	Siemens **PLC** to control nuclear enrichment centrifuge speed
BLACKENERGY 2 (2010)	Western militaries, governments, industrial sites, etc.	Internet connected **HMI** to learn **industrial process** and graphical representation of industrial network
Dragonfly/HAVEX (2013)	Multiple sectors (energy, aviation, etc.) primarily the United States and Europe	OPC protocol to map out **industrial network**
Ukraine Cyber Attack/ BLACKENERGY 3 (2015)	Three power companies in Ukraine	Gained access to corporate network and pivot into **SCADA** networks
CRASHOVERRIDE (2016)	Transmission level substation in Ukraine	Sophisticated attack learning **industrial process**, leveraging **OPC protocol** and **HMI** against grid operations
TRITON (2017)	Petrochemical company in Saudi Arabia	Gained remote access to Triconex Safety **controllers** and entered it in failed safe state leading to automatic shutdown of the **industrial process**

Fieldbus protocols were designed to make information available as quickly as possible for industrial process operations without any consideration to security of network communication. "*Security by Obscurity*" or "*Air Gapping*" defined the security policies of automation community as the communication was restricted to industrial shop-floor and never interacted with outside the industrial system's barrier. When Ethernet technology paved its way into industrial networks, it brought along network vulnerabilities of IT systems. Exploitation of these vulnerabilities resulting in the violations of security objectives is termed as a threat or an attack. Common types of attacks are Distributed Denial-of-Service (DDoS), Man-in-the-Middle (MitM), eavesdropping, virus/trojan/worm and physically breaking into a system [Dzu05]. Advanced Persistent Threats (APT) are the highly skilled adversaries repeatedly targeting the industrial systems for espionage or malfunctioning the underlying process to cause monetary, social, environmental and personnel destructions. Some of the cyber incidents in last two decades [Lee17] are summarized in table 1.2. From the exploits of the table 1.2, it can be observed that the vulnerabilities of industrial communication network and its constituent infrastructure are being targeted to achieve adversarial goals.

A network security design principle of "*Security in Depth*" has been formulated as the countermeasure strategy to detect and analyse security vulnerabilities in industrial networks. At the perimeter of each layer in multi-layered industrial communication, the employed security measures isolates the subsequent layer from external threats. The outermost layer restricts the access to industrial networks to authorized personnel through Firewalls and Virtual Private Networks (VPN). DMZ tightens the security between the office and the industrial networks with shareable equipments placed in this layer to avoid direct access to industrial networks. To prevent unauthorized physical access to physical equipments, '*hardening the equipment*' strategy is followed through password-protection on every device, blocking unused ports on switches and routers, periodic update of installed software and operating system. In order to maintain data *confidentiality*, *integrity* and *authentication* of the communication channels, application of cryptographic algorithms to industrial network security is the hot area of research.

As the security mechanisms for network security evolve, the older industrial equipments are mostly incapable of deploying advanced measures. To counteract the scenario, an effective security policy needs to be formulated which requires analysis of network security policies and network infrastructure, including network design and protocols, for detecting and mitigating the vulnerabilities. Intrusion Detection Systems (IDS) address all these requirements through monitoring the network to detect known attack patterns and/or deviations in the monitored behaviour of system or network.

IDS technologies can be differentiated based on the types of monitored events and analysis techniques/methodologies employed over the monitored events to detect cyber threats. The most common IDS are Host-based Intrusion Detection Systems (HIDS) and Network-based Intrusion Detection Systems (NIDS). NIDS monitors network traffic and application protocol activity in network segments. HIDS monitors characteristics and events occurring within hosts or devices on which they are deployed, such as HMI, SCADA servers and Operator/Engineering workstations. The intrusion detection methodologies can be categorized as: signature-based, anomaly-based and stateful protocol analysis. Signature-based detection compares known threat signatures to observed events through comparison operations such as string comparison. It is ineffective to multi-event attacks and unknown threats. Anomaly-based methods are effective in detecting unknown attacks using statistics, expert-knowledge and machine learning methods to compare normal activity profile against observed events to identify significant deviations. If the complex and real world activity of industrial operations are not meticulously captured, it results in high false positives. Stateful protocol analysis compares predetermined profiles based on network protocol standard against observed protocol activity to detect deviations. The analysis requires constant updating as and when protocol standard specification changes.

1.1.1 Anomaly Detection in Industrial Networks

Within an industrial network, the network elements and segments are connected by network switches, and firewalls deployed at the edges of segments aid in filtering the flow of authorized traffic across. Network monitoring collects and analyses data from the network segments (network traffic) and elements (switch and firewall logs) to provide network transparency in terms of participants with their networking related properties and communication links between them. This information creates the foundation on which anomalies are detected by anomaly detection systems. It learns the network's topology, the communication links, time-related behaviour and communication content to flag deviations as anomalies. Germany's Federal Office for Information Security (BSI) in its cybersecurity recommendations on production networks [BSI19] outlined anomalies in industrial networks and related categories of feature requirements for anomaly detection system. The selected feature requirements of detecting the outlined anomalies in an industrial network are summarized as follows:

(1) Category A: general requirements

- overview of all devices communicating in the network,
- identification of protocols used in the network,
- identification of all the communication links in the network,

(2) Category B: unusual or exceptional activities in an ICS network

- identification of new devices in the network,
- identification of communications between two devices that was previously non-existent,
- identification of new protocols or changes in protocol among individual components,

(3) Category C: abnormal events in logs typical of production environments

- identification of ICS-specific function codes that have not previously been used,
- the ability to determine whether an access attempt (e.g. read/write) pertains to an address that is not normally used by the device at hand,

(4) Category D: unusual changes in process data (sensor data, control data, etc.)

- identification of changes in time-related behaviour,
- identification of deviations within defined value ranges

The BSI recommendation highlights the difference between "*passive*" and "*active*" solutions for network monitoring and anomaly detection. The *passive* system collects data passively using network or wire taps, and has no impact on data or time-sensitive behaviour of the network. An *active* system creates requests to network switches or devices thus generating data in the industrial network's traffic. The system needs to be configured to ignore its own data for the analysis. In addition, the change in time-sensitive behaviour caused due to additional data transmission needs to be factored in while deploying such a system.

1.2 The Self-learning Approach

Industrial systems are commissioned for longer duration amounting to decades. The technological advancements in industrial security mechanisms are not easily deployable to non-compliant industrial network infrastructures. The industrial components can't be always upgraded to communicate with latest industrial protocols as it is an expensive process for the plant owner. Nevertheless, it is essential to design the security of industrial network with all the available information. When access to network infrastructure information, such as asset inventory and network policies, of an existing industrial system is unavailable, designing the security policies or configuring the countermeasures such as NIDS is challenging. An empirical solution is to self-learn the network infrastructure information (topology, assets and communication links) and the *spatio-temporal* behaviour from passive monitoring of industrial network traffic, in conjunction with an anomaly detection system to detect anomalies. However, in the absence of industrial process information, learning the process behaviour from the network traffic for anomaly detection is a challenge.

A self-learning framework for passive network monitoring and anomaly detection in industrial networks (refer fig. 1.2) consists of following components:

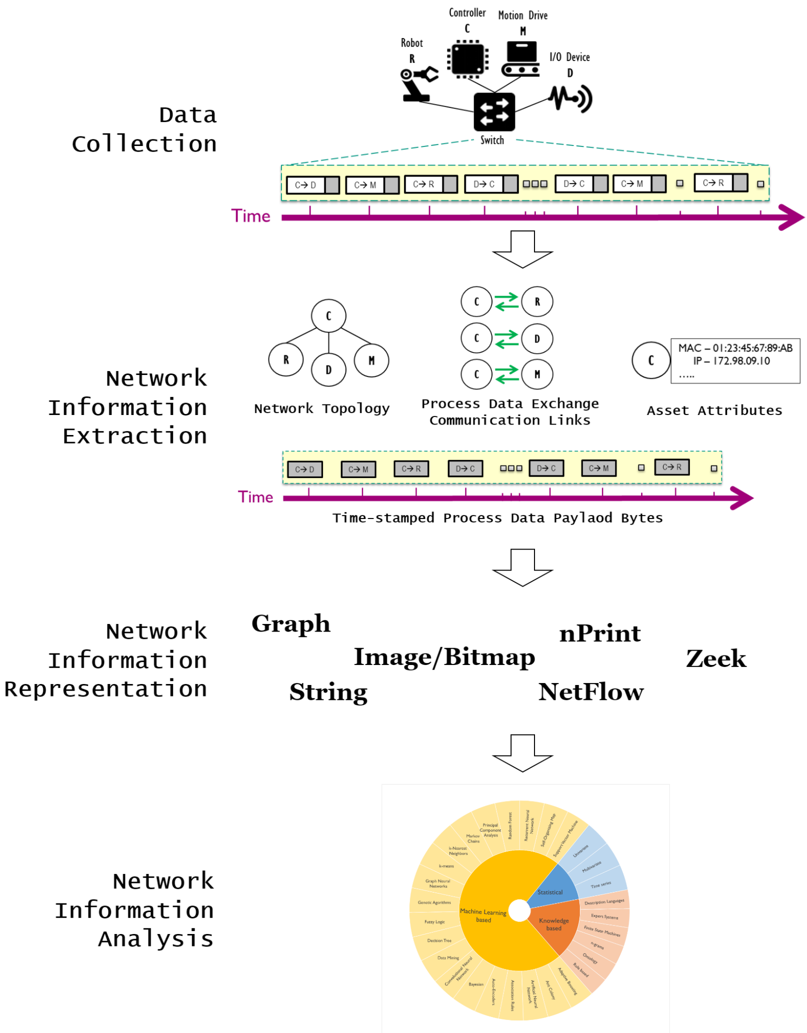

Figure 1.2: A Self-learning pipeline for network traffic analysis.

(a) **Data Collection and Network Information Extraction.** The unencrypted industrial network traffic is collected through network taps on wires or mirror ports on network switches. The modern industrial protocols are based on Ethernet and the data is transmitted across as series of Ethernet frames between the participants. Each frame is dissected to extract information from *structured Header* and *unstructured Data* parts for network monitoring or NIDS and process data analysis or condition monitoring, respectively, as shown in fig. 1.3. Different types of information or *features* can be summarized from dissected frames: flow statistics (eg. throughput (*Bytes/second*)), protocol transactions (eg. TCP Handshake), process values (eg. Modbus registers), and topological information (eg. sender and receiver Media Access Control (MAC) addresses).

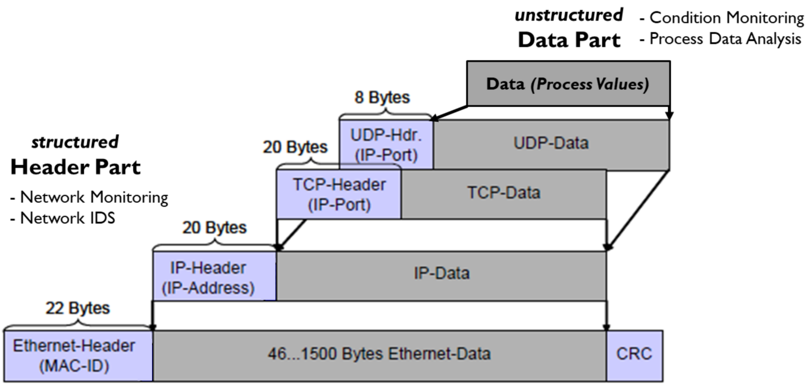

Figure 1.3: Dissection of an Ethernet Frame.

(b) **Network Information Representation.** The representation of extracted network information depends on the requirements of the analysis method. NetFlow was introduced by Cisco to represent network flow information capturing ingress and outgress quantitative features such as network throughput rate (*Bytes/second*)). String-based representation is the staple for n-gram analysis methods [Wre13] and other string-based intrusion detection methods such as ZOE [Wre18]. Graph representation have been used for graph-based analysis methods such as GNN [Bus21, Puj21]. The network information can be represented as images or bitmaps for image analysis based intrusion detection methods [Kim05]. There are also proprietary representation used by respective methodologies such as nPrint [Hol21] and Zeek/Bro [Pax99].

Figure 1.4: Summary of Anomaly-based Intrusion Detection System methodologies [Hin18].

(c) **Network Information Analysis.** After network information is represented in an analysis method's required format, it forms the basis for learning the network's assets, communication links, time-based behaviour and process communication. The deviations from the learnt behaviour are reported as *anomalies*. Hindy et. al. [Hin18] have summarized the different categories of anomaly-based intrusion detection techniques as *statistical*, *knowledge-based* and *ML-based*, shown in fig. 1.4. Univariate models, multivariate models and time-series analysis are employed as statistical methods for anomaly-detection. For knowledge-based anomaly detection, finite state machines, description languages, expert systems, n-grams and rule-based methods are grouped together in this category. ML-based techniques encompasses data mining methods along with neural networks, deep learning methods, support vector machines, decision trees, bayesian networks, markov models, genetic algorithms, etc.

1.3 Research Objectives and Contributions

An industrial system's behaviour consists of two components: (a) the networking operations beginning at the start-up of the system until the process data exchange begins, and (b) the periodic and deterministic industrial process. The network information representing the two aspects of an industrial system's behaviour are embedded in the multiple protocols communicating over the networking infrastructure. One or multiple protocols are employed at different stages of the networking operations to configure the network assets for process data exchange.

To self-learn an industrial system's behaviour from network traffic, the network information needs to be systematically extracted while enumerating the networking operations. After the network is made transparent for downstream analysis, the system behaviour needs to be learnt from the extracted information for anomaly detection. The research questions that arises to self-learn an industrial system's behaviour from the passively monitored network traffic and the empirical solutions developed to answer the research objectives are summarized in this section.

In the quest for finding the solutions to research questions, a *PROFINET* based industrial system is used for developing and evaluating the proposed solutions. *PROFINET* has 18% market share in the industrial networks that are installed globally in 2021 as compared to 17% *EtherNet/IP* [NID21]. Additionally, *PROFINET* is the leader of Industrial Ethernet technology in the European market which concluded its selection as the industrial system under consideration for the presented work. A *PROFINET*-based scaled-down industrial system with real industrial components and fully functional networking infrastructure, labelled as *Festo Demonstrator*, developed at IT-Security Laboratory, Fraunhofer IOSB [Pfr16, 20] is employed for the reported analysis. The network attacks targeted at the *Festo Demonstrator*'s underlying network and process operations are scripted, executed and resulting anomalies are passively captured from the network traffic. The details of the demonstrator and network attack scenarios are provided in section 2.5.

1.3.1 Industrial Network Transparency

The networking operations such as neighbourhood detection and connection establishment that an industrial communication network undergoes, contains all the required information for network monitoring. The industrial network's topology, communication relations, assets and protocol data being exchanged during the industrial process operation are the network information to be extracted from the network traffic passively. For extracting the network information in a self-learning scenario, detecting the different industrial operations from start-up to process data exchange from the network traffic in a systematic way is required and defines the research objective for network transparency.

Research Question 1: *How to extract information of an industrial system from its network traffic?*

An industrial system's process is conceptualized and engineered outside the networking operations in an engineering tool. The devices are configured as per the requirements of process and the process control logic is loaded onto the controllers. As soon as the system starts up, the devices are communicating with each other to establish their neighbours in the vicinity and communication links between master devices and their corresponding slave devices. In chapter 3, different industrial networking operations are mapped to their industrial operation counterparts. *PROFINET*'s specifications are followed to correctly map network protocols to detect networking operations and their constituent stages. For this, the *Python*-based *Scapy* framework [Bio22] is utilized to dissect network packets in order to extract relevant information for detecting networking operations and populate different network information: *network topology*, *assets and their attributes*, *communication links established between master and slave* (referred as *PROFINET Connection*), *time-stamped process data payload bytes*.

Contribution 1: A Python-based framework that passively captures the network traffic from *PROFINET*-based system and extracts relevant information to make the industrial network transparent for analysis.

1.3.2 Industrial Operations Behaviour

The industrial operations executed in an order create the foundation for process data exchanges realizing the underlying intended process. Enumerating these operations through the analysis of multiple protocol communications observed in the traffic and tracking their executions helps to define the industrial system's operation behaviour. Addition of a device to the network and device's attempt to establish communication for process data exchange triggers execution of network operations. Monitoring the valid operations of devices, communication links or the industrial system would detect the adversarial actions in the context of employed Industrial Ethernet technology's network protocol communication specifications. The research objectives for self-learning an industrial system's operation behaviour are: *(a)* selecting the appropriate network information extracted from network frames for operation enumeration, *(b)* representing the executions of operations and their constituent stages to learn the operation's behaviour, and *(c)* selection or development of an analysis method to detect deviations.

Research Question 2: *How to enumerate and track an industrial system's operations from network traffic?*

In chapter 3, the network information made available through the developed network transparency solution are utilized to enumerate different industrial operations whenever they occur. An industrial system's operation behaviour is considered at device-level, connection-level and system-level to track operational state changes in devices, established process exchange communication links and the overall system. *Finite State Machines (FSM)* are conceptualized for each device, connection and industrial system, where nodes represent the stages of industrial operations and the edges are the transitions that are triggered by the events observed in the extracted network information from the traffic. The self-learning methodology is summarized in table 1.3. A *Python*-based framework that extracts the network information from *PROFINET*-based industrial system's network traffic to instantiated appropriate FSM models (Device, Connection or System) and track the industrial operations is developed as *PROFINET Operations Enumeration and Tracking (POET)*. In section 3.5.1, POET is employed to detect anomalies triggered by a network attack targeted at *Festo Demonstrator*'s device *Turntable-Motor*. The attack is executed through valid *PROFINET Discovery and Configuration Protocol (PN-DCP)* exchanges, resulting in invalid *PROFINET* operation transition for the device leading to successful detection and reporting by POET.

Contribution 2: Graph-represented Finite State Machine models for valid industrial operations and their transitions at device-level, connection-level and system-level.

Contribution 3: *PROFINET* Operations Enumeration and Tracking (POET) as a framework to enumerate *PROFINET* operations from the events embedded in network information and track the operation transitions to report invalid transitions as anomaly, acting as a Protocol-analysis based Network-based Intrusion Detection Systems (NIDS) for *PROFINET*.

Table 1.3: Summary of proposed behavioural analysis methods.

	Extracted Network Information	**Network Information Representation**	**Network Information Analysis**
Industrial Operations Behaviour	*Network Topology* *Asset and attributes* *PROFINET Connection*	*Finite State Machines for PROFINET operations (section 3.4)*	*Finite State Machines (section 3.4)*
Spatio-temporal Process Behaviour	*Time-stamped Process Data Payload Bytes*	*String-represented Process Payload Byte Profiles (section 4.2.3)*	*- Regular Expressions-coupled Suffix Tree (section 4.3)* *- Machine Learning models: LOF, IF, OC-SVM (section 4.4)*
		Graph-represented Graph Snapshots (section 5.4)	*Anisotropic Graph Neural Networks: MPNN, GatedGCN, GT (section 5.5)*

1.3.3 Industrial Process Behaviour

The industrial process behaviour is defined by the periodic, deterministic and ordered exchange of process parameters between industrial components. The *spatio-temporal* characteristics of the process data exchange represents the industrial process behaviour, where a finite set of process values are exchanged in a redundant manner. An adversary with access to the information on underlying industrial process of an industrial system could tamper either the process parameter values or delay the network frames to exploit the data integrity (DI), time to response (TRE) and resource availability (RA) security objectives of the industrial system. In a self-learning analysis approach, there are no insights into the semantics of process values being exchanged as the information on the programmed industrial process is unavailable. However, the *spatio-temporal* characteristics of finite set of payload bytes extracted from the process data frames are representative of embedded process values. The representation of payload bytes with temporal information of their occurrences to learn the *spatio-temporal* characteristics and detecting their deviations as anomalies are the research objectives for self-learning an industrial system's process behaviour.

Research Question 3: *How to represent the spatio-temporal characteristics of an industrial system's process?*

Research Question 4: *How to learn the spatio-temporal characteristics of an industrial system's process for anomaly detection from the network traffic?*

The industrial process is expressed as the constituent sub-processes of an industrial system. A sub-process represents the process data communication link (Connection) established between a pair of master and slave devices exchanging a particular process parameter of the overall process. The overall interplay of different process parameter exchanges in the constituent sub-processes drives the industrial process. With the coarse and fine granular perspectives of an industrial process, in this thesis two different representations for the *spatio-temporal* characteristics and related process data analysis methodologies have been proposed in chapter 4 and chapter 5, summarized in table 1.3.

In chapter 4, the ordered process data payload bytes are represented as a *String*, where the finite set of payload byte strings of every sub-process are encoded as alphabets. The alphabet-encoded payload bytes, when sequentially arranged, define the encoded network traffic and every pair of adjacent alphabet-encoded payload bytes defines a transition. The temporal information associated with process payload bytes is utilized to define the transition intervals for each transition observed.

A *Python*-based framework called *Payload Bytes Profiling (PBP)* extracts the process data payload bytes, encodes them in alphabets and profiles the *spatio-temporal* characteristics for sub-processes and the industrial process is developed as the empirical solution. The *transition profile* of PBP captures the finite set of alphabet-encoded payload bytes representing the production cycle which is extracted using Regular Expression-coupled Suffix Tree algorithm, refer section 4.3. The transition intervals for each transition pair in the transition profile is modeled using non-parametric Machine Learning (ML) models Local Outlier Factor (LOF), Isolation Forest (IF) and One-Class Support Vector Machines (OC-SVM), and stored as the *interval profile* in PBP of a sub-process or the overall process, refer section 4.4. The combination of *transition* and *interval profiles* in a PBP of a sub-process/process models the *spatio-temporal* characteristics and detects the deviations as anomalies, refer section 4.4.3 The PBP framework successfully detected the anomalies triggered by changing the process values in *Festo Demonstrator*'s sub-process to move the *Lower Belt* device in opposite direction, refer section 4.6.2. In addition to anomaly detection, the production cycle of *Festo Demonstrator*'s industrial process is successfully detected through its transition profile extraction from the encoded traffic using PBP framework, refer section 4.6.1.

Contribution 4: Payload Bytes Profiling (PBP) as a systematic framework to represent the *spatio-temporal* characteristics of an industrial system's process and its constituent sub-processes as spatial (*transition profile*) and temporal (*interval profile*) profiles from passively monitored network traffic.

Contribution 5: A methodology to extract the periodic process data exchange at process and sub-process granularity with Regular Expression-coupled Suffix Tree algorithm from *String* represented network traffic.

Contribution 6: A methodology based on non-parametric Machine Learning (ML) models (Local Outlier Factor, Isolation Forest and One-Class Support Vector Machines) to capture intervals between process payload bytes as the *temporal* characteristic of an industrial system's sub-process/process.

In chapter 5, the ubiquitous *Graph* representation of an industrial network to learn the *spatio-temporal* characteristics of the underlying industrial process is proposed. The graph structure represents the network topology with *nodes* being the industrial components and the *edges* being the communication link between the components. Especially, for capturing the process behaviour in the graph, the edge represents the communication link established for process data exchange. The network traffic encoding scheme from chapter 4 is utilized to represent the process data exchanges as a string of alphabet-encoded process payload bytes with temporal information. In addition to the transition interval, the differences in the *Cycle Counter* values of transitioning payloads is utilized to capture the network behaviour extracted from the *structured Header* part of network frames. To represent the *spatio-temporal* characteristics of industrial process, a *graph snapshot* over a fixed time window on the traffic is utilized, refer section 5.4. For each sub-process represented by the edge of the graph snapshot, the payload byte transitions along with interval and cycle counter differences are binary encoded.

In order to learn the *spatio-temporal* characteristics represented in graph snapshots, Graph Neural Networks (GNN) models are employed to compute embeddings for the nodes, edges and overall graph for downstream ML tasks such as *node/edge/graph classification*, *link prediction*, etc., refer section 5.3. Three different anisotropic GNN architectures (Message Passing Neural Network (MPNN), Gated Graph Convolutional Network (GatedGCN) and Graph Transformer (GT)) are evaluated for learning the process behaviour from the graph snapshots and anomaly detection is modeled as the *graph classification* task to classify each graph snapshot as *normal* or *anomalous.* An isotropic GNN, Graph Convolutional Network (GCN), is used as the baseline, refer section 5.5. The learnt GNN architectures are employed to detect anomalies in the process traffic, triggered by changing the process values in *Festo Demonstrator*'s sub-process to move the *Lower Belt* device in opposite direction. The anisotropic GNNs are successful in detecting the anomalies in comparison to isotropic GNN architecture.

Contribution 7: A *Graph* representation of an industrial system's *spatio-temporal* characteristic process as *graph snapshots* over the passively monitored network traffic.

Contribution 8: Demonstration of anisotropic Graph Neural Networks (GNN) architecture usage to learn the process behaviour with *graph classification* as the ML task to classify *graph snapshots* as *normal* or *anomalous.*

1.4 Thesis Outline

In *Chapter 2*, a brief introduction to industrial communication protocols, cyber threats in industrial networks, IDS taxonomy for industrial networks and publicly available datasets for industrial cybersecurity are sketched. The description of *Festo Demonstrator* and implemented attack scenarios are also provided.

Chapter 3 details the networking operations of a *PROFINET*-based industrial production system detected from its network traffic to populate the network infrastructure information and make the industrial network transparent for further analysis. The *POET* framework that enumerates and tracks the *PROFINET* operations using different Finite State Machine models at different granularities and its usage for anomaly detection are outlined.

In *Chapter 4*, the self-learning approach for process behaviour analysis based on *String* representation of the process data extracted from the network traffic is presented. The details of *PBP* framework with *spatial* and *temporal* characteristic modeling methods for anomaly detection are described.

Chapter 5 describes the usage of anisotropic GNN architectures to self-learn the *spatio-temporal* characteristics of an industrial process which is graphically represented as *graph snapshots.* The *graph snapshots* are classified as either *normal* or *anomalous.*

In *Chapter 6*, the contributions of the dissertation ensued from the reported solutions to the research questions with discussion and an outlook to improve the solutions are provided.

2 Preliminaries

In this chapter, the foundational information required for the reported research work are provided.

2.1 Industrial Communication Protocols

In 1970*s*, the introduction of new technologies such as programmable microcontrollers and digital signal processors led to the replacement of traditional two-wire signalling technique for analogue control loops between connected industrial equipments with digital controllers. The *International Standards Organisation (ISO)* created the seven layered communication model *Open System Interconnection (OSI)*, which facilitated creation of multiple communication protocols and services. As per the different communication characteristic requirements, the standard OSI reference model is reduced to protocol stacks: *TCP/IP* stack, and *Manufacturing Automation Protocol/Enhanced Performance Architecture (MAP/EPA)* stack, refer fig. 2.1. The reduction is driven by the real-time availability and deterministic communication requirement of industrial network communication. The reduced protocol stacks reduced the time delays incurred due to passing of information between layers and its processing at each layer [Gal12].

Serial-based fieldbus protocols built on the MAP/EPA model replaced expensive point-to-point cabling of devices to a central control room, and brought at forefront the concepts of decentralization, modularity of industrial systems and communication between intelligent devices for process and diagnostic information exchange. *PROFIBUS*, *Controller Area Network (CAN)* and *Foundation Fieldbus (FF)* are some of the earliest fieldbus protocols which were independently developed in parallel for automation. The *International Electrotechnical Commission (IEC)* and *Instrumentation Society of America (ISA)* organizations jointly defined a standard for fieldbus protocols IEC 61158 to standardize independent fieldbus developments.

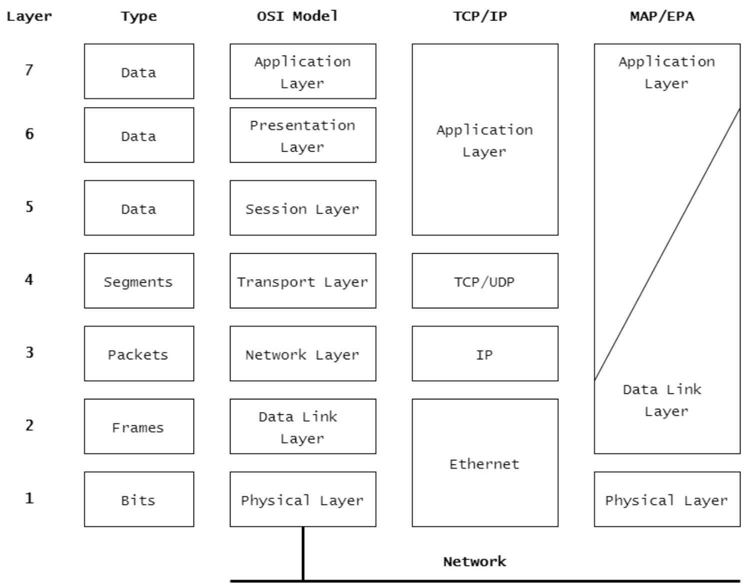

Figure 2.1: An overview of network communication models.

The incorporation of Ethernet technology into industrial networking introduced Ethernet-based industrial communication protocols. Fast data transfer rates of Ethernet accelerated *Real-Time Ethernet (RTE)* protocols for time-critical industrial applications such as motion controls. Ethernet also introduced switched networks to replace older hub based networks for efficient and error-less data retransmission in the industrial automation domain. *PROFINET*, *EtherCAT*, *EtherNet/IP*, *Modbus/TCP* are some of the RTE industrial protocols within the IEC 61784 standard for the Industrial Ethernet protocols.

Different application domains such as factory automation and motion controls have different real-time requirements which need different implementations of the *TCP/IP* protocol stack to achieve the determinism. These different RTE implementations, refer fig. 2.2, are categorized based on transmission time [Dec05, Dan14] as follows:

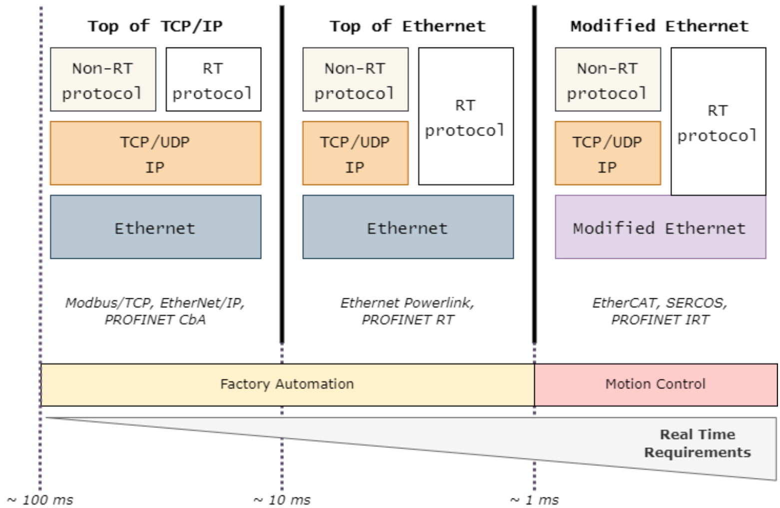

Figure 2.2: An overview of different Real-Time Ethernet/Industrial Ethernet implementations.

On top of TCP/IP. Human centric systems such as process monitoring and engineering systems need network data transmission time around 100 *ms* for observation. The *application layer* is responsible for scheduling communication to meet the requirements. This allows communication outside the network boundaries with remote devices incurring non-deterministic delays. The scheduling devices need to be equipped with adequate resources to handle the delays. Industrial protocols *Modbus/TCP* and *EtherNet/IP* are based on this RTE implementation.

On top of Ethernet. For process control systems, the timing requirements are below 10 *ms*. Modifications at the *application layer* to use standard packets and at the *transport layer* to use custom ethertypes for real-time communication is performed. The network devices must have the knowledge of custom protocols to prioritize the custom ethertypes within the network. Within the

RTE implementation, *Ethernet Powerlink* and *PROFINET Real-Time (RT)* are the widely popular protocols.

Modified Ethernet. Motion control applications require a cycle time $< 1\,ms$, which can be achieved only when the *data link layer* is modified to apply mechanisms and infrastructure to allow real-time communication. Customized hardware with switching functionality is employed to achieve highly deterministic timing requirements. *PROFINET Isochronous Real-Time (IRT)*, *EtherCAT* and *Serial Real-time Communication System (SERCOS)* are the Industrial Ethernet protocols based on the RTE implementation.

Time Sensitive Networking (TSN) is a set of IEEE 802 standards [21d] to make Ethernet deterministic by design. It aims to develop mechanisms that guarantee determinism and bounded latency in high throughput networks operating at $10\,Mbps$, $100\,Mbps$, $1\,Gbps$ or $10\,Gbps$. It offers real-time capability in industrial networks with $gigabit$ Ethernet which is restricted in IEC 61784 Industrial Ethernet protocols. The TSN standards offer specifications for *Frame Preemption* (IEEE 802.1Qbu), *Scheduled Traffic* (IEEE 802.1Qbv), *Ingress Policing* (IEEE 802.1Qci), *Seamless Redundancy* (IEEE 802.1CB), *Time Synchronization* (IEEE 802.1AS), *Stream Reservation* (IEEE 802.1Qcc) and many more [Bra21, 21d]. The existing Industrial Ethernet protocols such as *PROFINET* and *EtherCAT* are incorporating TSN in their protocol stacks to support TSN technology for real-time networking [21c, 21b].

The *Open Platform Communications Unified Architecture (OPC UA)* framework merged with TSN, referred as *OPC UA TSN*, is the next-generation solution to integrated communication and data infrastructure requirement for seamless merging of IT and OT applications in real-time [Tri21, Hum22]. The OPC UA PubSub communication model introduced by the OPC Foundation in IEC 62641-14 builds the foundation for secure communication across devices in field level to MES/ERP and further into the Cloud. Using TSN as the communication protocol at *OSI layer 2*, the real-time deterministic control behaviour can be achieved. The TSN-enabled Switches are already available and realizing the benefits of TSN. Different automation manufacturers have began to integrate *OPC UA TSN* into their product portfolios such as *PROFINET@TSN* and *Sercos over TSN*.

2.2 Cyber Threats in Industrial Networks

Initial Access	Execution	Persistence	Privilege Escalation
Drive-by Compromise (DbC)	Change Operating Mode (COM)	Modify Program (MPgm)	Exploitation for Privilege Escalation (EfPE)
Exploit Public-Facing Application (EPFA)	Command-Line Interface (CLI)	Module Firmware (MF)	Hooking (H)
Exploitation of Remote Services (EoRS)	Execution through API (EtA)	Project File Infection (PFI)	
External Remote Services (ERS)	Graphical User Interface (GUI)	System Firmware (SF)	
Internet Accessible Device (IAD)	Hooking (H)	Valid Accounts (VA)	
Remote Services (RS)	Modify Controller Tasking (MCT)		
Replication Through Removable Media (RTRM)	Native API (NA)		
Rogue Master (RM)	Scripting (S)		
Spearphishing Attachment (SA)	User Execution (UE)		
Supply Chain Compromise (SCC)			
Transient Cyber Asset (TCA)			
Wireless Compromise (WC)			

Figure 2.3: The MITRE ATT&CK for ICS Matrix. (Part 1)

MITRE ATT&CK for Industrial Control Systems (ICS) [Ale20] is a knowledge base of cyber adversary behaviours targeted at the ICS technology domain and it categorizes adversary's attack life cycle into phases with targeted assets and systems. It describes an adversary's tactical objective for performing an action, the tactical actions and instances of adversary's techniques as the *tactics*, *techniques* and *procedures* (TTP). The relationships between tactics and techniques are organized in the ATT&CK Matrix, as shown in fig. 2.3 - fig. 2.5. As of *February* 2022, there are 78 techniques categorized into 12 tactics in ATT&CK for ICS. The adversary's tactics are categorized in the order of attack life cycle phases as `Initial Access`, `Execution`, `Persistence`, `Privilege Escalation`, `Evasion`, `Discovery`, `Lateral Movement`, `Collection`, `Command and Control`, `Inhibit Response Function`, `Impair Process Control` and `Impact`.

An attack sequence of real threat groups usually contains multiple techniques where the attacker moves from left (`Initial Access, ..`) to right (`.., Impact`) in the ATT&CK Matrix. The table 2.1 summarizes the MITRE techniques employed by the threat groups that executed APTs *Stuxnet* [22d], *Industroyer* [22a] and *Triton* [22e]. The initial stages of these APTs involves IT infrastructure which is expressed in the *ATT&CK for Enterprise* knowledge base [Str18]. As the adversary's targets and actions differ significantly between the Enterprise and ICS technology domains, MITRE adopted the separation of ATT&CK methodology and consequently introduced the different knowledge bases.

Evasion	Discovery	Lateral Movement	Collection
Change Operating Mode (COM)	Network Connection Enumeration (NCE)	Default Credentials (DC)	Automated Collection (AC)
Exploitation for Evasion (EfE)	Network Sniffing (NS)	Exploitation of Remote Services (EoRS)	Data from Information Repositories (DfIR)
Indicator Removal on Host (IRoH)	Remote System Discovery (RSD)	Lateral Tool Transfer (LTT)	Detect Operating Mode (DOM)
Masquerading (M)	Remote System Information Discovery (RSID)	Program Download (PD)	I/O Image (II)
Rootkit (R)	Wireless Sniffing (WS)	Remote Services (RS)	Man in the Middle (MitM)
Spoof Reporting Message (SRM)		Valid Accounts (VA)	Monitor Process State (MPS)
			Point & Tag Identification (PTI)
			Program Upload (PU)
			Screen Capture (SC)
			Wireless Sniffing (WS)

Figure 2.4: The MITRE ATT&CK for ICS Matrix. (Part 2)

The realization of all the MITRE ATT&CK *techniques* in a testbed is not possible given the domain, protocol characteristics and resource availability constraints of the testbed. For the reported research and the testbed employed, *procedures* for `Modify Parameter (MPrm)` and `Denial of Service (DoS)` are implemented (section 2.5.3) and utilized for the development and evaluation of self-learning anomaly detection systems.

Command and Control	Inhibit Response Function	Impair Process Control	Impact
Commonly Used Port (CUP)	Activate Firmware Update Mode (AFUM)	Brute Force I/O (BFI)	Damage to Property (DtP)
Connection Proxy (CP)	Alarm Suppression (AS)	Modify Parameter (MPrm)	Denial of Control (DoC)
Standard Application Layer Protocol (SALP)	Block Command Message (BCM)	Module Firmware (MF)	Denial of View (DoV)
	Block Reporting Message (BRM)	Spoof Reporting Message (SRM)	Loss of Availability (LoA)
	Block Serial COM (BSC)	Unauthorized Command Message (UCM)	Loss of Control (LoC)
	Data Destruction (DD)		Loss of Productivity and Revenue (LoPR)
	Denial of Service (DoS)		Loss of Protection (LoP)
	Device Restart/Shutdown (DRS)		Loss of Safety (LoS)
	Manipulate I/O Image (MII)		Loss of View (LoV)
	Modify Alarm Settings (MAS)		Manipulation of Control (MoC)
	Rootkit (R)		Manipulation of View (MoV)
	Service Stop (SS)		Theft of Operational Information (TOI)
	System Firmware (SF)		

Figure 2.5: The MITRE ATT&CK for ICS Matrix. (Part 3)

Table 2.1: Summary of ICS APTs mapped to the employed ATT&CK for ICS techniques.

Tactic	Techniques Employed by APTs		
	Stuxnet	Industroyer/ CRASHOVERRIDE	Triton/ TRISIS/ HatMan
Initial Access	*EoRS, RTRM, RS*	–	–
Execution	*EtA, H, MCT, NA,UE,CLI*	*CLI*	*H, MCT, NA, S, EtA*
Persistence	*MPgm,PFI*	–	–
Privilege Escalation	–	–	*EfPE*
Evasion	*M, R*	*M*	*COM, EfE, IRoH,M*
Discovery	*NS, RSID*	*NCE, RSD, RSID*	*RSD*
Lateral Movement	*DC, PD, LTT*	–	*PD*
Collection	*II, MPS*	*AC, MPS*	*DOM, PU*
Command and Control	*CUP, SALP*	*CP*	*CUP, SALP*
Inhibit Response Function	*MII*	*AFU, BCM, BRM, BSC, DD, DoS, DRS, SS*	*SF*
Impair Process Control	*MPrm, UCM*	*BFI, UCM*	–
Impact	*DtP, MoC, MoV*	*DoC, DoV, LoC, LoP, LoV, MoC, MoV*	*LoS*

2.3 Intrusion Detection System (IDS) Taxonomy for Industrial Networks

Hu et al. [Hu18] defines IDS for industrial networks or ICS (ICS IDS) as "*devices or software applications or their combinations monitoring the behaviors of ICS for detecting malicious activities or policy violations by collecting and analyzing all available data (e.g. protocol specifications, system logs, host data, network traffics, sensor measurements, together with the domain-specific knowledge of industrial control)*". They contend that the traditional IDS taxonomy is designed for information systems and do not pay attention to peculiarity of ICS domain on its close relationship with the physical world. Hence, a new ICS IDS taxonomy is proposed based on the detection techniques and ICS characteristics as three categories: *protocol analysis-based*, checks protocol specification violations in an industrial control network; *traffic mining-based*, analyses non-linear and complex relationships between the network traffic and the normal/abnormal system behaviour, and *control process analysis-based*, detects semantic attacks tampering with industrial process data or operating rules of specific control systems.

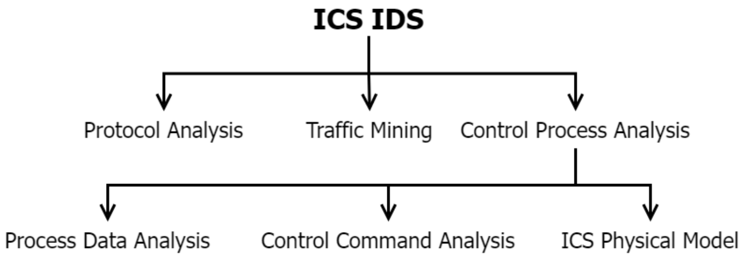

Figure 2.6: Hu et al.'s IDS Taxonomy for ICS/Industrial Networks [Hu18].

Protocol analysis-based IDS. The industrial protocols were designed with the goal of reliability and efficiency of ICS without any consideration to the cybersecurity aspects. The lack of encryption and authentication mechanisms make them vulnerable to cyber threats, and led to the emergence of protocol analysis-based IDS. A protocol specification defines the message format and the networking operations allowed by the protocol. A protocol analysis-based IDS uses the specifications of a protocol to detect its violation in message formats and/or their exchanges to identify the abnormal behaviours in industrial communication network of ICS. Cheung et al.'s *Modbus/TCP* based IDS [Che07], Snort-based IDS for *Modbus* by Morris et al. [Mor12], Lin et al.'s Bro-based IDS for *DNP3* protocol [Lin13a] and Hong et al.'s IDS for IEC 61850 standard protocols [Hon14] are some of the reported work in the literature for this category of ICS IDS.

Traffic mining-based IDS. The traffic data is collected from different regions in ICS and data mining or data analysis methods are applied to detect anomalous behaviours in the network communication of ICS. Usage of Principal Component Analysis (PCA) by Hou et al. [Hou12], Neural Networks (NN) by Ashfaq et al. [Ash17], One-Class Support Vector Machines (OC-SVM) by Maglaras et al. [Mag14], Caselli et al.'s Discrete-Time Markov Chain (DTMC) model [Cas15] and Ant Colony based IDS by Aghdam et al. [Agh16] are some of the reported work for traffic mining-based IDS.

Control process analysis-based IDS. The semantic information and peculiarity of ICS hasn't been considered by the traditional IDS system. A control process analysis-based IDS includes *process data analysis-based*, *control command analysis-based* and *ICS physical model-based* intrusion detection techniques. Hadžiosmanović et al.'s IDS based on semantic analysis of process variables [Had14], Kiss et al.'s Gaussian Mixture Model (GMM) dependent IDS [Kis15] and Gao et al.'s NN-based behavioural model [Gao10] are some of the published process data analysis-based IDS. Carcano et al.'s IDS for *Modbus* control command [Car09] and Lin et al.'s semantic analysis technique for control commands in distributed ICS [Lin13b] are examples of control command analysis-based IDS. The linear state-space model for ICS by Cárdenas et al. [Cár11], Myers et al.'s process mining based IDS and intrusion detection and mitigation mechanism for smart grids reported by Sridhar et al. [Sri14] are few examples of ICS physical model-based IDS.

2.4 Datasets for Industrial Cybersecurity

The quality and characteristics of datasets employed in the development of ICS cyber threat detection methods play an important role in driving the Industrial Cybersecurity research. Sommer and Paxson in their seminal paper on application of Machine Learning to network security [Som10] highlighted the importance of dataset quality, "*an anomaly detection system that strives to find novel attacks using only simulated activity will often lack any plausible degree of realism or relevance*". In the light of this observation, different publicly available ICS datasets used by the reported Industrial Cybersecurity research in the literature are analysed and compared for different characteristics: *ICS testbed class*, *ICS domain*, *ICS protocol coverage*, *data type* and *ICS threats coverage*.

The datasets are classified according to the taxonomy introduced by Holm et al. [Hol15]. This taxonomy differentiates between `Physical`, testbeds only using real hardware and software; `Software`, testbeds based on software simulation and emulation; `Semi-Physical`, testbeds composed of real and simulated components, and `Virtualized`, testbeds based on the virtualization technology. The ICS domain related to each dataset is classified based on the categories defined by Teumim [Teu10]. The format and type of the data provided by the datasets defines the extent of features used for the analysis algorithms. Sommer and Paxson [Som10] highlighted this as a common pitfall in developing network intrusion detection methods, "*the temptation to base the feature set on the dataset that happens to be at hand for evaluation*". The network data are provided either in already dissected (`Dissected`) or raw form (`PCAP`). In some cases, the datasets focus exclusively on process data (`Process Values`). The ICS network protocols used for generating the dataset are also extracted for the analysis. The attack behaviour from the datasets are analysed and categorized according to *MITRE ATT&CK for ICS* knowledge base for qualitative evaluation of ICS threats coverage.

Table 2.2: Comparison of public ICS datasets.

Dataset	Testbed Class	Domain	Data Provided	Protocol Coverage	Technique Coverage
SWaT [Mat16]	*Physical*	*Water Supply*	*Process Values*	*EtherNet/IP*	*MPrm*
GAS Pipeline [Mor14]	*Physical*	*Gas Oil*	*Dissected, Process Values*	*Modbus*	*MPrm, UCM, SRM, DRS,COM, DoS, RSD,NCE*
WADI [Ahm17]	*Physical*	*Water Supply*	*Process Values*	*EtherNet/IP*	*MPrm*
TEP Simulation [Rie17]	*Semi-physical*	*Chemical Industry*	*Process Values*	*Not Available*	*Not Available*
CSET Modbus [Lem16]	*Software*	*Electric Power*	*PCAP*	*Modbus*	*ERS, PTI, UCM, CUP*

TECHNIQUES: CUP - Commonly Used Port; DoS - Denial of Service; DRS - Device Restart/Shutdown; ERS - External Remote Services; MPrm - Modify Parameter; NCE - Network Connection Enumeration; PTI - Point & Tag Identification; RSD - Remote System Discovery; SRM - Spoof Reporting Message; UCM - Unauthorized Command Message; COM - Change Operating Mode

The table 2.2 summarizes the analysis of publicly available datasets: iTrust SUTD's Secure Water Treatment (`SWaT`) [Mat16] and Water Distribution (`WADI`) [Ahm17]; Morris et al.'s Gas Pipeline dataset (`Gas Pipeline`) [Mor14]; Tenessee Eastman Process Simulation dataset (`TEP Simulation`) from Rieth et al. [Rie17] and Modbus Dataset from CSET 2016 (`CSET Modbus`) from Lemay et al. [Lem16].

It can be observed from the table that there is lack of *PROFINET* based dataset availability which made any of these publicly available datasets not feasible for the reported *PROFINET*-based Industrial Cybersecurity analysis. Hence, a *PROFINET*-based miniaturized *physical* industrial process realized at the IT-Security Laboratory is employed to generate the dataset for the analysis.

2.5 System under Consideration

The IT-Security Laboratory, Fraunhofer IOSB offers a platform for industrial cybersecurity research and educational trainings. Different miniaturized manufacturing/production processes are realized as real-world demonstrators with extended virtualized infrastructure. The industrial network vulnerabilities in communication protocols or industrial components are exploited to develop threat detection methods and countermeasures for secure and safe process control communication. In particular, the datasets collected from demonstrators are real-world, encapsulating process and network communication with subtleties that can't be emulated completely by emulators/simulators. Different threat models attacking ICS are simulated through attack scripts and executed on the demonstrators to capture their effects in network traffic for downstream analyses.

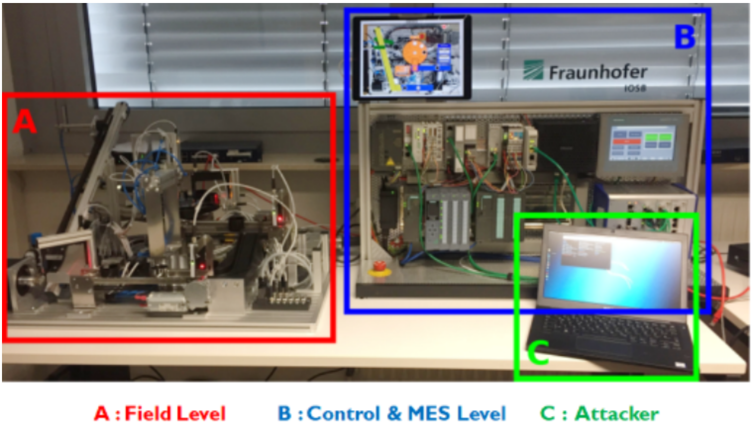

Figure 2.7: The Festo Demonstrator at IT-Security Lab, Fraunhofer IOSB.

A miniaturized manufacturing process is realized on a demonstrator labelled as the *Festo Demonstrator*, shown in fig. 2.7. In the figure, the red-coloured outlined box A shows the field level with real-world automation components where the scenario process is realized. The process is controlled by real-world ICS components such as PLC and HMI, highlighted in blue-coloured outlined box B. MES with overview on execution of controlled process can be visualized on a *Tablet* in the top-left corner of B. The attack scripts simulating an attacker/adversary exploiting industrial network vulnerabilities are executed on the laptop in green-coloured outlined box C.

2.5.1 Industrial Process Scenario

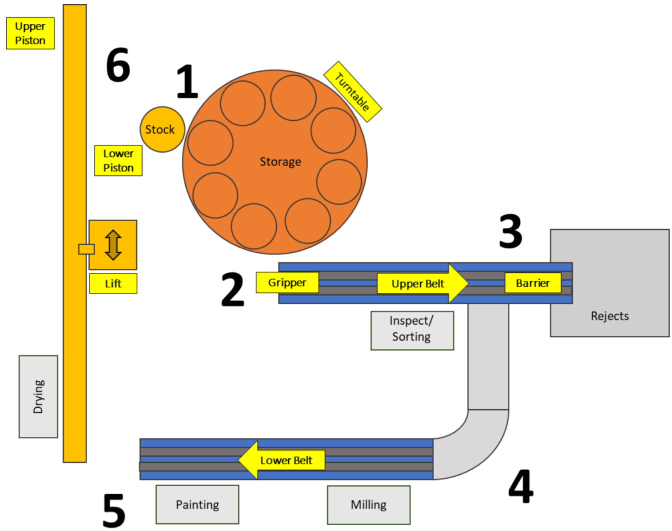

Figure 2.8: The Painting Process of the Festo Demonstrator.

The process scenario realized in the *Festo Demonstrator* is a simplified painting process, shown in fig. 2.8. The *workpieces* are stored in a vertical cylinder-shaped (open at both ends) *Stock*. The process begins when pneumatic *Piston* at the bottom of *Stock* pushes a *workpiece* onto the *Storage*, a cylindrical turntable with 8 *workpiece* positions (**1**). In the next process step, a pneumatically operated *Gripper* picks up the *workpiece* from the *Storage* and places it onto the upper conveyor *Belt* (**2**). The upper *Belt* moves the *workpiece* to *Sorting/Inspection* system. It is inspected for defects and quality control checks. If it is defective, the *Barrier* opens up and it is moved to *Rejects* bin (**3**), otherwise the *Barrier* diverts it to slide onto lower conveyor *Belt* for further processing (**4**). On the lower *Belt*, the *workpiece* follows through *Milling* and *Painting*, and is transported to *Lift*'s platform at the end of the *Belt* (**5**) for drying. At the end of the *Lift*, a final quality control of *workpiece* takes place. The painting process of *workpiece* ends and the pneumatic *Piston* at the top of *Stock* pushes the *workpiece* back into the *Storage* (**6**). The process begins anew.

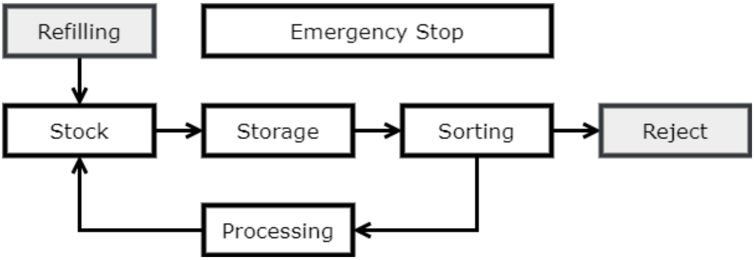

Figure 2.9: The process flow in the Festo Demonstrator.

In summary, the industrial process is a closed-loop of *workpieces* (*Stock*, *Storage*, *Sorting*, *Processing*) that can be entered (*Refilling*) and exited (*Rejects*) at one point each, shown in fig. 2.9. The transportation of *workpieces* through various stations is indicated by arrows. The process is parallelized so that at same time there can be up to 6 *workpieces* in *Stock*, 8 *workpieces* in *Storage*, 2 *workpieces* in *Sorting*, 2 *workpieces* in *Processing* and 10 *workpieces* in *Rejects*. The *Emergency Stop* when activated instantaneously disconnects power supply to moving actuators.

2.5.2 Industrial Network Communication

The industrial process is controlled through network communication between PLCs, I/O devices, actuators and process-associated sensors, and PLCs with MES and HMI. The process control is divided into 3 production cells, each with its own control system, as shown in fig. 2.10. The direction of arrow indicates *Master-to-Slave* control in the process.

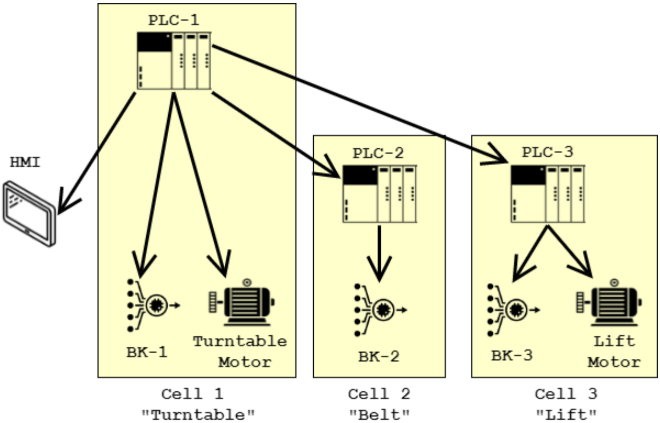

Figure 2.10: Different production cells of the Festo Demonstrator.

Cell 1 "Turntable". The *PLC-1* controls the *Turntable-Motor* of *Storage*, and bus coupler *BK-1* that relays digital/analog IO signals to the pneumatically operated *Gripper*. The *PLC-1*, *Turntable-Motor*, *BK-1* and associated sensors are part of Cell 1.

Cell 2 "Belt". The *PLC-2* controls the bus coupler *BK-2* that relays digital/analog IO signals to upper conveyor *Belt* to move. The *PLC-2*, *BK-2* and corresponding sensors constitute Cell 2.

Cell 3 "Lift". The *PLC-3* controls the *Lift-Motor* of *Lift*, and bus coupler *BK-3* that relays digital/analog IO signals to pneumatic *Piston*s at lower and upper end of the *Stock*. The *PLC-3*, *Lift-Motor*, *BK-3* and necessary sensors are part of Cell 3.

The *PLC-1* controls the *HMI*, *PLC-2* and *PLC-3*. It is the global *Master* in the industrial process network.

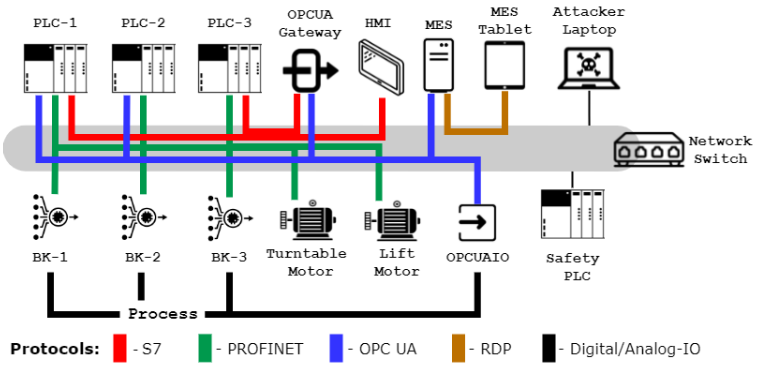

Figure 2.11: The networking infrastructure of the Festo Demonstrator.

In fig. 2.11, the network infrastructure (physical topology) of the *Festo Demonstrator* is shown. All the components are connected in STAR topology with the *Network Switch* at the center. The PLCs communicate to bus couplers and motors through *PROFINET* protocol, whereas PLCs to *HMI* communication is through *S7Comm* protocol. The process information is relayed to *MES* through *OPC UA* protocol from *OPC UA*-compatible PLCs. In case of non *OPC UA*-compatible *PLC-3*, an *OPC UA* gateway collects information from *PLC-3* through *S7Comm* and relays it to *MES*. *RDP* protocol is used to connect *MES* server to a *Tablet* to visualize the process execution.

2.5.3 Industrial Network Attack Scenarios

An *adversary* is assumed to have gained access to the *Festo Demonstrator*'s network infrastructure. In addition, it has the knowledge on the employed industrial components and process parameters being exchanged. Two attack scenarios exist where the *adversary* targets the industrial process through (i) changing the industrial component's configuration to make it unavailable for communication (*'Rename Attack'*), and (ii) setting the PLC into "FORCE" operational mode to enforce change in the process parameters for an industrial component (*'Force Attack'*).

Rename Attack. Within *PROFINET* networks, the components are addressed through their *logical names* for process data exchange (more details in section 3.3) via the unencrypted *PROFINET* protocol. An *adversary* exploits the *PROFINET* protocol design flaw and changes the *logical name* of *Turntable-Motor* to *"ufo"* via the *PN-DCP* protocol as shown in fig. 2.12. As a result, the other industrial components are not able to identify the component with the name *"Turntable-Motor"* and the process stops. Further details on *PN-DCP* and its role within *PROFINET* communication will be presented in section 3.3.2.

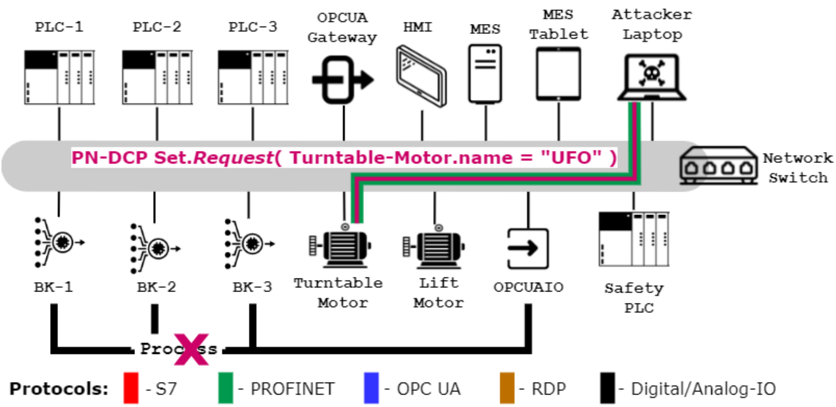

Figure 2.12: The Rename Attack on the Festo Demonstrator.

Force Attack. In certain events, an industrial component of the process is faulty and doesn't communicate the process parameters required for the industrial process to move forward. The process parameters for the faulty component can be "forced" to be sent in the process by the *Engineering Tool*, such as the *TIA-Portal* for *Siemens PLC*s, for the process continuation. An *adversary* with the knowledge of the *Festo Demonstrator*'s process, *enforces PLC-2* to change the "`direction`" process parameter of *BK-2* and reverses the lower conveyor *Belt*'s direction, as shown in fig. 2.13.

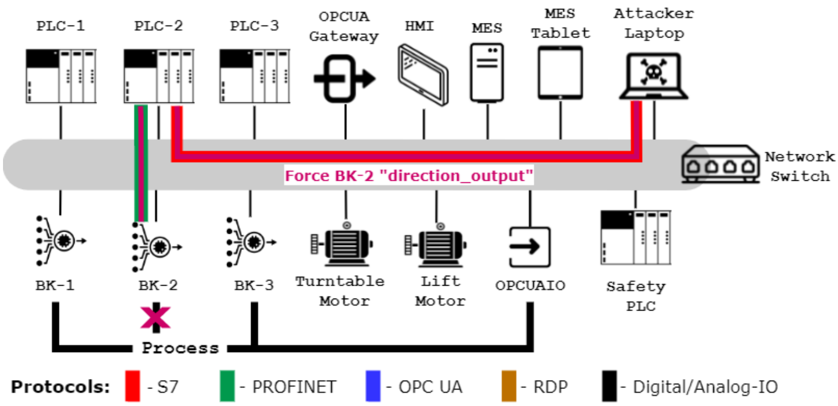

Figure 2.13: The Force Attack on the Festo Demonstrator.

2.5.4 Network Traffic Capture

For development and evaluation of the analytical methodologies proposed in the presented work, the network traffic from the *Festo Demonstrator* is captured through its *Network Switch*. In particular, one of the network interfaces on the *Switch* is configured to *mirror* the traffic passing through it and is termed as the *"Mirror Port"*. The network traffic is captured for two use cases:

1 **Baseline Behaviour.** Normal operation of the process is captured to *"learn"* the normal behaviour characteristics by the analytical methodologies. No attacks are executed during the capture, and *Baseline Dataset* row of table 2.3 summarizes the captured data.

2 **Anomaly Detection.** In order to test the performance of analytical models in distinguishing *normal* traffic from *anomalies*, the attack scripts are executed during the normal operations. The complete traffic inclusive of attack scripts execution and normal operations is captured for anomaly detection performance evaluation. The information on the captured data is summarized in *Evaluation Dataset* row of table 2.3.

Table 2.3: Summary of datasets captured from the Festo Demonstrator's communication traffic.

	Network Packets Count	**Process Duration**
Baseline Dataset	*60,523,524*	*2 hours 06 minutes*
Evaluation Dataset	*40,518,174*	*1 hour 24 minutes*

In both the use cases, the industrial process begins with *'referencing'* phase followed by *'actual process'* phase. In the *'referencing'* phase, the industrial components are brought to the initial state of the industrial process. In the following *'actual process'* phase, the process operations are executed. The *'referencing'* is crucial and unavoidable in the *Festo Demonstrator*. Therefore, the captured network packets are distinguished too for the two phases.

In addition, prior to *'referencing'* phase, the *Festo Demonstrator* undergoes through System Startup, where all the device configurations are transmitted across via different protocols.

The dataset generated from the *Festo Demonstrator*, normal and anomalous, is utilized for the development and evaluation of the solutions for self-learning of industrial operation behaviour in chapter 3, and industrial process behaviour in chapter 4 and chapter 5.

3 Self-learn *PROFINET*-based Industrial Communication Networks

Industrial control networks or industrial networks comprises of communication infrastructure between ICS components and network protocols managing the information flows across the automation system. Industrial network operations have strict requirements for real-time data transmission and deterministic communication. It entails further requirement of low jitter from the industrial network protocols to transmit data periodically and maintain strict order for the automated processes [Mes18]. Alarms and diagnostic information must be transmitted aperiodically at occurrence and request, respectively.

Proprietary fieldbus protocols were developed to satisfy these requirements such as *PROFIBUS*, *Modbus*, etc. *PROFINET* is the result of adapting *PROFIBUS* to real-time technology and standardized in IEC 61158 & IEC 61784. *PROFINET Input/Output (PNIO)* is the variant developed for distributed Input/Output (I/O) with provider-consumer distribution model for data exchange. It has RT and IRT channels for cyclic process data exchange with average cycle times of $50 - 100\,ms$ and $< 1\,ms$, respectively. RT channel is used for acyclic alarm transmission and *UDP/IP* channel for acyclic diagnostic data exchange. In addition, *PROFINET* defines different classes of components characterized by their functionality and participation at different stages of industrial communication - IO Controller, IO Supervisor and IO Device. The transmission of data from IO Controller/Supervisor to IO Device is designated as *output data* whereas IO Device to IO Controller/Supervisor is *input data* [V P18].

In this chapter, the challenges encountered in self-learning the industrial operation behaviour of a *PROFINET*-based system from its monitored traffic and the proposed solutions are outlined. The section 3.1 formulates the research problems followed with the foundational information on *PROFINET* in section 3.2. In section 3.3, the different networking operations executed through different network protocols in *PROFINET* systems and their corresponding data frame structures are provided. The proposed framework to enumerate and track the *PROFINET* operations from the analysis of traffic data is summarized in section 3.4. The section 3.5 presents the framework's usage for anomaly detection along with the chapter's review and conclusion.

3.1 Research Problem

Insights into the industrial system's operation are required for efficient monitoring and timely incident, *cyber* and *physical*, response. Interpretation of industrial system operations from its communication network characteristics contributes to being vigilant of cyber threats aimed at industrial process disruption. The networking operations such as neighbourhood detection, connection establishment that an industrial communication network undergoes contains all the required information for network monitoring. The industrial network's topology, communication relations, assets and protocol data being exchanged during the industrial process operation are the network information to be extracted from the network traffic passively. The challenge being addressed in this chapter is the detection of the different industrial operations from start-up to process data exchange from the network traffic in a systematic way for a self-learning approach.

In addition, the industrial operations executed in an order create the foundation for process data exchanges realizing the underlying intended process. Enumerating these operations through the analysis of multiple protocol communications observed in the traffic and tracking their executions helps to define the industrial system's operation behaviour. The addition of a device to the network and a device's attempt to establish communication for process data exchange triggers execution of network operations. Monitoring the valid operations of devices, communication links or the industrial system would detect the adversarial actions in the context of employed Industrial Ethernet technology's network protocol communication specifications. In this chapter, the challenge of representing and enumerating a *PROFINET* system's operations from monitoring the multiple protocol communication in the network traffic for self-learning an industrial system's operation behaviour is addressed. The enumerated industrial operations are furthermore tracked to detect deviations for anomaly detection in industrial system.

Interpreting *PROFINET*-based system operations from network traffic in a systematic way to make the network transparent for downstream analysis, such as modelling underlying ordered periodic and deterministic behaviour of industrial automation systems, is the research problem being addressed in this chapter. It is further formulated as following research questions:

1 How to extract *PROFINET*-based system operations from network traffic? [*Addressed in section 3.3*]

2 How to enumerate and track *PROFINET*-based system operations from network traffic? [*Addressed in section 3.4*]

3.2 Foundations

TCP (UDP)/IP communication model doesn't suffice to the industrial communication requirements of deterministic time responses and isochronous communication. Transmission duration of *UDP/IP* frames inclusive of processing time of a frame through standard communication stacks are found to be non-deterministic [Wil18]. A trimmed down version of communication stack is needed to be defined for real-time *PROFINET* communication.

For real-time data exchange, emphasis was put on omitting the management & extensive address information transmission, hence, eliminating Session & Presentation layer headers. The real-time frame doesn't have a Network layer header as it is not passing through any router. Generally the data to be transferred in real-time frame is way less than the standard Ethernet frame's 1518 bytes, there is no need of segmentation & reassembly of data omitting Transport layer header. Each *PROFINET* frame is identified by its IEEE assigned Ethertype 0x8892 contained in its VLAN tag for optimal processing by a device. Every *PROFINET* frame contains `Frame ID` which defines the associated communication channel of the connection. An *UDP/IP* protocol based on OSF DCE RPC is used to establish a connection between the controller and device before the real-time data exchange begins.

There are two real-time properties of *PROFINET* communication: non-synchronized real-time communication (RT) and synchronized real-time communication (IRT). Within *PROFINET*, process data and alarms are transmitted with RT communication with bus cycle times in the range of milliseconds. Isochronous data transfer with IRT communication are used in applications such as motion control requiring bus cycle times in range of microseconds.

PROFINET utilizes the provider-consumer model of communication for I/O data exchange between controllers and devices, as well as parametrization and diagnosis information exchange between supervisors and devices. *PROFINET* provides following communication services:

1 Cyclic real-time I/O data exchange between provider and consumer at parametrized increments.

2 Acyclic real-time transmission of acknowledged alarm in the case of system-defined & user-defined events related to controlled process.

3 Acyclic transmission of parameters & diagnostic information for configuring devices and reading out their status information.

PROFINET further defines 3 real-time classes for data exchange: RT_Class_1, RT_Class_3, RT_Class_UDP. RT_Class_1 is used for non-synchronized real-time communication within a subnet, where the minimum transmission cycle is generally 128 ms. The controller defines the time base of the cycle for the transmission of input data of devices. Each device adheres to its time interval for sending its input data. There is no synchronized start of bus cycle for all the devices. RT_Class_1 frames with VLAN tags are prioritized over *UDP/IP* frame by a standard industrial switch. RT_Class_3 involves optimized synchronous communication within a subnet where process data is sent in a specific order with maximum precision of 1 microseconds allowed jitter. This is referred as isochronous real-time functionality of *PROFINET*, where all the participating devices in IRT communication follow the strict order of data transmission. RT_Class_3 frames are transmitted without a VLAN tag through special isochronous capable hardware. RT_Class_UDP is used when real-time frames need to be sent non-synchronized between different subnets. RT_Class_UDP frames contain the destination network address and are transmitted without VLAN tags.

PROFINET defines following classes of components which participate in the industrial communication and drive the industrial process:

- IO Controller: It is the component with *master* functionality that executes the automation program, typically a PLC. It participates in parametrization, cyclic/acyclic data exchange and alarm processing with connected field devices. As the provider, it sends output data to devices and consumes input data from them.
- IO Supervisor: An IO Supervisor is used for the commissioning and diagnostic purposes. This is generally a programming device, personal computer or HMI.
- IO Device: It is a field device in the vicinity of process with *slave* functionality that sends process data and critical statuses (alarms & diagnostics) to connected IO Controller(s) via *PROFINET*. It consumes output data from controller(s) and acts as the provider to send input data to them.

An IO Device comprises of an Ethernet interface for communication and physical/virtual modules to handle the process data traffic. The device model of an IO Device consists of *slots*, *subslots*, *modules*, *submodules* and *channels*. The slot and subslot designates the insert slot of a module and submodules in an

IO field device, respectively. The modules provide the structuring and its submodules contain the implemented actual inputs and outputs channels. A module contains at least one submodule which always contains the process data with status information. The data within the submodule is addressed using an index. Cyclic IO data in submodule are accessed through slot/subslot combinations, whereas, acyclic read/write services utilize slot, subslot and index.

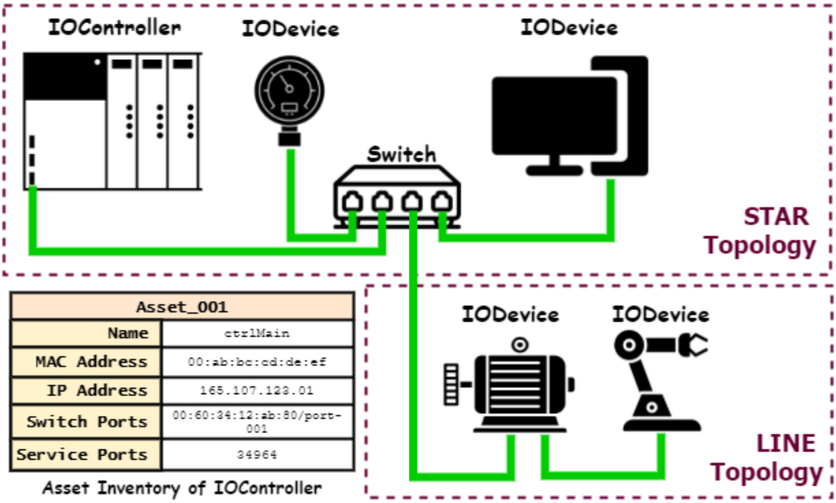

Figure 3.1: An example industrial system's network topology & asset inventory.

A *PROFINET* system contains at least one IO Controller and one or more IO Device. The IO Supervisor is used temporally for commissioning or troubleshooting purposes, however, it is capable of taking over the control of a device for the supervision of process. [1]

All the *PROFINET* components are connected via Switches in an automation system. Within the *PROFINET* communication network, a network asset's Ethernet interface is defined by its MAC address, IP address and name. When

[1] Hereafter, the references to device(s) and controller(s) mean IO Device(s) and IO Controller(s).

devices have additional switch ports, each port is identified by its MAC address. Port MAC addresses are used to avoid relearning of Switches' internal address table whenever their are different paths from same device. All these information constitute an Asset Inventory entry (example in fig. 3.1) of components which is maintained throughout its usage.

There are two variants of network topology observed based on bus structures: *star topology*, where multi-port industrial Switches are at the center with each port occupied by a *PROFINET* device, and *line topology*, where devices with integrated Switches are connected in a line. If one of the switch port fails in the line topology, the communication is interrupted, therefore, redundant paths have to be planned beforehand.

3.3 Network Operation Enumerations for *PROFINET*

Configuration and commissioning of *PROFINET*-based automation systems must follow certain mode of operations in a strict order. It begins with the System Engineering operation where an automation project is configured in an engineering tool. General System Description (GSD), an *XML* file provided by every device manufacturer, contains configuration information for parametrizing the devices for real systems. In addition, each device is assigned a logical name to address it within the *PROFINET* communication. Within the System Engineering mode, an IP address is assigned to each device for communication. Transmission intervals are defined for cyclic data exchange between controller and devices. After system engineering is completed, the configuration information is downloaded to the controller. As soon as the automation system is powered on (or reset), Neighbourhood Detection, Address Resolution and System Startup are the operations followed in the same order, as shown in fig. 3.2. With Address Resolution, the controller uses the system configuration information to assign the IP addresses to the devices identified through their pre-assigned logical names. System Startup operation mode is initiated by the controller to establish connection with devices and configure their I/O parameters. When the I/O parametrization ends successfully, the controller and devices step into Data Exchange mode to transmit process data, alarms and diagnostic information throughout the network.

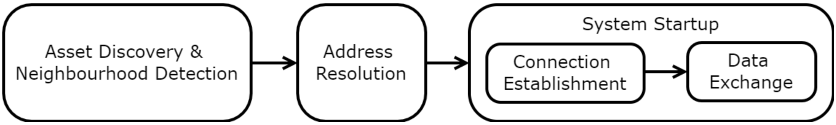

Figure 3.2: Networking operations of an industrial system.

Every mode of operation, from configuration to commissioning the *PROFINET*-based automation system, is accomplished through a complementary network operation involving specific network protocols. *Link Layer Discovery Protocol (LLDP)* is associated with Asset Discovery & Neighbourhood Detection networking operations to accomplish the Neighbourhood Detection *PROFINET* operation. The Address Resolution networking operation is performed by *PN-DCP* and *Address Resolution Protocol (ARP)* to execute Address Resolution *PROFINET* operations. The System Startup *PROFINET* mode involves *PROFINET Context Manager (PN-CM)* protocol reflecting Connection Establishment networking operation. *PNIO* protocol is associated with Data Exchange networking mode to achieve the Data Exchange *PROFINET* operations.

Enumeration of *PROFINET* operation modes is performed by passively monitoring/analysing the network traffic and identifying the associated network operation stage-by-stage. System Engineering mode is performed offline, hence, it can't be enumerated through analysing the network traffic.

In the following subsections, we will dive into each of the networking operations. For every network operation, the role it plays within *PROFINET*-based automation system communication and the network protocol utilized to achieve the goal will be outlined. The information extracted through dissecting protocol traffic and its usage in filling the asset inventory and network topology model are also presented.

3.3.1 Asset Discovery & Neighbourhood Detection

After the automation system is powered on, the field device's MAC interface and its Physical Device Management (PDev) gets activated to start transmitting the parameters. PDev contains hardware-level information such as interface name, switch port data, interface and Port MAC addresses and retentively stores IP address and logical name assigned to the device. Port information is used by devices to determine their neighbours on port-by-port basis. Neighbourhood Detection is accomplished through *LLDP* services as part of *PROFINET*'s overall concept "Device Replacement Without Engineering Tool". *LLDP*-capable devices communicate with their connected neighbours to cyclically exchange addressing information and consequently determine their physical location. A controller can use *PN-DCP* or *Simple Network Management Protocol (SNMP)* to query the *LLDP* information from devices and reproduce the system topology. When a field device is replaced with new MAC address, the system topology information stored by the controller is used to commission the new device automatically using it's assigned logical name.

An example of *LLDP* frame structure is shown in fig. 3.3. *LLDP* frames are identified through their IEEE-assigned `Ethertype` 0x88cc with fixed `multicast MAC address` 01-80-C2-00-0E. `Source MAC address` is Port MAC address of sending device's switch port. `LLDP Data Unit (LLDPDU)` contains chain of information blocks called Type Length Value (TLV) fields: `Chasis ID`, `Port ID`, `Organization Identifier (OID)`, `Time To Live (TTL)` and `End of LLDPDU`. `Chasis ID` corresponds to device's name. `Port ID` value has the form PORT-001, if one switch port is present, else PORT-001-RSTUV for multiport switches, where RSTUV represents the slot number. OID has value 24686 if *PROFINET* device else 0 when no value is provided by the manufacturer. TTL specifies the update time to ensure consistent data management and LLDP information validation.

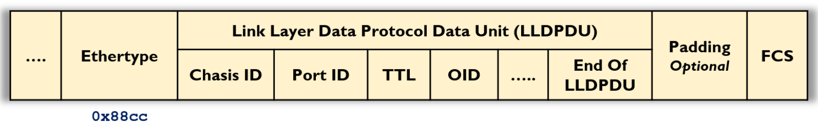

Figure 3.3: The structure of a *LLDP* frame.

LLDP frames are dissected to identify device names, number of switch ports and their MAC addresses for the Automated Asset Inventory, developed for the analysis in the reported work. Fig. 3.4 shows an example of attributes added for the device *Lift-Motor* of the *Festo Demonstrator* (refer section 2.5).

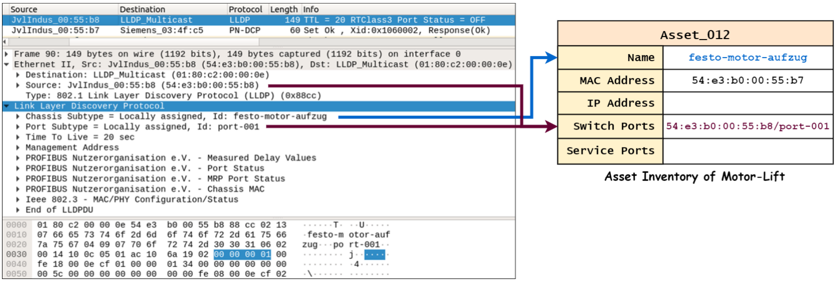

Figure 3.4: The demonstration of *Lift-Motor* device's Asset Inventory filled with information from *LLDP* frame (*Wireshark* snippet).

3.3.2 Address Resolution

Before the *PROFINET*-based automation system starts up and field devices start communicating, an IP address needs to be assigned to all devices by the controller. An IO Device is identified through its '*NameOfStation*' information stored in its PDev and an IP address defined during System Engineering mode is assigned to it. Address Resolution networking operation for every device takes place step-by-step as follows:

1 Controller starts with name resolution and checks for device with configured name through '*DCP Identify*' service of *PN-DCP*.

2 Address Resolution begins with checking if the IP address already exists to avoid assigning same IP address twice through *ARP*.

3 At the end of networking operation, the IP address is assigned to configured device through '*DCP Set*' service of *PN-DCP*.

The schematic communication order for Address Resolution between *PLC-3* and *Lift-Motor* is shown in fig. 3.5.

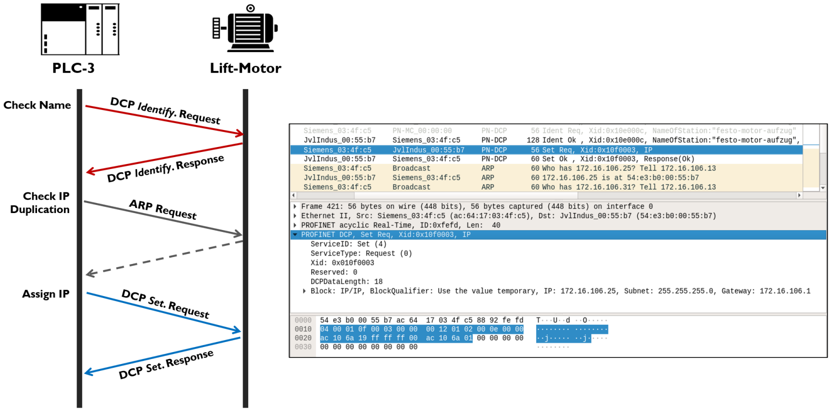

Figure 3.5: The Address Resolution handshake between *PLC-3* and *Lift-Motor*.

Frame structures of *DCP Identify*, *ARP* and *DCP Set* are outlined below with a brief description of relevant fields.

DCP Identify is the real-time service of *PN-DCP* with `Ethertype` 0x8892 and fixed PI `multicast MAC address` 01-0E-CF-00-00-00 for *request* frames. *DCP Identify request* and *response* frames are identified through their service identifiers in `DCP Header` with values 5.0 and 5.1, respectively (fig. 3.6). `DCP Data` field contains name of the device which is being checked for availability. `DCP Response` field is contained only in *response* frame instead of `DCP Data` to confirm the availability of device with searched name and parameters.

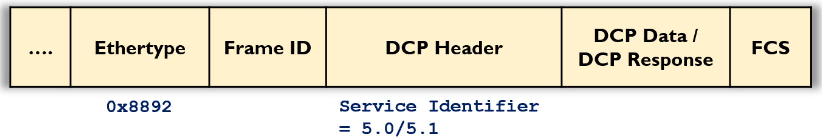

Figure 3.6: The structure of a *DCP Identify* frame.

ARP is the standard IT service for IP resolution with `Ethertype` 0x0806. The controller sends an *ARP request* with broadcast `destination MAC address` FF-FF-FF-FF-FF-FF to all devices checking for IP address availability. At the startup, since none of the devices are assigned any IP address, there wont be any *ARP response* frames. *ARP request* and *response* frames are identified by their operation code in `OPER` field with values 1 and 2, respectively. `SHA`, `SPA`, `THA` and `TPA` fields correspond to hardware (MAC) address and protocol (IP) address of sender and target, respectively (fig. 3.7). In *request* frame, `THA` is broadcast address, `TPA` is the IP address query, `SHA` and `SPA` are MAC and IP address of the controller.

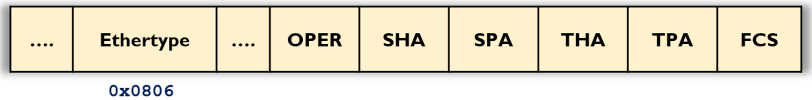

Figure 3.7: The structure of an *ARP* frame.

DCP Set is the real-time service of *PN-DCP* with `Ethertype` 0x8892. *DCP Set request* and *response* frames are identified through their service identifiers in `DCP Header` with values 4.0 and 4.1, respectively (fig. 3.8). With *DCP Set request* frame, the controller writes IP parameters contained in `DCP Data` field to a device.

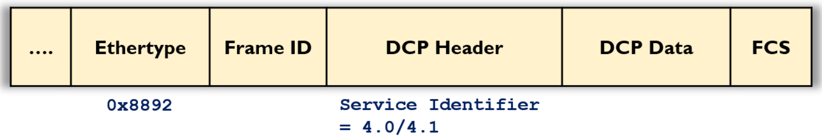

Figure 3.8: The structure of a *DCP Set* frame.

The information dissected from an *ARP request* packet is used to add the IP address to controller assets. Information dissected from *DCP Set* is used to add IP address information to configured devices, example shown in fig. 3.9.

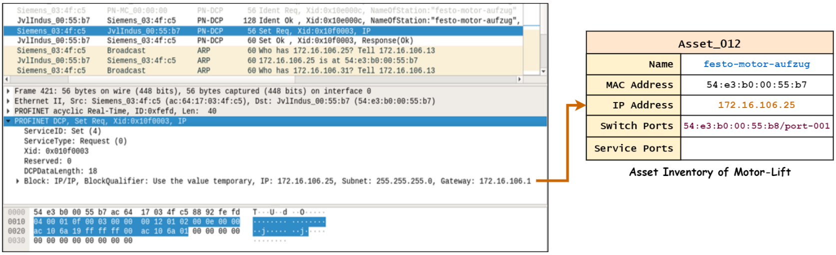

Figure 3.9: The demonstration of *Lift-Motor* device's Asset Inventory filled with information from *PN-DCP* frame (*Wireshark* snippet).

3.3.3 Connection Establishment

System Startup operation begins with establishment of communication relationships between the controller and devices via *PN-CM* communication exchanges. Through these established communication the controller transmits all the parameters for process data exchange to the internal module of devices. The process model and associated parameters for devices participating in the process are engineered & defined during System Engineering operation.

The '*connection*' between an IO Controller and an IO Device is established in an '*Application Relationship (AR)*' uniquely identified by an ARUUID. An ARUUID is generated by the engineering tool for specific system configuration & device. Within this *AR*, different '*Communication Relationship (CR)*' are established for different data exchanges. A device can be connected to multiple controllers through different *AR*s. An application can access data only through *CR*s established in an *AR*.

IOC-AR, *IOS-AR* and *Implicit AR* are different *AR*s defined by *PROFINET*. The *IOC-AR (Controller AR)* is between controller and device for cyclic data exchange of input & output data, read/write of acyclic data and alarms transmission. The *IOS-AR (Supervisor AR)* is between supervisor and device with properties same as *IOC-AR*. *Implicit AR* is between a controller/supervisor and device for the sole purpose of acyclic reading of data from device.

Different '*Communication Relationship (CR)*' are defined by *PROFINET* for different data requirements. *IO-Data CR (IOCR)* is the IO data communication relationship established for cyclic process data exchange in real-time. The controller transmits in *IOCR* various parameters for cyclic data exchange: transmission frequency, data length, specification of input/output data, sequence of transmitting data, etc. The controller can set up multiple *IOCR* with a device for different data objects with different transmission frequencies. The transmission frequency defines the fixed interval cycle for data exchange between controller and device in both directions (input & output). *Alarm CR* is established along with an *IOCR* for acyclic bidirectional transmission of process and diagnostic alarms via the real-time channel. *Record Data CR* is established for acyclic data exchange to read diagnostic, log and device data. It is explicitly used for writing the configuration and *AR* data.

A '*connection*' between a controller and a device is defined the least by 1 *AR*, 2 *IOCR* (input & output), 1 *Alarm CR* and 1 *Record Data CR*. *PROFINET* offers *PN-CM* protocol to handle the Connection Establishment network operation. The *PN-CM* network operation uses *UDP/IP* channel to transmit following frames in the strict order for establishing '*connection*' between controller and device:

- *Connect* frames establish *AR* and *CR*s channels.
- *Write* frames parametrize the device *submodule*s.
- *DControl* frames mark the end of parametrization from the controller.
- *CControl* frames mark the validation check of parameters, data structure build up and application readiness from the device.

The first successful exchange of I/O data after *CControl* frames mark the end of *PROFINET*'s System Startup operation mode. The schematic communication order for Communication Establishment between *PLC-3* and *Lift-Motor* is shown in fig. 3.10.

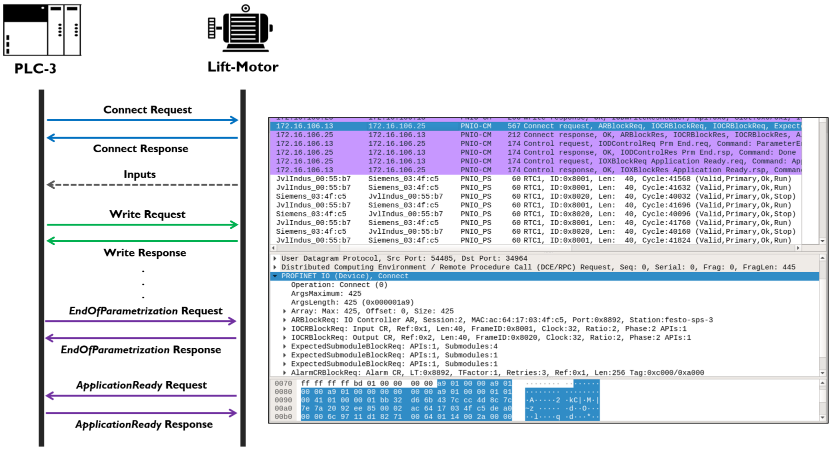

Figure 3.10: The Connection Establishment handshake between *PLC-3* and *Lift-Motor*.

Frame structures of *Connect*, *Write*, *DControl* and *CControl* are outlined below with a brief description of relevant fields. All *PN-CM* frames are transmitted with `Ethertype`=0x0800.

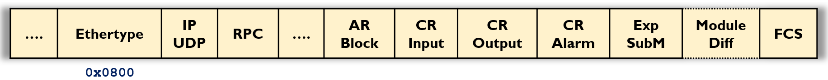

Figure 3.11: The structure of a *PN-CM Connect* frame.

A *Connect request* frame is sent from controller to device containing parametrization information for transmission frequency, length and order of process data, input & output data specification related to *submodules* and *subslots* of device. Any error in the parametrization information (such as absence of *submodule*) is notified by the device in a *Connect response* frame. `RPC` contains the data transmission format value either *Big Endian* or *Little Endian*. `AR Block` contains `ARUUID` and relevant information to establish *AR* between controller and device. `CR Input` and `CR Output` are the blocks with instructions for input and output data transmission including transmission frequency (refer fig. 3.11). `ExpSubm` is only present in *request* frames containing identification of device's *modules* and *submodules* to be parametrized for process data exchange. `ModuleDiff` is present only in *response* frames in the case of error in setting up device's *modules* and *submodules*.

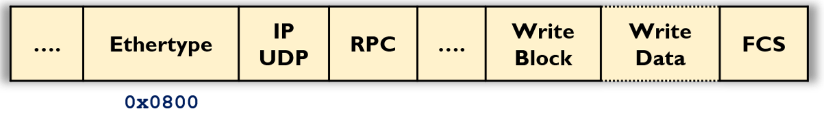

Figure 3.12: The structure of a *PN-CM Write* frame.

The *Write* frame has a `Write Block` field containing information for addressing the data objects of *submodules* and its length (refer fig. 3.12). The `Write Data` field is only present in a *Write request* frame from controller to device and contains the parametrization data for the addressed data object.

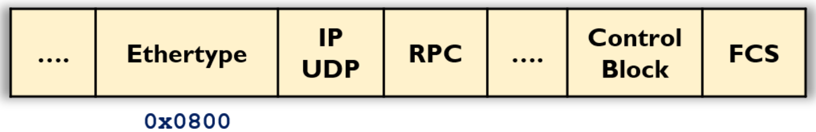

Figure 3.13: The structure of a *PN-CM DControl* frame.

DControl request frame is sent from controller to device notifying the end of parametrization and the device acknowledges it in the *response* frames (refer fig. 3.13).

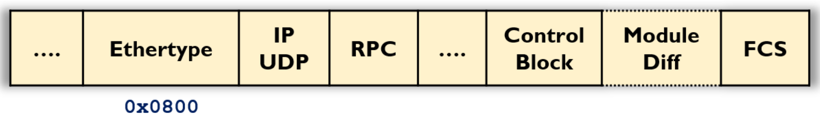

Figure 3.14: The structure of a *PN-CM CControl* frame.

A *CControl request* frame is sent from device to controller indicating its application readiness (refer fig. 3.14). It might contain `ModuleDiff` to notify error in writing data/parameters to identified *submodules*. If no `ModuleDiff` is found by the controller in the *CControl request* frame, it acknowledges device's application readiness in *response* frame. With this frame, the *AR* is established and all the frames required for startup are transmitted.

The GSD of a device contains modules and submodules information reflecting the position of process data within the payloads. Since in a self-learning analysis from network traffic the GSD isn't accessible, the aforementioned information is extracted through reading and interpreting *Connect* frames. *PN-CM* frames are dissected to extract the input and output data specifications for device's *submodule*s. It contains the data format type (endianess), order of data, length of data and the position of data within the payload bytes. An example of payload bytes of input data frames without network header information is shown in fig. 3.15.

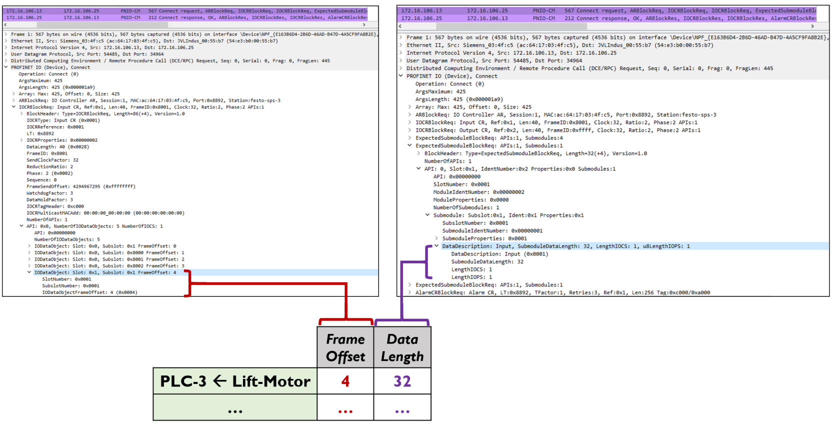

Figure 3.15: The demonstration of extracting Input Data parameters between *PLC-3* and *Lift-Motor* from *Connect* frame (*Wireshark* snippet).

These specifications are used in Data Exchange network operation's *PNIO* frames to extract process bytes. The '*connection*' between controller and device following a Connection Establishment network operation guides building the logical network topology of the system. These separated ISO layer connection state information between network assets is maintained throughout the automation system's runtime and deviations are reported, an example shown in fig. 3.16.

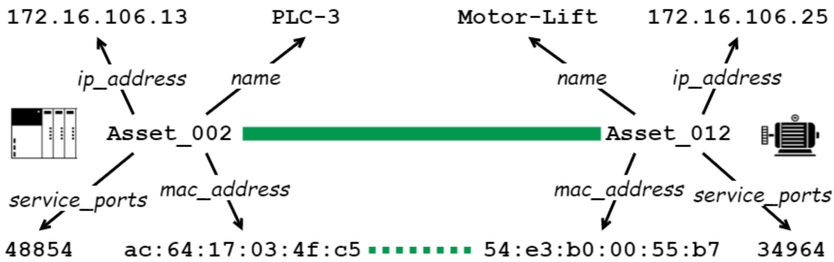

Figure 3.16: The logical network topology between *PLC-3* and *Lift-Motor*.

3.3.4 Data Exchange

Once the System Startup operation establishes *AR* and data specific *CR*s between devices and controller, the connection-oriented communication channel is set for exchange of cyclic process data, acyclic diagnostic data and alarms. Cyclic process data are transmitted in real-time in *IOCR* channel whereas *Record Data CR* transmitted acyclic data via *UDP/IP*. *PNIO* protocol defines the format and context for data exchange.

Cyclic *PNIO* frames are sent unacknowledged between controller and devices. The validity of transmitted data is exchanged through status information in field `IOPS (Input Output Provider Status)` and `IOCS (Input Output Consumer Status)` by the provider and consumer, respectively. After *CControl* frames are acknowledged by the controller, the first valid exchange of I/O data with `IOPS=GOOD` ends Connection Establishment operation. Data Exchange operation begins with cyclical exchange of process data at configured/parametrized fixed intervals.

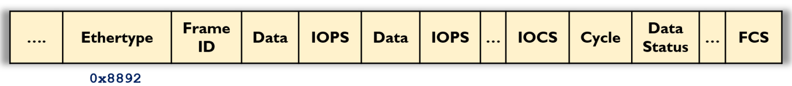

Figure 3.17: The structure of a *PNIO* frame.

PNIO cyclic data frame is transmitted in real-time with `Ethertype=0x8892` and its structure outlined in fig. 3.17. `Frame ID` distinguishes between cyclic and acyclic real-time frame. The process data bytes from `Data` field are extracted using information from *Connect* frame dissection, as shown in fig. 3.18.

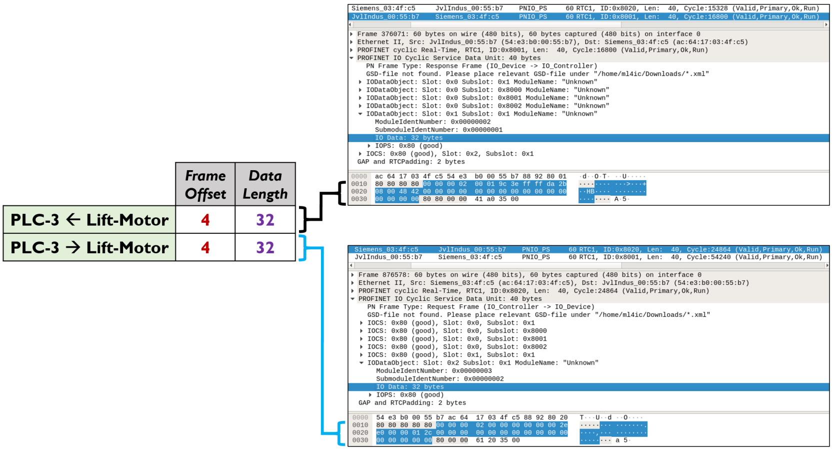

Figure 3.18: The demonstration of extracting process data between *PLC-3* and *Lift-Motor* from *PNIO* frames (*Wireshark* snippet), using parameters extracted from *Connect* frame.

3.4 *PROFINET* Operations Enumeration and Tracking

Through monitoring an industrial system's communication network, its characteristics are observed to build normal operations behaviour baseline. Deviations from the baseline behaviour could be triggered by *physical* or *cyber* threats. A systematic framework is needed to enumerate operations extracted from network traffic and track them to report deviations.

PROFINET-based automation systems follow strict order of operations. The operation mode and corresponding network operations with associated network protocols have been outlined in section 3.3. All of those network operation's dissected information are combined to systematically iterate over the *PROFINET* operation modes as and when there occurrences are observed through network analysis. *PROFINET* devices transit through *PROFINET* operations to establish *connection*s between them for cyclic and acyclic data exchange. These transitions also govern transitions in *PROFINET* connections, which constitutes the logical topology and industrial process behaviour of the *PROFINET* system.

Finite State Machines (FSM) [Kle56] are widely used for protocol specification (e.g. *TCP/IP* [Ste93]) [Hol93], where the valid *transitions* and *states* of message exchanges are defined [Reu02]. For the requirements of capturing the transitions between industrial operations triggered by the communication events, the FSMs are modelled to enumerate the *PROFINET* operations of the device, connection and system. *PROFINET* standard, the informative handbook on *PROFINET* [Pop14] and empirical information collected from analysing real-world *PROFINET*-based system communications are interpreted to model the operations in FSMs.

In the next subsections, each FSM is described with the overview of states and events triggering the transitions outlined in its state diagram. The transitions which are modelled based on empirical information are distinguished by dashed edges and details are presented. The FSM diagrams are accompanied with tabular details on source and target state, triggering event, protocol frame used for event detection and underlying *PROFINET* network operation of transition. Each row of the table represents a transition of FSM.

3.4.1 FSM *PROFINET* Device

States and transitions. FSM Device enters with *Active* state as soon as the system is powered on, shown in fig. 3.19. It transits either to *Neighbourhood Detection* or *Name Resolution* state depending on the event triggered. FSM follows through the transitions as and when the triggering event is detected in network traffic. The protocol frame triggering the events for *PROFINET* operation transitions of a device are summarized in table 3.1.

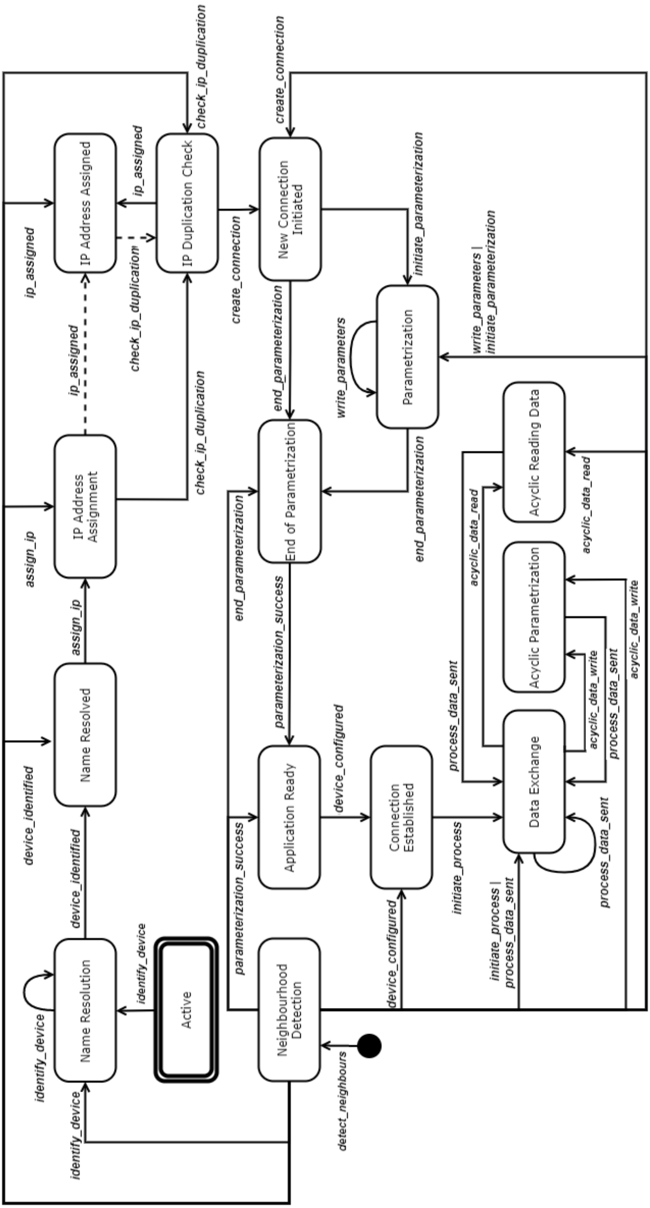

Figure 3.19: *PROFINET* Device State Machine.

LLDP frames are periodically transmitted by *PROFINET* device as per their TTL value for consistent *LLDP* information validation (refer section 3.3.1). Hence, *Neighbourhood Detection* state can be arrived from any other state whenever *detect_neighbours* event is triggered by *LLDP* frame. Consequently, all the states are reachable with corresponding triggering events from *Neighbourhood Detection* state.

Table 3.1: Summary of FSM Device transitions with *triggering event*, *event detection protocol* and *PROFINET operation.*

Source	**Target**	**Trigger Event**	**Protocol Frame**	***PROFINET* Operation**
*	Neighbourhood Detection	*detect_neighbours*	*LLDP*	Asset Discovery
Active Name Resolution Neighbourhood Detection	Name Resolution	*identify_device*	*DCP Identify request*	Address Resolution
Neighbourhood Detection Name Resolution	Name Resolved	*device_identified*	*DCP Identify response*	
Neighbourhood Detection Name Resolved	IP Address Assignment	*assign_ip*	*DCP Set request*	
Neighbourhood Detection IP Address Assignment IP Duplication Check	IP Address Assigned	*ip_assigned*	*DCP Set response*	
Neighbourhood Detection IP Address Assigned IP Address Assignment	IP Duplication Check	*check_ip_duplication*	*ARP (Gratuitous ARP)*	
IP Duplication Check Neighbourhood Detection Name Resolved	New Connection Initiated	*create_connection*	*Connect request*	Connection Establishment
Neighbourhood Detection New Connection Initiated	Parametrization	*initiate_parametrization*	*Write*	
Neighbourhood Detection Parametrization	Parametrization	*write_parameters*		
Neighbourhood Detection Parametrization New Connection Initiated	End Of Parametrization	*end_parmaterization*	*DControl request*	
Neighbourhood Detection End Of Parametrization	Application Ready	*parmaterization_success*	*CControl response*	
Neighbourhood Detection Application Ready	Connection Established	*device_configured*	*PNIO*	
Neighbourhood Detection Connection Established	Data Exchange	*initiate_process*	*PNIO*	Data Exchange
Neighbourhood Detection Data Exchange Acyclic Reading Data Acyclic Parametrization	Data Exchange	*process_data_sent*		
Neighbourhood Detection Data Exchange	Acyclic Reading Data	*acyclic_data_read*	*Read*	
Neighbourhood Detection Data Exchange	Acyclic Parametrization	*acyclic_data_write*	*Write*	

Through network traffic analysis of *PROFINET*-based systems with Siemens PLC, transitions - *IP Address Assignment* to *IP Address Assigned* and *IP Address Assigned* to *IP Duplication Check* - have been modelled. Deviating from transitions mentioned in the literature, the *PROFINET* devices checked for IP duplication with *Gratuitous ARP* after the IP address has been assigned to them. These transitions are verified on different *PROFINET*-based systems available at IT-Security Lab, Fraunhofer IOSB.

Relationship between states and *PROFINET* operations. The transitions between states reflecting progress of *PROFINET* operations of a device are outlined in table 3.1. *Neighbourhood Detection* state constitutes Asset Discovery & Neighbourhood Detection *PROFINET* operation. States *Name Resolution*, *Name Resolved*, *IP Address Assignment*, *IP Address Assigned* and *IP Duplication Check* constitute Address Resolution *PROFINET* operation. *PROFINET*'s Connection Establishment operation consists of states *New Connection Initiated*, *Parametrization*, *End Of Parametrization*, *Application Ready* and *Connection Established.* States *Data Exchange*, *Acyclic Parametrization* and *Acyclic Reading Data* reflect Data Exchange *PROFINET* operation.

3.4.2 FSM *PROFINET* Connection

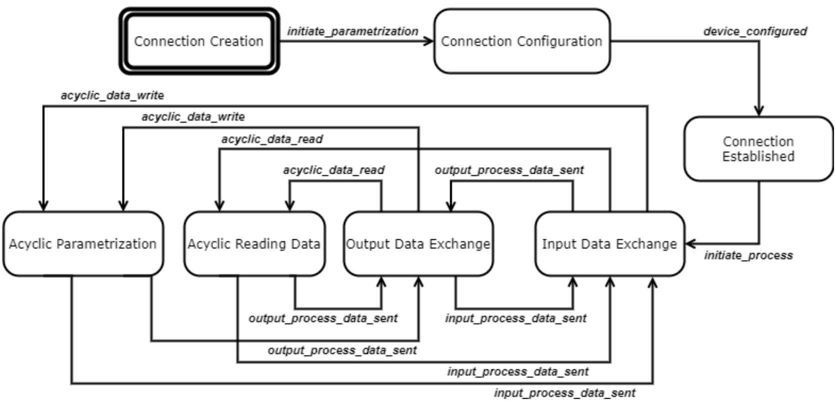

Figure 3.20: *PROFINET* Connection State Machine.

States and transitions. A *PROFINET* connection is established between *PROFINET* devices through *PROFINET*'s *PN-CM* protocol handshake (refer section 3.3.3). The cyclic and acyclic data exchange takes place through this connection. Hence, each connection is identified through MAC addresses of participating *PROFINET* devices. FSM Connection enters with *Connection Creation* state as soon as *Connect request* frame is sent by the controller, shown in fig. 3.20. FSM follows through the transitions as and when the triggering event is detected in network traffic. The protocol frame triggering the events for *PROFINET* operation transitions of a connection are summarized in table 3.2. In particular, events *output_process_data_sent* and *input_process_data_sent* are triggered by transmission of *PNIO* frames from controller to device and vice versa, respectively (refer section 3.3.4).

Table 3.2: Summary of FSM Connection transitions with *triggering event*, *event detection protocol* and *PROFINET operation*.

Source	Target	Trigger Event	Protocol Frame	*PROFINET* Operation
Connection Creation	Connection Configuration	*initiate_parametrization*	*Write*	Connection Establishment
Connection Configuration	Connection Established	*device_configured*	*PNIO*	
Connection Established	Input Data Exchange	*initiate_process*	*PNIO*	Data Exchange
Input Data Exchange	Output Data Exchange	*output_process_data_sent*		
Acyclic Parametrization				
Acyclic Reading Data				
Output Data Exchange	Input Data Exchange	*input_process_data_sent*		
Acyclic Parametrization				
Acyclic Reading Data				
Input Data Exchange	Acyclic Parametrization	*acyclic_data_write*	*Write*	
Output Data Exchange				
Input Data Exchange	Acyclic Reading Data	*acyclic_data_read*	*Read*	
Output Data Exchange				

Relationship between states and *PROFINET* operations. The transitions between states reflecting progress of *PROFINET* operations of a connection are outlined in table 3.2. States *Connection Creation*, *Connection Configuration* and *Connection Established* constitute Connection Establishment *PROFINET* operation. *PROFINET*'s Data Exchange operation are reflected in states *Input Data Exchange*, *Output Data Exchange*, *Acyclic Parametrization* and *Acyclic Reading Data*.

3.4.3 FSM *PROFINET* System

States and transitions. FSM System initializes with *Inactive* state and transits into *Powered On* as soon as *PROFINET* traffic triggers event *pn_traffic_detected*, show in fig. 3.21. FSM follows through the transitions as and when the triggering event is detected in network traffic. The protocol frame triggering the events for *PROFINET* operation transitions of a system are summarized in table 3.3. Event *all_connections_established* is triggered when all the FSM *PROFINET* Connection instances have arrived in state *Connection Established*.

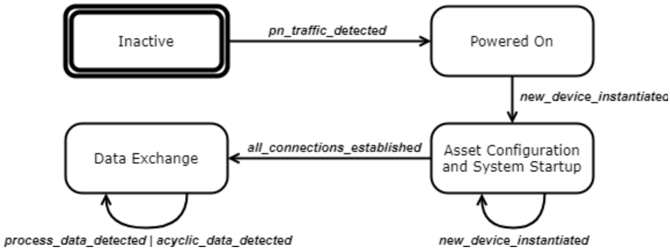

Figure 3.21: *PROFINET* System State Machine.

Table 3.3: Summary of FSM System transitions with *triggering event*, *event detection protocol* and *PROFINET operation*.

Source	Target	Trigger Event	Protocol Frame	*PROFINET* Operation
Inactive	Powered On	*pn_traffic_detected*	*LLDP*	Asset Discovery
			DCP Identify request	Address Resolution
Powered On	System Startup	*new_device_instantiated*	*Connect request*	Connection Establishment
System Startup	System Startup	*new_device_instantiated*		
System Startup	Data Exchange	*all_connections_established*	*PNIO*	Data Exchange
Data Exchange	Data Exchange	*process_data_detected*	*PNIO*	
Data Exchange	Data Exchange	*acyclic_data_detected*	*Write/Read*	

Relationship between states and *PROFINET* operations. The transitions between states reflecting the progress of *PROFINET* operations of a system are outlined in table 3.3. State *Powered On* reflects either *PROFINET* operation Asset Discovery & Neighbourhood Detection or Address Resolution if the event *pn_traffic_detected* is triggered by *LLDP* or *DCP Identify request* frames, respectively. State *Asset Configuration & System Startup* (referred as *System Startup* in table 3.3) reflects Connection Establishment *PROFINET* operation. *PROFINET*'s Data Exchange operation of cyclic and acyclic data transmission is reflected in state *Data Exchange*.

3.4.4 POET: A Framework for *PROFINET* Operations Enumeration and Tracking

PROFINET system, connection and device FSMs are realized in a *python*-based framework, termed as PROFINET Operations Enumeration and Tracking (POET), for enumeration and tracking *PROFINET*-based industrial network communication. It is implemented with *pytransitions* [Pyt] and logs transitions in FSM instances of *PROFINET* System, Connection and Device continuously. Each FSM *PROFINET* System instance is identified by a name given while initialization, whereas the device name extracted from network traffic (*DCP Identify request*/*LLDP* frame) is used for identifying FSM *PROFINET* Device instance. FSM *PROFINET* Connection instance is identified by the connection identifier created from concatenating MAC addresses of devices.

POET clearly satisfies the BSI's "*Category A - general requirements*" outlined in section 1.1.1, for an anomaly detection system to identify communicating devices, protocols and communication links (modeled as *PROFINET* Connection) in the industrial network.

3.5 Discussion and Summary

3.5.1 Anomaly Detection with POET

Industrial networks are vulnerable to different threat behaviours, each utilizing different techniques to exploit industrial network characteristics. *MITRE ATT&CK for ICS* [Str18] is a knowledge base of such industrial system targeted threat behaviours, collected through cyber threat intelligence reports of known cyber incidents, refer section 2.2. Some threat behaviours (such as *Modify Parameter*, *Denial of Service*) are targeted at the industrial network operation to disrupt the underlying industrial process.

Pfrang et al. [Pfr17] outlined threat scenarios targeted at real-world *PROFINET*-based systems with two different techniques to take over control of a *PROFINET* device. Within *PROFINET* networks, the devices are identified through the assigned logical name (refer section 3.3) for process data exchange.

The first attack of [Pfr17] demonstrated how an attacker changes the name of a device utilizing *PN-DCP* protocol and disconnecting it with other devices. A similar attack was performed on the *Festo Demonstrator* and POET was employed to monitor network traffic. The attack triggered events which aren't allowed for FSM Device instance of 'festo-motor-scheibe' and were reported in POET's logger, as shown in fig. 3.25. The blue edges are the *valid PROFINET* operation transitions triggered by the appropriate protocol events (a, b, .., f) as shown in fig. 3.22 - fig. 3.24. As per the *PN-DCP* protocol specification, the packets (x) are *valid*, however, they violate *PROFINET* device's *valid* operation transition represented as red edges, and thus detected.

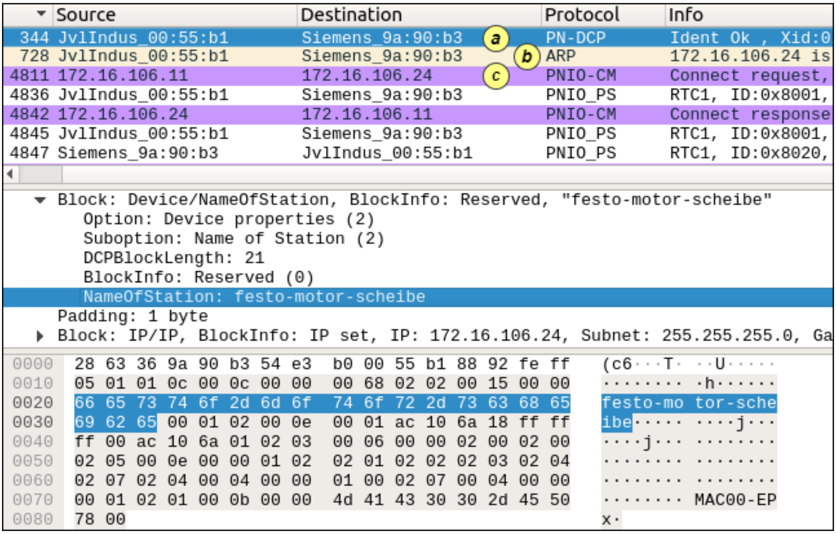

Figure 3.22: The network events, (a) *device_identified*, (b) *check_ip_duplication* and (c) *create_connection*, triggered by the network packets sent in the traffic of the *Festo Demonstrator* with the POET.

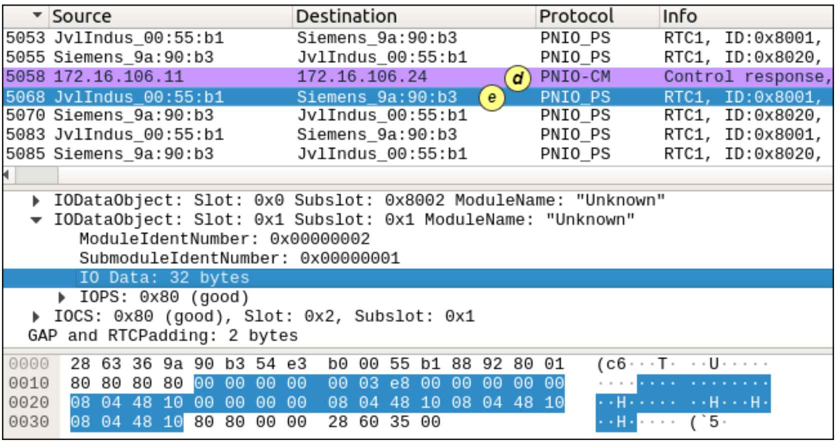

Figure 3.23: The network events, (d) *device_configured* and (e) *initiate_process*, triggered by the network packets sent in the traffic of the *Festo Demonstrator* with the POET.

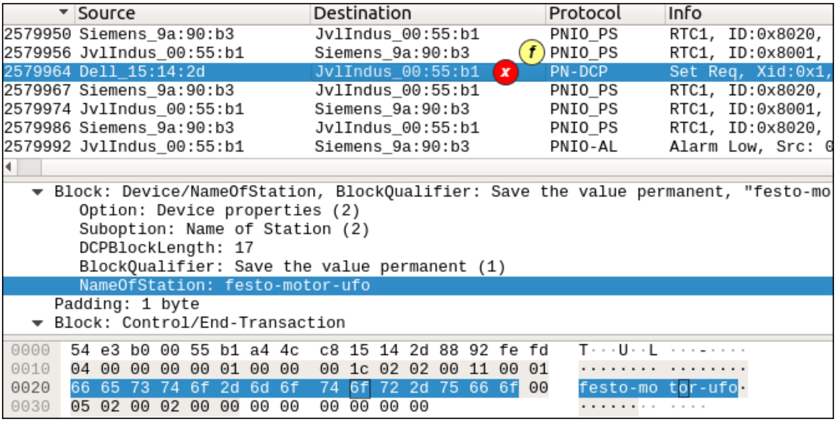

Figure 3.24: The network events, (f) *process_data_sent* and (x) *identify_device*, triggered by the network packets sent in the traffic of the *Festo Demonstrator* with the POET.

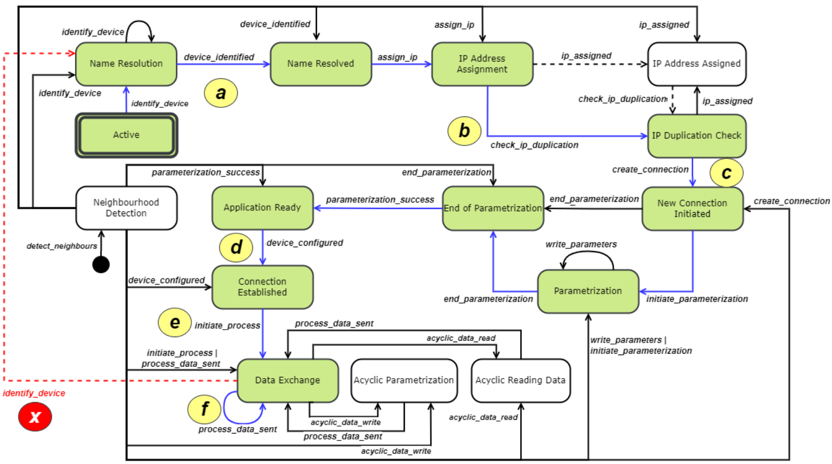

Figure 3.25: The detection of Rename Attack on the *Festo Demonstrator* with the POET. The network event "(x) *identify_device*" triggered an invalid transition to 'Name Resolution' operation.

[Pfr17] outlined the second attack which is essentially disrupting *PROFINET* network operations by establishing new connection with device. This action initiates Connection Establishment *PROFINET* operation which isn't valid transition state for FSM Device instance. Employing POET in such scenarios would also detect this attack and enhance visibility to unwarranted events in *PROFINET* networks with explanation.

POET's capability to detect the two attacks outlined in [Pfr17] satisfies the BSI's "*Category B - unusual or exceptional activities in an ICS network*" requirements outlined in section 1.1.1. It identifies the change in protocol communication between industrial components and detects new devices in the network through FSM Device and Connection models. Additionally, any other attacks that would violate the validity of an industrial operation through protocols other than *PN-DCP* and *PN-CM* would be detected by POET.

3.5.2 Review

Protocol-analysis based IDS have been proposed for industrial protocols such as *DNP3*, *Modbus/TCP*, *GOOSE*, etc. [Hu18], where protocol specifications are utilized to build system profile and the deviations are reported. The proposed FSM-based framework, POET, can be categorized along with them as Protocol-analysis based IDS for *PROFINET*.

POET is the first-of-its-kind Protocol-analysis based IDS for *PROFINET* to incorporate empirical behaviour of *PROFINET* system collected from real-world systems. The current version of POET uses empirical information collected from *PROFINET* systems incorporating Siemens PLC, it could vary with other PLC environments (eg. CODESYS [17]), devices and can be adapted. The extraction of network events triggering the transitions of industrial operations would be adapted to the new protocol communication stack.

POET offers insights into the operations of *PROFINET*-based systems at ascending granularity of system, connection between devices and device. This granularity helps to specify events proficiently to be used in downstream analysis for anomaly detection. For example, [Pfr17] outlined enhanced Snort for

PROFINET which was used to detect the two attacks mentioned in their work. It can be integrated with POET to trigger alarms when unwarranted transitions occur.

At the end, this chapter demonstrates a successful workflow to interpret an industrial protocol specification and the empirical information collected from its real-world industrial system usage to design a Protocol-analysis based IDS. A similar workflow could be utilized for another industrial protocol such as *EtherNet/IP* utilizing the outlined FSM models at different granularities of system, connection and device. The FSM models would have to be adapted for states and triggering events.

3.5.3 Conclusion

In this chapter, the *PROFINET* network traffic is mapped to different *PROFINET* operations for interpreting the underlying status of industrial communication. In section 3.3 the solution to research problem 1 is outlined, where different protocols associated with *PROFINET* operations are mapped to *PROFINET* network operations. For every network operation, the role it plays within *PROFINET*-based automation system communication and the network protocol utilized to achieve the goal has been presented. Thus satisfying the BSI's "*Category A - general requirements*" for network transparency as outlined in section 1.1.1.

In addition, the protocol associated communication behaviour and their detection through protocol frame analysis has been outlined. As the solution to research problem 2, section 3.4 modelled operations of *PROFINET* system, connection and device as Finite State Machines (FSM) to systematically enumerate and track *PROFINET* operations. *PROFINET* Device, Connection and System FSMs are realized in a *python*-based framework named PROFINET Operations Enumeration and Tracking (POET). Its successful usage as Protocol-based IDS in detecting cyber attack on real-world *PROFINET* demonstrator has been presented in section 3.5.1. This demonstrates POET as an anomaly detection solution that satisfies the BSI's "*Category B - unusual or exceptional activities in an ICS network*" requirements as outlined in section 1.1.1.

In conclusion, the challenge of self-learning *PROFINET*-based industrial communication networks is solved through interpretation of network traffic to *PROFINET* operations. The workflow developed in this chapter to interpret an industrial protocol's specification with the empirical information from its real-world usage to develop an anomaly detection system can be replicated further.

Data Exchange dominates the *PROFINET* operations where the actual industrial process is realized. In the next chapter, the research problems associated with analysing the *PNIO* frames for anomaly detection of industrial process behaviour through *Byte Profiling* is being addressed.

4 String-based Self-learning Process Behaviour from Network Traffic

An industrial process at its core is the interplay of instruction exchanges between actuators/sensors and controllers of an industrial control network. It could be either of 3 types, characterized by the stationarity of process data: *discrete*, *continuous*, or *batch* [Wil18]. Discrete processes are non-stationary as the process data vary at fixed time intervals and in a fixed order following a time schedule. Automotive, manufacturing and textile industry are its application domains. In continuous processes, the process data output remains temporally constant, and hence is stationary. Petrochemical industry and refineries are some of its application domains. Within batch processes, the goods are processed continuously at fixed time schedules. Breweries, pharmaceutical and food industry are driven in a batch process.

Industrial processes have stringent deterministic requirements to transmit information in real-time with low jitters. Additionally, the process parameters have to be transmitted periodically in a fixed order while maintaining temporal consistency. In order to fulfil high determinism requirements, industrial communication protocols implemented reduced OSI/ISO stack architectures to avoid transmission delays of information processing between layers.

Since 2000, Advanced Persistent Threats (APT) modified the instruction exchanges within industrial processes, utilizing industrial protocols, to cause structural damage to components (*Stuxnet*) or HSE hazards (*Industroyer/CrashOverride, Triton*) [Mak21]. Continuous monitoring and analysis of industrial process could detect the initiation of such attacks and reduce dwell time to accelerate mitigation. The analysis requires understanding the process behaviour of an industrial system to distinguish an anomaly from normality. These anomalies represent malfunctioning of components due to modification of process parameters or change in overall industrial process logic as effects of the cyber threat. In a self-learning setup, where the information on process parameters is not at one's disposal, the process behaviour is interpreted through analysis of industrial network traffic. With chapter 3 as foundational knowledge on *PROFINET* operations, in this chapter, the discrete process behaviour of *PROFINET*-based industrial system is modeled from its network traffic.

In order to understand the underlying industrial process from *PROFINET* network traffic, a closer look at constituents and characteristics of industrial process is required. An industrial process encompasses multiple sub-processes that are executed in periodic order at deterministic intervals as programmed during System Engineering *PROFINET* operation (refer section 3.3). Each *sub-process* is represented by the *connection* between a pair of controller and device, established through *PN-CM* protocol (refer section 3.3.3). Within this sub-process, the process data consisting of process parameters are exchanged periodically in strict intervals. The periodic and deterministic requirements of industrial process defines its *spatio-temporal* characteristics. In particular, the periodicity in process data exchange represents the *spatial* characteristic, and deterministic intervals the *temporal* characteristic of an industrial process and its constituent sub-processes. In addition, the *production cycle* of industrial process is defined by the deterministic and periodic execution of process parameters of constituent sub-processes.

This chapter addresses the challenge of representing the *spatio-temporal* characteristics of an industrial process as *String* and self-learning the process behaviour for anomaly detection. In section 4.1, the research problems concerning representation and learning are formulated. The solutions are proposed in section 4.2 - section 4.5. In section 4.6, the usage of the proposed framework for production cycle and anomaly detection tasks on the *Festo Demonstrator* are presented. The section ends with chapter's review and conclusion.

4.1 Research Problem

An *adversary* targets the industrial process through exploiting its *spatial* and *temporal* characteristics. With the knowledge of process parameters and employed industrial components, the *adversary* could either change the component configurations or process parameters. The manipulation of process parameters could be triggered to:

i set the component in an operation state that violates the execution order of the process' constituent sub-process parameters (*spatial* characteristic exploitation).

ii delay the communication between process parameter exchanges to have cascading effect on the process (*temporal* characteristic exploitation).

Protocol analysis-based Intrusion Detection Systems (IDS), such as POET (refer section 3.4.4), and traffic mining-based IDS does not have insights into the process data being exchanged in an industrial control network. Hence, process analysis-based IDS with insights into industrial process characteristics are employed for detecting the *adversary*'s actions [Hu18]. In general, process variables are known to the process analysis-based IDS for process behaviour analysis [Had14, Cas15, Ant19, Por21]. As already pointed out previously, in a self-learning setup, the explicit knowledge on process variable is not available, the process behaviour needs to be interpreted through analysing the industrial network traffic.

In chapter 3, the communication handshake between a Controller and Device to establish a connection as channel for cyclic and acyclic data exchange was explained, refer section 3.3.3. Within this *PROFINET* operation, the parameters for cyclic data exchange such as transmission frequency, data length, specification of input/output data, etc are transmitted to the Device by the Controller. This information is utilized to extract process payload bytes from *PNIO* frames in input/output direction. Analysis of extracted payload bytes from different *connections/sub-processes* to interpret underlying process behaviour as its spatio-temporal characteristics is the addressed research problem of this chapter. In particular, the spatio-temporal characteristics extraction and modeling at different granularities of sub-process (connection between Controller and Device) and process (industrial process with all connections). The research problem can be reformulated as following question:

1 How to model *spatial* characteristics of industrial (A) sub-process and (B) process? [*Addressed in section 4.3*]

2 How to model *temporal* characteristics of industrial (A) sub-process and (B) process? [*Addressed in section 4.4*]

3 How to combine *spatial* and *temporal* characteristic models of industrial (A) sub-process and (B) process? [*Addressed in section 4.5*]

4.2 Payload Bytes Profiling

The quest of learning the discrete process behaviour of an industrial system from its network traffic begins with outlining the requirements and assumptions made for the analysis. Then the framework for capturing the *spatial* and *temporal* characteristics of an industrial process is outlined.

4.2.1 Requirements

The process data needs to be extracted from payloads of network packets. Even though the industrial protocols are not encrypted, the process data being transferred and its values within the payload are unknown to a process semantics-agnostic sniffer. *Modbus* protocol is an exception to the restriction on interpreting process values from network traffic for analysis. In particular, Hadžiosmanović et al. [Had14] extracted the process variables from the *Modbus* traffic for process analysis-based threat detection. Within the *PROFINET* environment, GSD files of *PROFINET* device provided by the manufacturer contain information on which modules contain which device-specific physical parameters and at what positions of the payload (refer section 3.3). In a self-learning scenario, access to GSD files is not at the disposal, however, the position of these process parameters within the payload and their length is exchanged during *Connection Establishment* handshake for input and output data exchange (refer section 3.3.3). This information is utilized to extract payload bytes containing the process values embedded within from *PNIO* frames.

In addition, through *Connection Establishment* operation, which *PROFINET* component is the Controller and which is the Device is known, as Controller always initiates *PN-CM* handshake. For example, consider component *A* sending *Connect request* frame to component *B*. After successful *Connection Establishment* operation, *A* is designated as the Controller and *B* as the Device.

In the traffic captured from the *Festo Demonstrator* (refer chapter 2), 5 *Controller~Device* connections were found: *PLC-1~Turntable-Motor*, *PLC-1~BK-1*, *PLC-2~BK-2*, *PLC-3~Lift-Motor* and *PLC-3~BK-3*. The extracted process bytes from payload for output (Controller⟶Device) and input (Controller⟵Device) data exchanges are stored with corresponding timestamps.

As already mentioned in chapter 2, before the demonstrator is brought into production, *referencing* is done to bring the system components at their initial states of industrial process. In the captured process data, the data points of '*referencing*' are explicitly distinguished from '*actual process*'.

4.2.2 Assumptions

For a discrete industrial process, the Controller is hypothesized to send a finite set of process parameters to the Device periodically at deterministic intervals with jitter. The Device communicates its dynamic process values which either varies in case of Motion Drives/Motor or remains finite in case of Bus Couplers, as observed in the demonstrator. As the output process data exchange is relatively consistent to input data, only the output process data exchange is henceforth considered for modelling the *spatio-temporal* characteristics of industrial process.

4.2.3 Framework

The representation of network traffic as a String, either payload information or network information, for intrusion detection has been explored by PAYL [Wan04], POSEIDON [Bol06], Anagram [Wan06] and McPAD [Per09]. These frameworks analyse payload content of each packet for positional variations at a time. However, they do not correlate the order of packets in the traffic or the duration between the transmission of different packets, which define the *spatio-temporal* characteristic of industrial process.

To represent the finite set of payload bytes of the process/sub-process and to characterise their *spatial* and *temporal* behaviour, a framework is proposed to encode (i) network traffic as sequence of alphabet-encoded process payload bytes, and (ii) interval between payload transitions. Since the payload bytes of process/sub-process are being profiled, the framework is called *Payload Bytes Profiling (PBP).*

The *spatial* characteristic of process/sub-process is represented by sequences of alphabet-encoded payload bytes, where the change from one alphabet to neighbouring alphabet represents a transition in process/sub-process state. The time duration between the transitions represents the *temporal* characteristic of the process/sub-process. The *transition profile* is a substring extracted from encoded traffic consisting of ordered set of alphabet-encoded payloads representing recurring transitions of the process/sub-process cycle. The intervals between the transitions of *transition profile* defines the *interval profile* for the modeled process/sub-process.

Following are the steps in creation of *Payload Bytes Profile* for a connection (sub-process) explained with example of *PLC-1~Turntable-Motor*:

(a) **Payload Bytes Extraction.** Using the position and length of process parameter set during *Connection Establishment* operation, the payload bytes are extracted from *PNIO* frames in output and input data exchange, as shown in fig. 4.1. As reasoned before, only the output direction is being considered further.

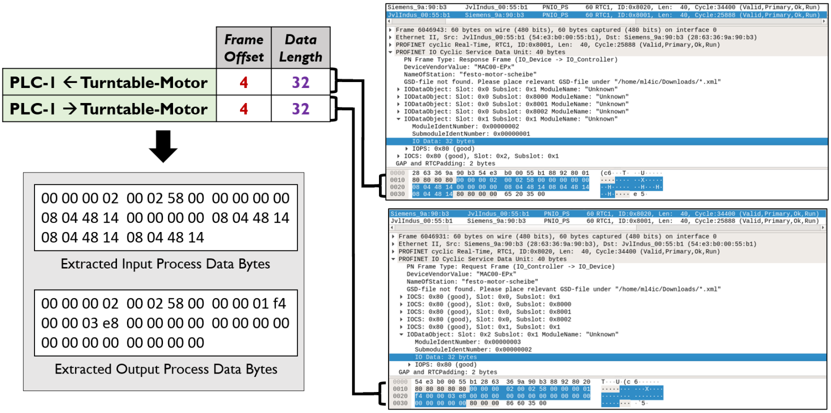

Figure 4.1: Process payload bytes extraction between *PLC-1* and *Turntable-Motor* from *PNIO* frames (*Wireshark* snippet), using parameters extracted from the *Connect* frame.

(b) **Payload Bytes Encoding.** The payload bytes are converted to hexstring or bitstring representation, depending on the length of the payload bytes. The payload byte of length <1 is considered to be encoded as *bit*-level process parameters, hence, the bitstring representation. Each hex/bit-string is mapped to an alphabet from the English language, as shown in fig. 4.2. If the number of finite hex/bit-string surpasses 26, a new encoding scheme could be used, which remain consistent across the framework.

Payload Hexstring	Encoded Alphabet
0000000000000000000000000000000000000000000000000000000000000000	A
00000000000000000000000000003e8000000000000000000000000000000000	B
00000001000003e7ffffff38000003e800000000000000000000000000000000	C
000000020000a924000000c8000003e800000000000000000000000000000000	D
000000020000a92400000000000003e800000000000000000000000000000000	E
000000020000c800000001f4000003e800000000000000000000000000000000	F
0000000200019000000001f4000003e800000000000000000000000000000000	G
0000000200025800000001f4000003e800000000000000000000000000000000	H
0000000200032000000001f4000003e800000000000000000000000000000000	I
000000020003e800000001f4000003e800000000000000000000000000000000	J
000000020004b190000001f4000003e800000000000000000000000000000000	K
0000000200057800000001f4000003e800000000000000000000000000000000	L
0000000200064000000001f4000003e800000000000000000000000000000000	M
0000000200000000000001f4000003e800000000000000000000000000000000	N

Figure 4.2: Alphabet-encoding of payload hexstring of sub-process *PLC-1~Turntable-Motor*.

(c) **Network Traffic Encoding.** The timestamped payload bytes extracted are ordered in their occurrences and encoded with corresponding alphabet encodings. In fig. 4.3, the complete encoded traffic for a sub-process *PLC-1~Turntable-Motor* captured from the demonstrator is shown.

Encoded Traffic
ABCDEFGHIJKLMNFGHIJKLMNFGHIJKLMNFGHIJKLMNFGHIJKLMNFGHIJKLMNFGHIJKLMNFGHIJKLMNFGHIJKLMNFGHIJKLMNFGHIJKLMNFGHIJKLMNFGHIJKLMNFGHIJKLM

Figure 4.3: Alphabet-encoded network traffic of sub-process *PLC-1~Turntable-Motor*.

(d) **Transition Interval Computation.** The transition interval between transitions is computed as the number of timestamps before the observed payload byte, A, changes to another, B, as shown in fig. 4.4. The transition intervals for all the observed transitions of a sub-process in the traffic are collected and saved for later temporal modeling.

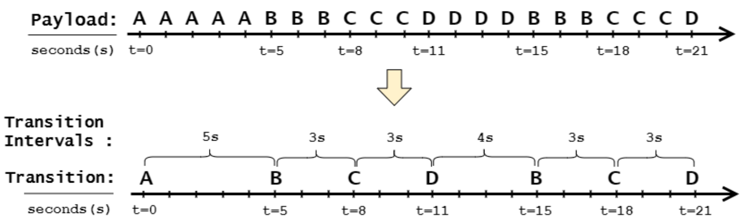

Figure 4.4: The demonstration of encoding the transition intervals from encoded traffic.

(e) **Transition Profile Extraction.** The *transition profile* for a sub-process is extracted from analysing it *spatial* characteristics, which is presented in section 4.3.

(f) **Interval Profile Extraction.** The *interval profile* is the result of *temporal* characteristics modeling of all the constituent transitions of *transition profile* of the sub-process, explained in section 4.4.

Additionally, the different encoded payload bytes of constituent sub-processes of industrial process are collected and re-encoded with corresponding sub-process index as suffixes, as shown in fig. 4.5.

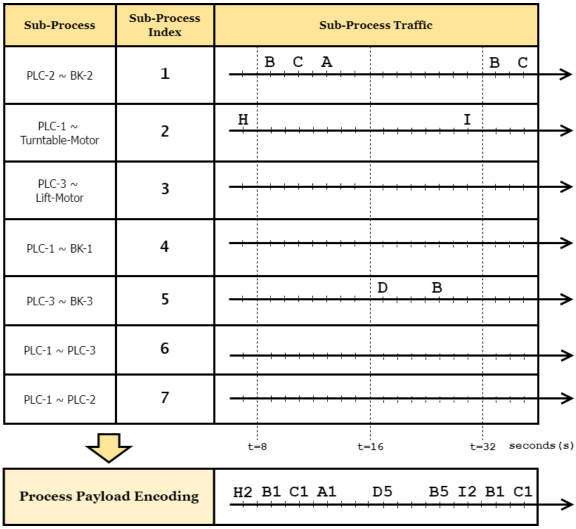

Figure 4.5: The demonstration of encoding traffic of the *Festo Demonstrator*'s process.

In summary, *Payload Bytes Profile* for *Connection X*, $\mathbf{P_X}$, extracted from encoded process data exchange traffic, $\mathbf{S_X}$, for observed duration $\mathbf{D_X}$, can be defined as $\mathbf{P_X} = (\mathbf{F_X}, \mathbf{A_X}, \mathbf{E_X}, \Lambda_\mathbf{X}, \Gamma^\mathbf{X})$, where

- $\mathbf{S_X} = (\mathbf{s_1}, \mathbf{d_1}), .., (\mathbf{s_n}, \mathbf{d_n}), \mathbf{s_i} \in \mathbf{A_X}, \mathbf{s_i} \neq \mathbf{s_{i+1}}$,
- $\mathbf{D_X} = (\mathbf{d_n} - \mathbf{d_1} + 1), \mathbf{d_i} < \mathbf{d_{i+1}}$, and
- $\mathbf{F_X}$: 'finite' set of payload strings (hex/bit-string) of *Connection X*
- $\mathbf{A_X}$: alphabet set s.t. $|\mathbf{A_X}| = |\mathbf{F_X}|$
- $\mathbf{E_X}$: encoding function, $\mathbf{F_X} \rightarrow \mathbf{A_X}$ s.t.
  $\forall \mathbf{i}, \mathbf{j} \in [1, |\mathbf{F_X}|], \mathbf{i} \neq \mathbf{j}, \mathbf{a_i}, \mathbf{a_j} \in \mathbf{A_X} : \mathbf{a_i} \neq \mathbf{a_j}$
- $\Lambda_\mathbf{X}$: ordered alphabets representing *Connection X*'s cyclic data,
  $\Lambda_\mathbf{X} \subseteq \mathbf{A_X}$
- $\Gamma^\mathbf{X}$: transition intervals between elements of $\Lambda_\mathbf{X}$,
  $\forall \mathbf{i}, \mathbf{j} \in [1, |\Lambda_\mathbf{X}|], \mathbf{j} > \mathbf{i}, \gamma_i, \gamma_j \in \Lambda_\mathbf{X}, \Gamma_\mathbf{ij}$ = transition interval model for transition $\gamma_i \rightarrow \gamma_j$

In next sections, the *spatial* and *temporal* characteristic modeling methodology employed to extract *transition* and *interval profiles* of a sub-process/ process's *Payload Bytes Profile* is detailed. These *profiles* are then used in conjunction for *spatio-temporal* modeling of corresponding sub-process or overall industrial process of the system.

4.3 Spatial Characteristic Modeling

The *spatial* characteristics of an industrial process is the periodicity in process data exchange. The process parameters are sent by the Controller to the Device in a definite order cyclically. In the encoded traffic, the alphabet-encoded payload bytes follows an order, which repeats contiguously, as it can be observed in fig. 4.6 for sub-process *PLC-1~Turntable-Motor.* Transitions between the payload bytes occur in the order: F⟶G, G⟶H, H⟶I, I⟶J, J⟶K, K⟶L, L⟶M, M⟶N, N⟶F, and repeat. The *transition profile* for this sub-process is FGHIJKLMN, and extracting such *transition profile* from encoded traffic of a sub-process/process requires string-based algorithms. An efficient string-based algorithm is required that extracts the *transition profile* over a large text.

String Basics. For the ease of exploring the string-based algorithms, certain definitions need to be established. Consider a string **S** of length n over an alphabet $\mathcal{A}$ with size $|\mathcal{A}| = \sigma$. A *substring* of **S**, **S[i .. j]**, denotes all characters of **S** starting index i and ending at index j of **S**, for $i \leq j$. The substring **S[0 .. i]** denotes a *prefix* of **S** of length $(i-1)$, and the substring **S[i .. n-1]** is the i-th *suffix* of **S**, denoted by **S(i)**. For example, given string **T**=ABCDBCDB, $|\mathbf{T}| = 8$, **T[0 .. 4]**=ABCD is the *prefix* and **T(3)**=**T[3 .. 7]**=DBCDB is the 3-rd *suffix.*

The *transition profile* extraction can be reformulated as finding the longest repeating and non-overlapping substring (LRS) from the encoded traffic. For example, for **T**, the LRS is 'BCD' repeating 2 times i.e. A <u>BCD</u> <u>BCD</u> B.

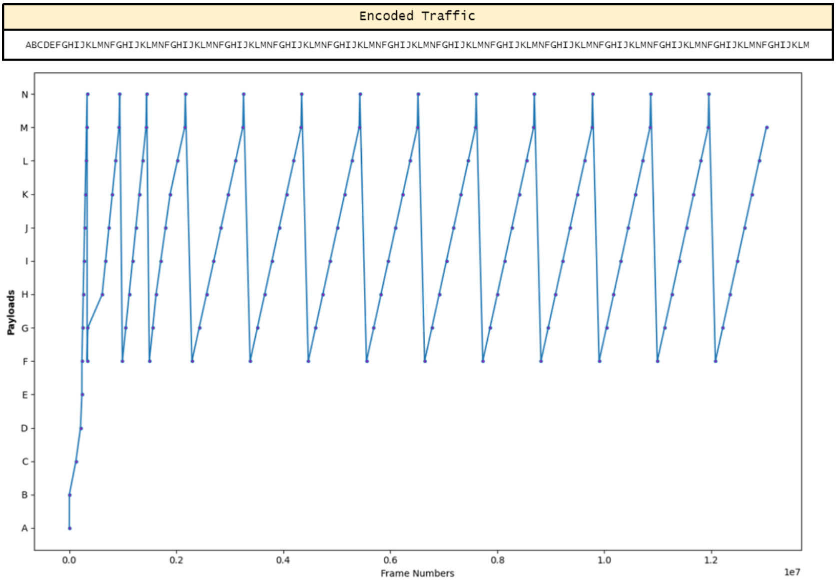

Figure 4.6: *Spatial* Characteristics of sub-process *PLC-1~Turntable-Motor.*

Dynamic Programming. With Dynamic Programming, the LRS for a string of length n can be found in $\mathcal{O}(n^2)$ time complexity. As outlined in listing 4.1, for a given input string $\mathbf{s}$, a 2-dimensional array is initialized with zeros. Whenever the characters of two substrings match, their indices are saved. To avoid overlapping substrings, the length of LRS should be less than the difference between indices of substrings. The LRS is found by taking the maximum value from 2-dimensional array and the ending index of suffix. Building of 2-dimensional array takes $\mathcal{O}(n^2)$ time, hence the similar overall complexity. The process of finding the LRS of string $\mathbf{T}$ is shown in fig. 4.7.

	j	0	1	2	3	4	5	6	7
i		A	B	C	D	B	C	D	B
0	A	0	0	0	0	0	0	0	0
1	B	0	0	0	0	1	0	0	1
2	C	0	0	0	0	0	2	0	0
3	D	0	0	0	0	0	0	3	0
4	B	0	0	0	0	0	0	0	0
5	C	0	0	0	0	0	0	0	0
6	D	0	0	0	0	0	0	0	0
7	B	0	0	0	0	0	0	0	0

T = 'ABCDBCDB'

suffixEndIndex (m) = 3

P_Length (n) = 3

P = T[(n - m+ 1) : m] = 'BCD'

Figure 4.7: The demonstration of extracting Longest Repeated and Non-Overlapping Substring (LRS) using Dynamic Programming.

Listing 4.1: The Dynamic Programming for LRS extraction.

```
Input String: s
Output String: p

n = len(s)

# Matrix to store values
m = [[0 for i in range(n + 1)]
        for j in range(n + 1)]

p = "" # Output
p_len = 0 # Length of output

suffixEndIndex = 0
for i in range(1, n + 1):
  for j in range(i + 1, n + 1):

    # m[i-1][j-1] < (j-i) to avoid overlapping
        if (s[i - 1] == s[j - 1] and m[i - 1][j - 1] < (j -
            i)):
      m[i][j] = m[i - 1][j - 1] + 1

        # updating maximum length of the substring
        # and updating the ending index of the suffix
```

```
        if (m[i][j] > p_len):
            p_len = m[i][j]
            suffixEndIndex = max(i, suffixEndIndex)
        else:
      m[i][j] = 0

# If we have non-empty result, then insert
# all characters from first character to
# last character of string
if (p_len > 0):
    for i in range(suffixEndIndex - p_len + 1,
        suffixEndIndex + 1):
    p = p + s[i - 1]
```

Suffix Tree and Regular Expression Search. The search for an efficient string-based algorithm over a large text in comparison to Dynamic Programming, found its nest in the field of Computational Biology, in particular Genomics. Genomics is analysis of the genome of an organism, a biological sequence of Deoxyribonucleic acid (DNA), Ribonucleic acid (RNA) or proteins, to find related ancestors (*phylogenetics*), repeated segments containing functional information and their mutations (*genetics*) and many more biological properties. A genome is usually very large in size, for example, human genome is upto 3.2 billions base pair (base pair = [A, T, G ,C]) [Col03]. Dan Gusfield [Gus97] described different data structures and algorithms for string manipulations on such larger text, such as suffix tree, suffix array, etc. Amongst them, Suffix Tree has been widely used in biological sequence analysis tasks such as finding tandem repeats, which are repeated segments appearing adjacent to each other. The other applications of Suffix Tree are in exact string matching, data compression, longest repeated substring search, etc [Abo09].

Suffix Tree was introduced by Pete Weiner, in 1973 [Wei73], with linear construction time $\mathcal{O}(n)$ for a string of length n. McCreight, in 1976 [McC76], and Ukkonen, in 1995 [Ukk95], have incrementally improved on the space-complexity of constructing a suffix tree in linear time. In particular, Ukkonen's is an on-line algorithm, where the whole string doesn't have to be read and stored at once, and is easy to understand.

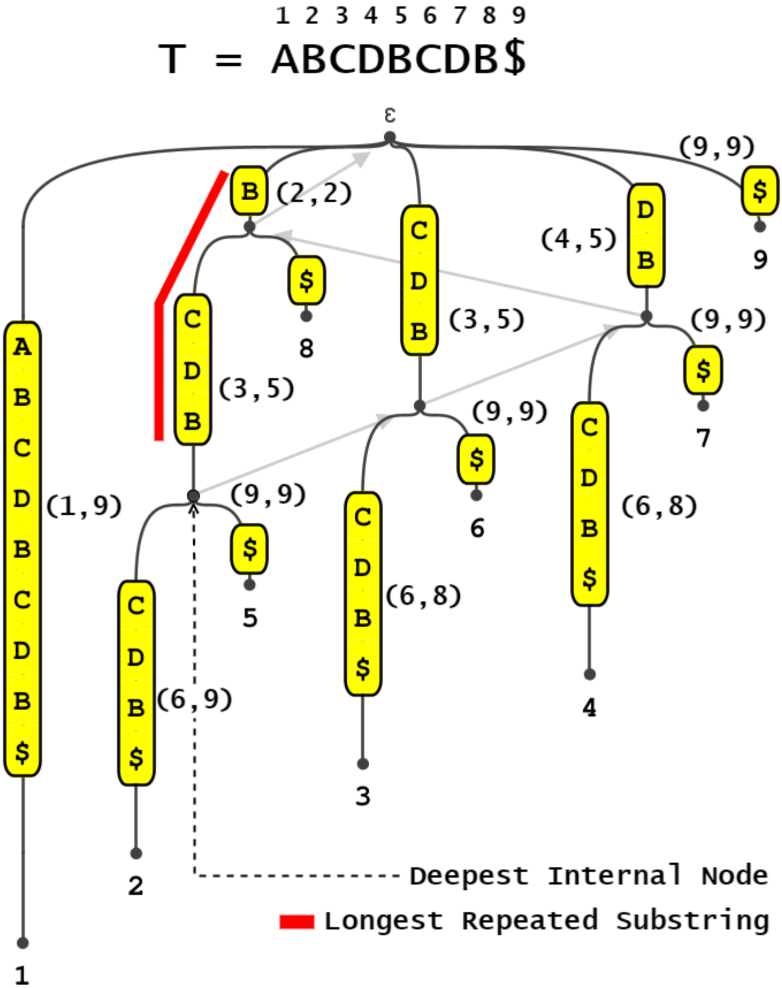

Figure 4.8: The demonstration of extracting Longest Repeated Substring using Suffix Tree.

A suffix tree for a string **S** of length n is a rooted directed tree such that:

- It has exactly n leaves numbered from 1 to n.
- Except the root, each interval node has at least two children.
- Each edge is labelled with a non-empty substring of **S**.
- No two outgoing edge-labels from a node begin with same character.
- For any leaf i, the i-th suffix is constructed by concatenating edge-labels on the path from root to leaf i i.e. **S**[i .. n], for $1 \leq i \leq n$.
- For any non-leaf node k, concatenation of edge-labels on the path from root to k is substring **S**[1 .. k] as its path label, and occurs z times, where z is the number of leaves in the subtree of node k.
- For any non-leaf node k with path label *ap*, where a is a character and p is a string, a suffix link is a pointer from k to another node whose path label is p.

In fig. 4.8, a suffix tree for string **T**=ABCDBCDB is shown. Every string is appended with the terminal character, $, to ensure that no suffix is prefix of another, and every suffix belongs to a leaf. For example, for a string PP, without the terminating character, the second suffix, which is a prefix of the first, won't find its place at the leaves of the suffix tree. Additionally, for linear space complexity, each edge-label, a substring of **S**, **S[i .. j]** is replaced with a pair of position indices (i, j).

In order to find the longest repeated substring in string **S** with a suffix tree, we search for the deepest internal node of the suffix tree and its path label is the solution [Sun09]. Such is the case because for a substring repeating at i and j positions in **S**, it has to be the common prefix of i-th suffix and j-th suffix. Hence, the longest repeated substring is found at the deepest internal node. As the suffix tree has at most n nodes, the traversal to find the deepest takes $\mathcal{O}(n)$ time. For our example string **T**, the longest repeated substring is BCDB, as shown in fig. 4.8.

The longest repeated substring search with suffix tree is linear in time in comparison to quadratic Dynamic Programming solution. However, the caveat with the suffix tree's solution is that it retrieves the longest repeated overlapping substring. In order to extract the non-overlapping string, a regular expression (RegEx) search is made recursively in linear time [Cox07]. At every step, the RegEx: `^(\w*).*\1\$` , finds the word group which is the suffix and prefix of the string, then it is sliced off from the end of the string. It is performed recursively until no more such word group can be found and the word group is returned as the non-overlapping repeating substring. For the output of suffix tree search, BCDB, the non-overlapping substring is BCD, which occurs 2 times in the input string **T** i.e. A BCD BCD B.

The longest repeating non-overlapping string of length m from a string of length n can be extracted from a string's suffix tree with additional RegEx operation in linear operations of $\mathcal{O}(n + k * m)$, where $m < n$ and it repeats atleast k times, for $2 \leq k \leq \frac{n}{k}$. The overall algorithm is outlined in listing 4.2 which is further used for *transition profile* extraction from the encoded traffic.

Listing 4.2: The RegEx-coupled Suffix Tree algorithm to extract LRS.

```
Input String: s
Output String: p

T = Build_SuffixTree(s)
T.doDepthFirstTraversal()
d = T.deepestInternalNode()

# Longest Repeated Substring
lrs = T.getPathLabel(T.root(), d)

# RegEx
p = Get_NonOverlapping_String(lrs)
```

The RegEx-coupled Suffix Tree algorithm is employed to extract *transition profile* for all the sub-processes. An example is outlined in fig. 4.9 to demonstrate the *transition profile* extraction for sub-process *PLC-1~Turntable-Motor*, resulting to FGHIJKLMN, as it was expected at the beginning of the section.

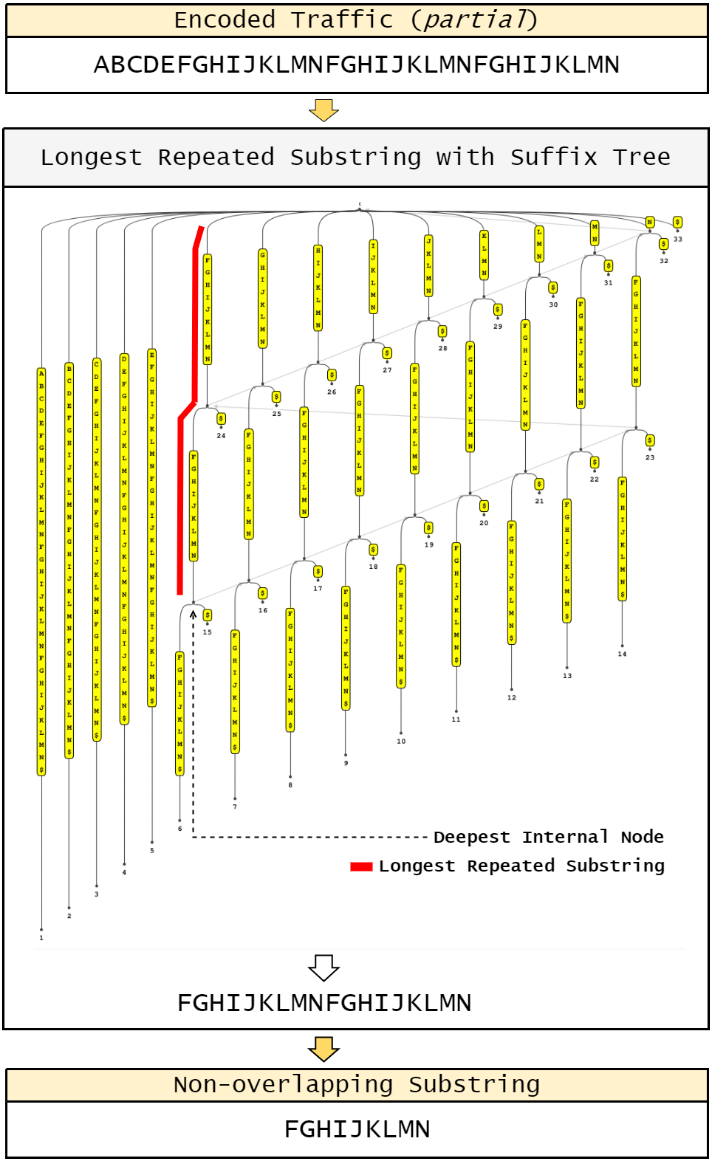

Figure 4.9: The demonstration of extracting *Transition Profile* of sub-process *PLC-1~Turntable-Motor* using Suffix Tree.

The *transition profile*, FGHIJKLMN, is used to (a) check if the order of transitions are being followed in the traffic, and (b) distinguish payload bytes outside the sub-process and flag their occurrences. In fig. 4.10, the payloads and the transitions outside the *transition profile* are flagged as '*invalid transitions*', represented with red coloured line. The red line at the beginning corresponds to '*referencing*' phase of the demonstrator, where the process values other than the ones constituting the 'actual process' are transmitted. This difference in process data exchange at different phases is clearly distinguishable using the *transition profile*.

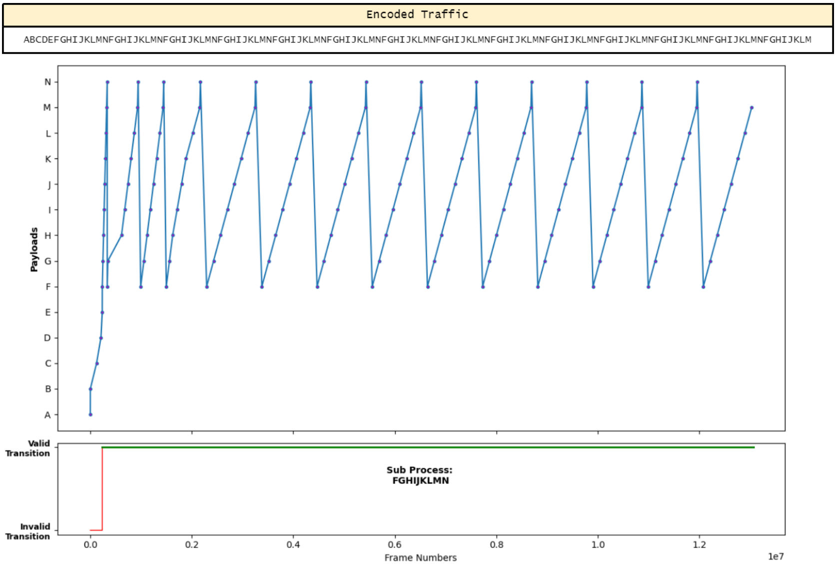

Figure 4.10: Classification of transitions as *Valid* or *Invalid* using *Transition Profile* of sub-process *PLC-1~Turntable-Motor* - FGHIJKLMN.

The RegEx-coupled Suffix Tree algorithm is applied over the encoded traffic of the process and '*valid/invalid transitions*' are detected, as shown in fig. 4.11. The red line at the beginning corresponds to ´*referencing*' phase of the demonstrator and only partial encoded traffic from the beginning is shown for brevity.

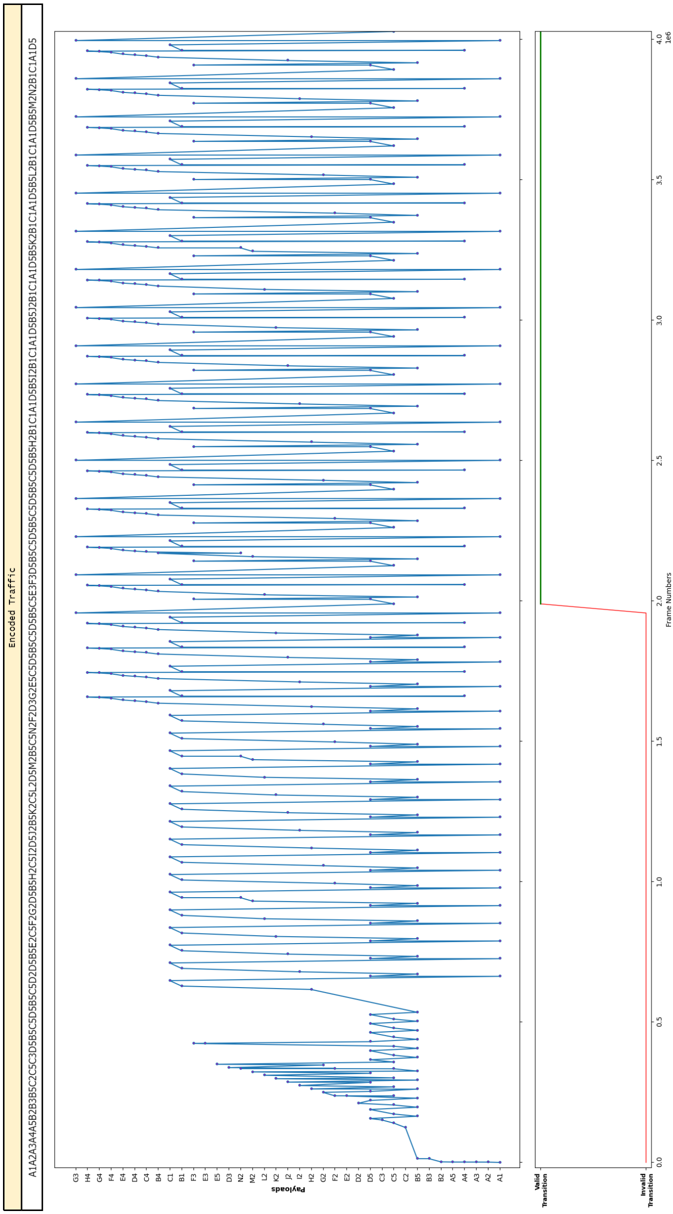

Figure 4.11: Classification of transitions as *Valid* or *Invalid* using *Transition Profile* of the *Festo Demonstrator*'s process.

4.4 Temporal Characteristic Modeling

The *temporal* characteristics of an industrial process is the deterministic intervals between the process data exchanges. At their core, the industrial processes are characterized by their deterministic time requirements, which required customized industrial protocols to satisfy them. Modeling the *temporal* characteristics contributes toward detecting subtle changes of the industrial process at fine-granularity level. A *classifier* is trained on transitioning time intervals between process parameter, sent from the Controller to the Device, collected during the "normal" operation of industrial process. Whenever the transition interval deviates from learned *temporal* characteristics, the *classifier* classifies it as an *anomaly* (or an *outlier*).

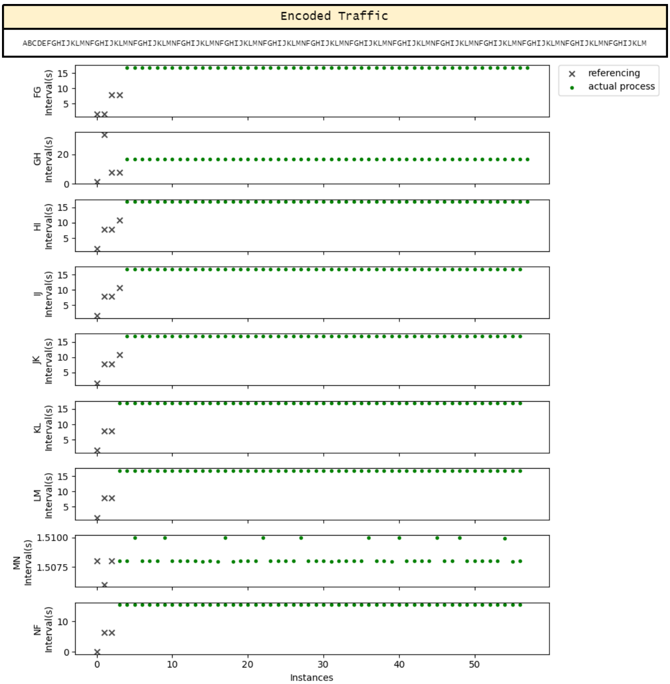

Figure 4.12: *Temporal* Characteristics of sub-process *PLC-1~Turntable-Motor*: intervals for every transition in *transition profile*.

For a sub-process/process, the transition intervals between payload bytes constituting its *transition profile* are modeled by set of *classifiers*, each for a transition, and collectively constitutes *interval profile* of the sub-process/process. As an example, for sub-process *PLC-1~Turntable-Motor*, the transitions between payload bytes of its *transition profile* (FGHIJKLMN) are FG, GH, HI, IJ, JK, KL, LM, MN, AND NF, and their corresponding transition intervals for the encoded traffic are shown in fig. 4.12. The ´*referencing*' and ´*actual process*' phases data points are differentiated as shown in the figure.

4.4.1 Model Selection

The selection of *classifier* for *interval profile* needs to satisfy following criteria:

- **Non-parametric method.** The transition intervals are collected in a self-learning environment where no prior information about the data is available. The parameters of the model are not foreseeable and are not fixed. Non-parametric methods are robust and employed in such scenarios, hence, the required *classifier* must be non-parametric.
- **Univariate analysis.** Only the interval feature is being model, therefore, the *classifier* must perform well on univariate data.
- **One-class classification.** The transition intervals are collected from the normal operations of the industrial process. All the collected data points belong to one class "normal". One-class classifiers that learn on a particular class and detect anomalies/outliers deviating from the learnt one-class features. Hence, the *classifier* must be an on-class classifier.

Based on above criteria, density-based Local Outlier Factor (LOF), boundary-based One-Class Support Vector Machines (OC-SVM) and tree-based Isolation Forest (IF) methods are selected. They have been widely used for anomaly/outlier detection in research as well as real-world applications with efficient performances [Pol22] . In addition, for a self-learning system, analysing the problem from different perspective brings in-depth insights.

4.4.1.1 Local Outlier Factor

Local Outlier Factor (LOF) is a density-based anomaly detection algorithm that compares density of a given data point to its neighbours to determine if its anomalous or not. The anomalous points are hypothesized to lie in lower density regions as compared to populous normal points. The idea of LOF was proposed by Breunig et al. [Bre00] to find anomalous points by calculating the local deviation of a given data point to its neighbouring points. The settings and process of calculating LOF of a data point p i.e. $LOF(p)$ are defined as follows:

k-distance & k-neighbours. The distance between point p and point o, where o is p's k-th nearest neighbour, defines *k-distance* of p i.e. $kd(p)$. All the points that are in neighbourhood of p at distance less than $kd(p)$ are its *k-neighbours* i.e. $N_{kd}(p)$.

Reachability distance. The *reachability distance* of point p to point o, $reachdist_{kd}(p, o)$, is the maximum of distance from o to p and *k-distance* of o. In other words, if p is in *k-neighbours* of o ($N_{kd}(o)$), then *reachability distance* is *k-distance* of o ($kd(o)$), otherwise its actual distance between them $d(p,o)$.

$$reachdist_{kd}(p, o) = \max\{kd(o), d(p,o)\}$$

Local reachability density. The *local reachability density* of point p, $LRD_{kd}(p)$, is the inverse of the mean of *reachability distance* of p from its neighbours. Intuitively, the lower the $LRD_{kd}(p)$, the less dense the p's neighbourhood and thus longer to reach its neighbourhood points.

$$LRD_{kd}(p) = \frac{1}{\sum_{o \in N_{kd}(p)} \frac{reachdist_{kd}(p,o)}{|N_{kd}(p)|}}$$

Local outlier factor. The *local outlier factor* of p, $LOF_{kd}(p)$, is the ratio of average *local reachability density* of its neighbours to its own *local reachability density*. Intuitively, if p is an *inlier*, the average *LRD* of its neighbours is same as the $LRD_{kd}(p)$, amounting the ratio to 1. Otherwise, the point p is an *outlier* with high LOF score.

$$LOF_{kd}(p) = \frac{\sum_{o \in N_{kd}(p)} LRD_{kd}(o)}{|N_{kd}(p)|} \times \frac{1}{LRD_{kd}(p)}$$

4.4.1.2 Isolation Forest

Isolation Forest (IF), proposed by Liu et al. [Liu08], is an ensemble of optimized binary trees (iTrees), where the data points are recursively partitioned until all the instances are isolated. The data instance with shortest average path from root is the anomalous point. The intuition behind this is that the anomalies are "rare" as compared to normal points resulting in fewer partitions, hence, shorter path in tree.

For a dataset with n instances, an Isolation Forest is built by creating multiple iTrees, each recursively partitioning the data points until all n instances are isolated. For each data instance x, $h(x)$ is its path length measured as the number of edges traversed from root to x in an iTree. It is normalized with the average path length $c(n)$. Liu et al. observed that an iTree is similar to Binary Search Tree (BST) and the value of $c(n)$ can be thought of as path length of unsuccessful search in the BST. The value of $c(n)$ is calculated based on following equation, where $H(i)$ is the harmonic number, which can be estimated by $\ln(i) + 0.5772156649$ *(Euler's constant)*:

$$c(n) = 2H(n-1) - (\frac{2(n-1)}{n})$$

The anomaly score for each data instance x is calculated as

$$s(x,n) = 2^{\frac{-E(h(x))}{c(n)}},$$

where $E(h(x))$ is the average path length of x from a collection of iTrees. An anomaly score of 0 indicates x to be normal, and closer to 1 indicates high probability of being anomalous.

4.4.1.3 One-Class Support Vector Machines

One-Class Support Vector Machines (OC-SVM), proposed by Schölkopf et al. [Sch99], are one-class classifiers derived from the SVM classifier [Bos92]. It finds a hyperplane that separates given data points of a particular class from the origin. Any new point near to the origin or on its side of hyperplane would be classified as an anomaly. The hyperplane must be positioned in such a way that it maximizes the distance between nearest sample of given class and origin, this distance is called "margin".

When the input space is not linearly separable, it is projected to a higher dimensional feature space F where the hyperplane can separate the data points using a non-linear function $\phi(.)$. The hyperplane is projected back to linear space and has a non-linear shape. In particular, $K(x,x_i) = \phi(x)^T\phi(x_i)$ is called the *kernel function*. The most widely employed *kernel function*s are linear, polynomial, sigmoid, and Radial Basis Function (RBF).

To prevent the classifier from overfitting, slack variables ξ are introduced to allow some data points to lie within the margin. [Sch99] defines the regularization parameter υ for tuning the trade-off between maximizing the margin and number of training data points to be within the margin.

The objective function of OC-SVM for minimization can be formulated as

$$
\begin{aligned}
\min_{w,\xi,\rho} \left\{ \frac{\|w\|^2}{2} + \frac{1}{n\upsilon}\sum_{i=1}^{n} \xi_i - \rho \right\} \\
\text{subject to: } w^{\mathrm{T}}\phi(x_i) \geq \rho - \xi_i \quad \forall i, i = 1, .., n \\
\xi_i \geq 0 \quad \forall i, i = 1, .., n,
\end{aligned}
$$

where, w is the orthogonal vector to hyperplane, n is total number of sample points, ξ_i is the slack variable, ρ is the margin. υ is the regularization parameter that characterizes the solution: (1) sets an upper bound on fraction of outliers, and (2) is a lower bound on number of training data points as Support Vectors.

Solving the minimization problem with Lagrange Multipliers and using kernel function for dot-product calculation, the decision function for a data point x becomes

$$f(x) = \mathrm{sign}(w^{\mathrm{T}}\phi(x_i) - \rho) = \mathrm{sign}(\sum_{i=1}^{n} \alpha_i K(x, x_i) - \rho).$$

Here, α_i are the Lagrange multipliers and every $\alpha_i > 0$ is weighted in the decision function thus supports the machine. OC-SVMs thus create a hyperplane characterized by w and ρ with maximal distance from the origin in feature space F and separating all points from the origin.

4.4.2 Performance Metric

The efficiency of a *classifier*'s performance in detecting anomalies/outliers is governed by the choice of the evaluation metric during training phase. Choosing an incorrect metric results in poor modeling and eventually poor performance on anomaly/outlier detection. In a balanced dataset, normal (positive) and anomalies/outliers (negative) are equally distributed and *confusion matrix* [21a] elements, shown in table 4.1, are calculated.

Table 4.1: The Confusion Matrix

	Predicted Postive	**Predicted Negative**
Positive (P)	True Positive (TP)	False Negative (FN)
Negative (N)	False Positive (FP)	True Negative (TN)

Different metrics are calculated from the *confusion matrix*:

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
- Sensitivity/Recall = $\frac{TP}{TP+FN}$
- Specificity = $\frac{TN}{TN+FP}$

When the distribution of classes is highly skewed with only one class being present, standard performance metric such as *Accuracy* do not suffice. There are no negative classes to account for and every *classifier* trained on this skewed distribution gives high accuracy.

During the training of a *classifier* for anomaly detection on "normal" operational data, its performance is evaluated on how well it has learnt the data characteristics to recall them. Hence, *Sensitivity/Recall* will be used as performance metric during training.

When a skewed dataset with outliers/anomalies is being evaluated, the *classifier* is evaluated on how well it distinguishes anomalies/outliers from normal data points, reflected in its *Specificity* metric. The overall anomaly detection performance of the *classifier* in a skewed dataset is measured with *Balanced Accuracy* [Mow05] defined as

$$\text{Balanced Accuracy} = \frac{\text{Specificity} + \text{Sensitivity}}{2}$$

The higher the *Balanced Accuracy*, the better the *classifier* performs in distinguishing normal and outliers/anomalies.

4.4.3 Temporal Characteristic Modeling

For each sub-process, transition intervals of all the transition of its *transition profile* are modeled with OC-SVM, IF and LOF (section 4.4.1.1-section 4.4.1.3). The classifiers for transitions with these methods are implemented in a *scikit-learn python*-based library [Ped11]. Each method class outputs -1 when an outlier/anomaly is detected, otherwise $+1$ for *inlier*/normal data point. Following these observations, for a given sub-process Z and its transition XY, where X payload byte transits to Y payload byte, an interval instance w is classified with Z's interval profile constituent classifier Γ_{XY} as follows:

$$\Gamma_{XY}(w) = \text{sign}(\Gamma_{XY}^{LOF}(w) + \Gamma_{XY}^{OCSVM}(w) + \Gamma_{XY}^{IF}(w)) \tag{4.1}$$

where, Γ_{XY}^{LOF}, Γ_{XY}^{OCSVM} and Γ_{XY}^{IF} are XY transition interval classifier with LOF, OC-SVM and IF, respectively. The $\Gamma_{XY}(w)$ value -1 classifies the interval as an anomaly/outlier, otherwise normal.

Hyperparameter tuning. For each anomaly/outlier detection methodology, a grid search with cross-validation scheme is performed to get the best parameters of the methods for the optimal performance score. The score function chosen for the grid search is *Sensitivity/Recall*. Grid search exhaustively evaluates the grid of parameters provided to get the best score. In particular, for LOF, the value for number of neighbours (k) is tuned, as choosing a bigger neighbourhood could miss the local outliers. For OC-SVM, the *kernel function* ($\phi(.)$) appropriate for the given data set is tuned. For all the methods, the upper bound on number of allowed misclassified training data points is additionally tuned.

Modeling with tuned hyperparameters. The tuned hyperparameters for each anomaly/outlier detection methodology are used to model the transition intervals of a sub-process. In table 4.2, the performance metric on the training data by individual method for sub-process *PLC-1~Turntable-Motor* is presented. In certain cases, the recall is less than 100%, which results from either tuned upper bound on allowed misclassification or these misclassified data points lie in the region away from rest of the data points that methods couldn't model.

Table 4.2: Performance of temporal modeling methods for sub-process *PLC-1~Turntable-Motor*.

Transitions	**Sensitivity Score**		
	OneClass SVM	**Isolation Forest**	**Local Outlier Factor**
FG	0.98	0.98	1.00
GH	0.98	0.98	1.00
HI	1.00	0.98	1.00
IJ	0.98	0.98	1.00
JK	1.00	0.98	1.00
KL	0.98	0.98	1.00
LM	1.00	0.98	1.00
MN	0.81	0.98	1.00
NF	0.98	0.98	1.00

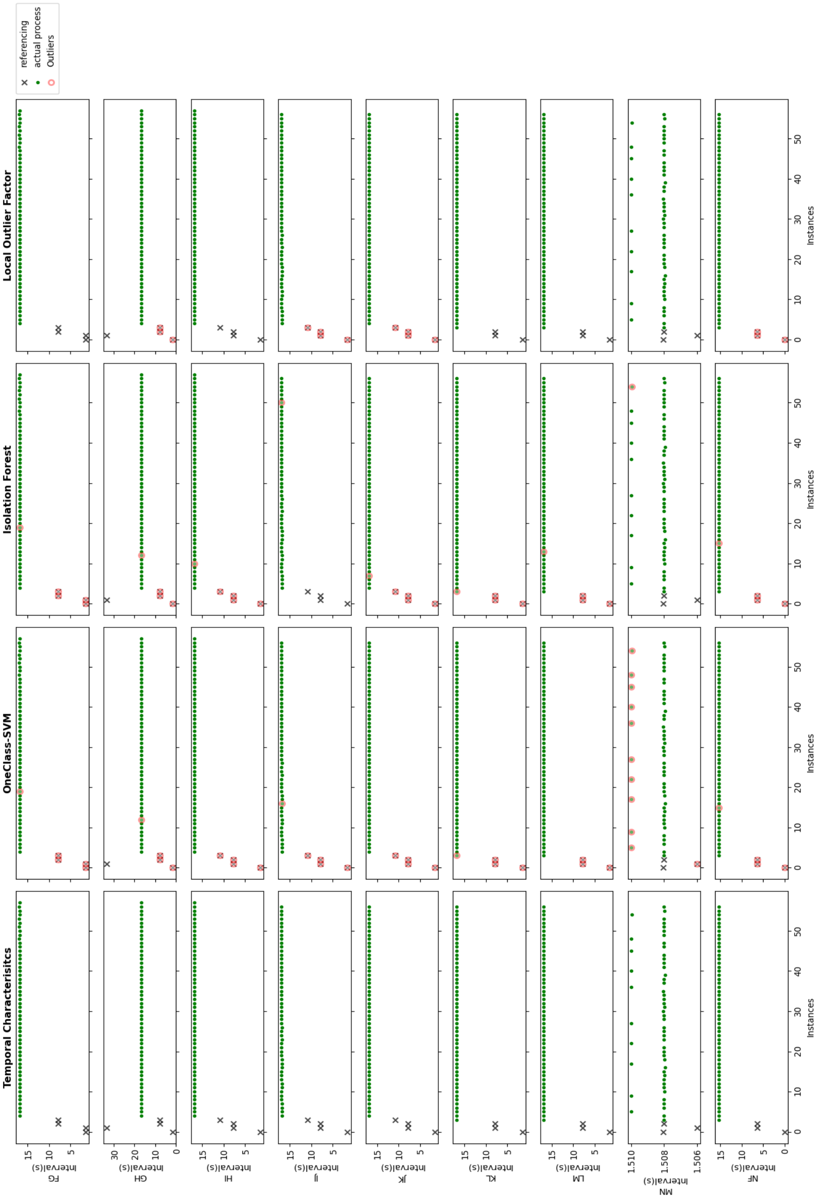

Figure 4.13: *Temporal* characteristic modeling of transitions in sub-process *PLC-1~Turntable-Motor.*

In fig. 4.13, the performance of the individual methods in detecting outliers/anomalies in the training data is shown. When the data points of '*referencing*' phase are different from '*actual process*' points, they are classified as "Outliers" (circled red). It is also evident from this figure that combining results from these methods is beneficial, with rare exceptions.

Interval profile. For each sub-process, its *interval profile* is the set of classifiers for intervals of transitions from its *transition profile*, each defined as eq. (4.1). As an example, for sub-process *PLC-1~Turntable-Motor*, its *interval profile* is $\{\Gamma_{FG}, \Gamma_{GH}, \Gamma_{HI}, \Gamma_{IJ}, \Gamma_{JK}, \Gamma_{KL}, \Gamma_{LM}, \Gamma_{MN}, \Gamma_{NF}\}$. In fig. 4.14, these are employed on the encoded traffic and classify a transition interval as '*valid/invalid interval*'. The red lines at the beginning reflects the '*referencing*' phase where certain transition have *invalid* intervals as compared to '*actual process*'. This difference in the transition intervals can be observed in the figure too. There are intervals misclassified amounting to 2/3 of constituting anomaly/outlier detection methods misclassifying these intervals.

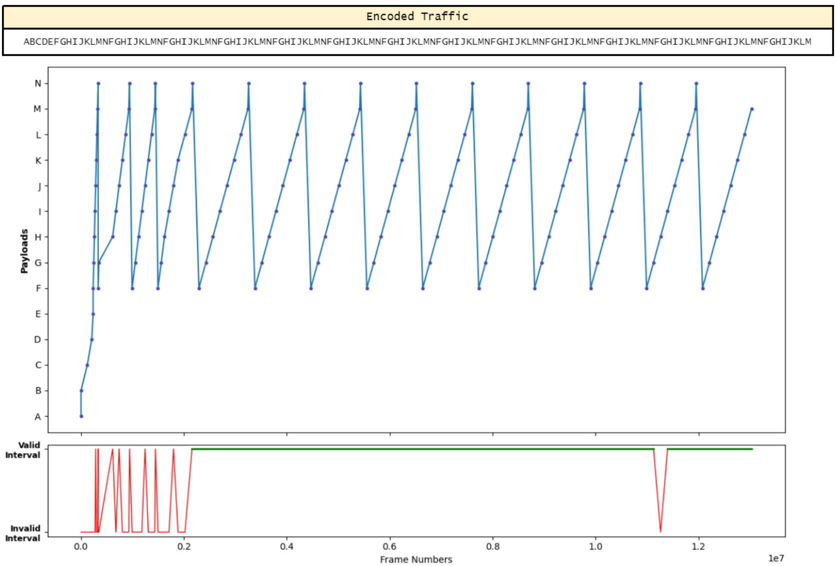

Figure 4.14: Classification of transition intervals as *Valid* or *Invalid* using *Interval Profile* of sub-process *PLC-1~Turntable-Motor*.

The *interval profile* is also applied over the transition intervals of process and *valid/invalid* transition intervals are detected, as shown in fig. 4.15. The red line at the beginning corresponds to the '*referencing*' phase of the demonstrator and only partial encoded traffic from the beginning is shown for brevity.

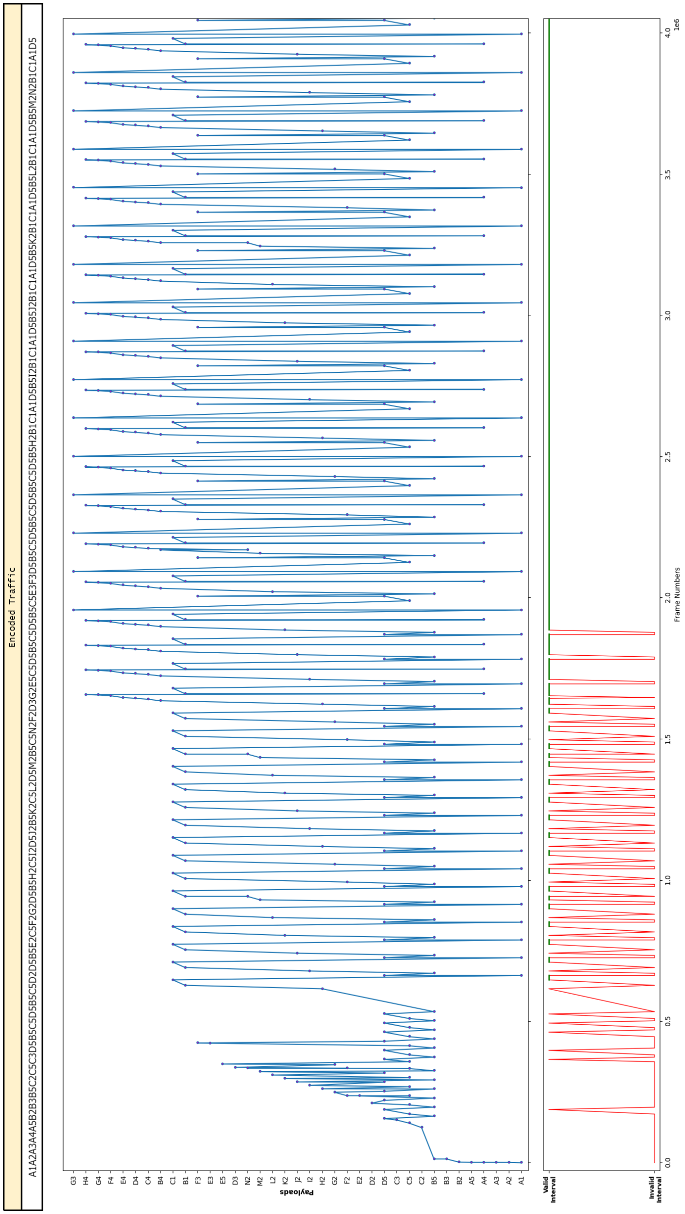

Figure 4.15: Classification of transition intervals as *Valid* or *Invalid* using *Interval Profile* of the *Festo Demonstrator*'s process.

4.5 Spatio-temporal Characteristic Modeling with Payload Byte Profiling

The *spatial* and *temporal* characteristics of an industrial process and its constituent sub-processes are captured in the *transition profile* and *interval profile* of their *Payload Byte Profiles.* For *spatio-temporal* characteristic modeling, their *transition* and *interval profiles* are combined to score a payload byte transition. For a sub-process *Z*, a transition instance t_{XY} of *Z*'s transition XY with d *seconds* transition interval, where X payload byte transits to Y payload byte in *Z*, has the *profile score* Θ^Z_{XY} as follows:

$$\Theta^Z_{XY}(t_{XY}, d) = \text{sign}(\Lambda_Z(t_{XY}) + \Gamma^Z_{XY}(d)) \tag{4.2}$$

$$\text{such that, } \Lambda_Z(t_{XY}) = \begin{cases} +1 & \text{if valid transition} \\ -1 & \text{otherwise} \end{cases}$$

$$\Gamma^Z_{XY}(d) = \begin{cases} +1 & \text{if valid interval} \\ -1 & \text{otherwise} \end{cases}$$

where Λ_Z is *Z*'s *transition profile*, and Γ^Z_{XY} is the constituent classifier for transition XY in *Z*'s *interval profile.* The *profile score* of -1 marks the transition instance *invalid* or *anomalous*, otherwise *normal.*

A transition instance which follows the order with incorrect transition duration would be flagged *valid* by the *transition profile.* Combining *interval profile* for the overall decision would correctly classify it as an *anomaly.* The advantage of such a combination can be observed in fig. 4.16, where transitions of sub-process *PLC-1~Turntable-Motor* are evaluated for *spatio-temporal* characteristics using eq. (4.2). The red lines at the beginning correspond to '*referencing*' phase and it can be observed how *interval profile* is correcting *transition profile*'s incorrect classification.

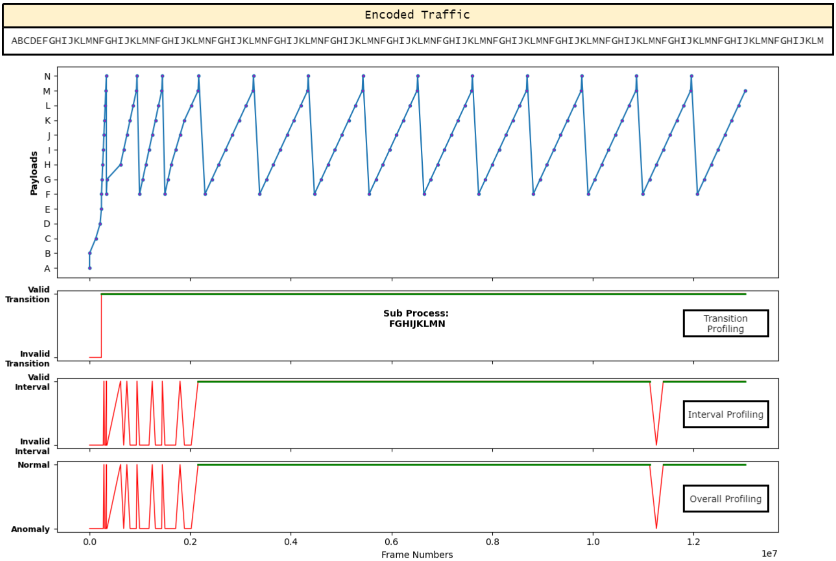

Figure 4.16: Classification of transition as *Normal* or *Anomaly* using *Transition Profile* and *Interval Profile* of sub-process *PLC-1~Turntable-Motor*.

The *spatio-temporal* characteristic modeling with PBP is also applied to the overall process and every transition is classified as *normal*/*anomaly*, as shown in fig. 4.17. The red line at the beginning corresponds to the '*referencing*' phase of the demonstrator and only partial encoded traffic from the beginning is shown for brevity.

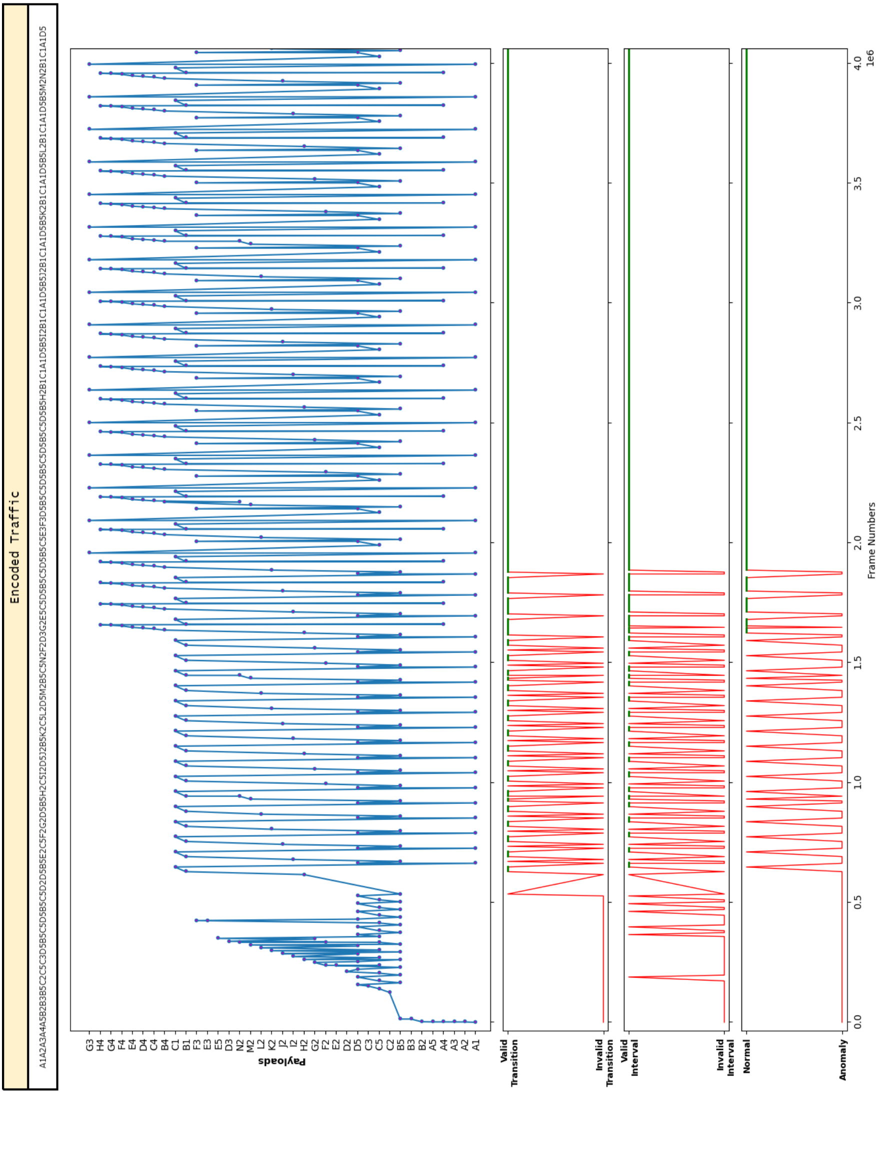

Figure 4.17: Classification of transition as *Normal* or *Anomaly* using *Transition Profile* and *Interval Profile* of the *Festo Demonstrator*'s process.

4.6 Discussion and Summary

4.6.1 Production Cycle Detection of *Festo Demonstrator*

The *production cycle* of an industrial process is defined by the deterministic and periodic execution of process parameters in constituent sub-processes. In every *production cycle*, a product is either newly produced or processed for enhancements as is the case with the *Festo Demonstrator* (refer chapter 2). Detecting the *production cycle* from the network traffic provides information that could be used for quantifying process throughput or optimization of the process utilizing timing information [Bun20].

In the PBP framework, the *production cycle* of an industrial process is its *transition profile* extracted from its encoded traffic. The framework is employed on the *Festo Demonstrator*'s process to encode the traffic (refer fig. 4.5) and extract its *transition profile*:

> C4D4E4F4G4H4A4D7B1C7C1A1F6G3E6C5D5F3B5F2B4C4D4E4
> F4G4H4A4D7B1C7C1A1F6G3E6C5D5F3B5G2B4C4D4E4F4G4H4
> A4D7B1C7C1A1F6G3E6C5D5F3B5H2B4C4D4E4F4G4H4A4D7B1
> C7C1A1F6G3E6C5D5F3B5I2B4C4D4E4F4G4H4A4D7B1C7C1A1
> F6G3E6C5D5F3B5J2B4C4D4E4F4G4H4A4D7B1C7C1A1F6G3E6
> C5D5F3B5K2B4C4D4E4F4G4H4A4D7B1C7C1A1F6G3E6C5D5F3
> B5L2B4C4D4E4F4G4H4A4D7B1C7C1A1F6G3E6C5D5F3B5M2B4
> N2C4

For interpretation of the extracted process' transition profile as its production cycle, the foundations of *PROFINET* operations from chapter 3 are reiterated and utilized. Within output data exchange, every process payload byte contains process parameters sent from the Controller to the Device. Using the engineering tool, the process payload bytes are mapped to process events engineered during System Engineering *PROFINET* operation (refer section 3.3).

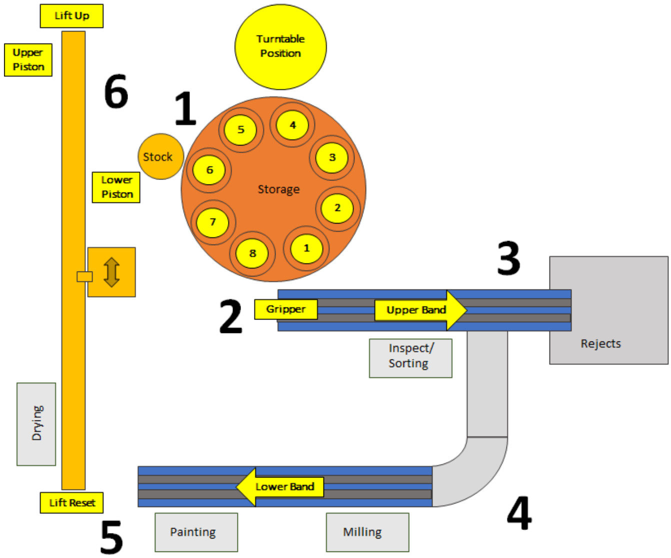

Figure 4.18: Setup of the *Festo Demonstrator* annotated with device's auxiliary information.

Furthermore, in fig. 4.18, the *Festo Demonstrator*'s devices and auxiliary process event information are highlighted in *yellow* colour. For *Turntable*, the different positions on the storage (`1-8`) are marked. The direction of *Upper Band* and *Lower Band*, and *Lift*'s position at the bottom (`LiftReset`) and the top (`LiftUp`) are shown. The associated information of *Gripper*, *Upper Piston* and *Lower Piston* are not shown for brevity. The complete mapping of encoded process payload bytes to corresponding process event instructions is summarized in table 4.3.

Table 4.3: Summary of the *Festo Demonstrator*'s payload bytes mapped to engineered process events.

Sub-Process	Alphabet-Encoded Payload	Engineered Process Event
PLC-2~BK-2	A1	`BandReset`
	B1	`MoveUpperBand_CloseBarrier`
	C1	`MoveLowerBand`
PLC-1~Turntable-Motor	F2	`TurntablePosition01`
	G2	`TurntablePosition02`
	H2	`TurntablePosition03`
	I2	`TurntablePosition04`
	J2	`TurntablePosition05`
	K2	`TurntablePosition06`
	L2	`TurntablePosition07`
	M2	`TurntablePosition08`
	N2	`TurntablePositionReset`
PLC-3~Lift-Motor	F3	`LiftPositionReset`
	G3	`LiftPositionUp`
PLC-1~BK-1	A4	`GripperReset`
	B4	`GripperPush`
	C4	`GripperDown`
	D4	`GripperClose_Up`
	E4	`GripperClose_Pull`
	F4	`GripperClose_Down`
	G4	`GripperClose`
	H4	`GripperUp`
PLC-3~BK-3	B5	`LowerPistonPull`
	C5	`LowerPistonPull_UpperPistonPush`
	D5	`LowerPistonPush`
PLC-1~PLC-3	E6	`LiftStay`
	F6	`MoveLift`
PLC-1~PLC-2	C7	`BandStay`
	D7	`MoveBand`

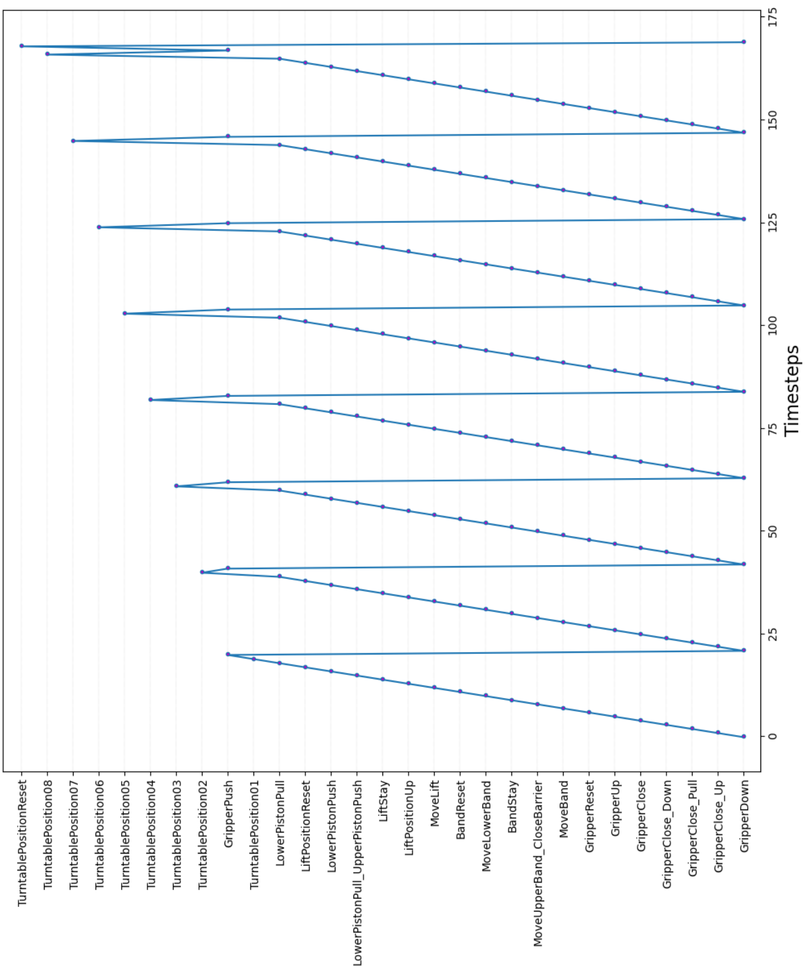

Figure 4.19: The *Festo Demonstrator*'s *production cycle* detected with PBP.

Using the table 4.3 on the extracted process' *transition profile*, as its *production cycle*, the order of process events at every timestep is represented in fig. 4.19. The detected *production cycle* has been confirmed as correct by the *Engineer* of the *Festo Demonstrator*'s automation project. It can be visually verified from fig. 4.19, the order of instructions sent from the Controller are same as process description in chapter 2:

1 *Gripper* to pick up a *piece* from *Turntable* (`GripperDown→GripperClose_Up`), and put down on the *Upper Band* (`GripperClose_Pull→GripperClose_Down →GripperClose→GripperUp→GripperReset`)

2 *Upper Band* to move while closing the barrier to slide down the *piece* to *Lower Band* (`MoveUpperBand_CloseBarrier`). Then, to move *Lower Band* to put the *piece* on *Lift*'s platform (`MoveLowerBand→BandReset`).

3 *Lift* to move the *piece* to the top followed by *Upper Piston* to push it into the *Stock* (`LiftPositionUp→LowerPistonPull_UpperPistonPush`).

4 *Lower Piston* to push the *piece* onto the *Turntable* out of the *Stock* (`LowerPistonPush`), and *Lift* to drive back to bottom (`LiftPositionReset→LowerPistonPull`).

5 *Turntable* to rotate to new position (`TurntablePosition01`), and *Gripper* to move out onto the *Turntable* (`GripperPush`).

6 1-5 are repeated for all of the *Turntable*'s positions (`TurntablePosition01 ,.., TurntablePosition08`), and restarts the *production cycle* with resetting the *Turntable*'s position counter (`TurntablePositionReset`).

4.6.2 Anomaly Detection with Payload Byte Profiling

An *adversary*, with the knowledge of process parameters and employed industrial components, could either change the process parameters or the component configurations. For the *Festo Demonstrator* use case, a *'Force Attack'* changes the process parameters of the *Lower Band* to move it in reverse direction chapter 2. The additional attack scenario, *'Rename Attack'*, changes the *logical* name of the *Turntable-Motor* to make it inaccessible to other device within the process. Detection of such attacks through process data analysis is evaluated with the PBP framework, and its qualitative comparison to detection performance of a commercial NIDS is made.

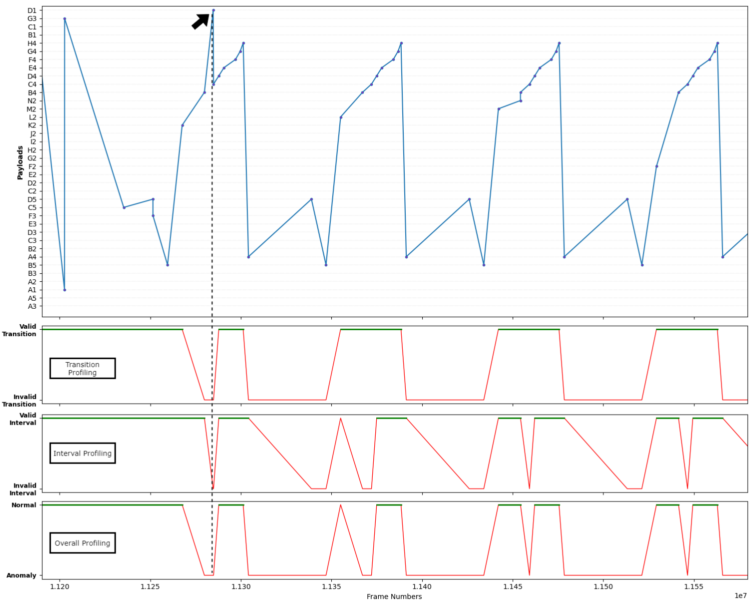

Figure 4.20: The demonstration of anomaly detection using PBP.

The network traffic from the *Festo Demonstrator* with simulated attacks is captured and used for the analysis. Through PBP, the *'Force Attack'* was easily detected as it did not suffice the already learnt *transition profile* for the *Demonstrator*'s process. In fig. 4.20, the anomalous instance and corresponding process payload byte is pointed with an arrow. The horizontal dashed line shows the *invalid* transition, interval and overall *anomaly* classification of this event in the traffic.

Table 4.4: The anomaly detection performance of Payload Bytes Profiling (PBP).

		Traffic		
	Overall	**Normal**	**Force Attack**	**Rename Attack**
Sensitivity	0.97	0.97	0.85	1.00
Specificity	0.77	0.00	**1.00**	0.00
Balanced Accuracy	0.63	0.49	0.93	0.50

In table 4.4, the overall performance for anomaly detection on the collected data is summarized. With the framework, 97% of the normal packets were correctly recalled and 77% of the anomalous instances were detected, from table's first column. To evaluate the performance at finer granularity, the *Specificity* and *Sensitivity* were calculated for normal and two attack traffic packets. The *'Force Attack'* as already mentioned is correctly classified as anomalous, however, *'Rename Attack'* packets were not detected at all. In the *'Rename Attack'*, only the device property is being manipulated not the process, hence, no change in process bytes exchanged in *output* direction.

The collected traffic is analysed through a commercial NIDS which successfully detected *'Rename Attack'*, but couldn't detect the change in the process payload bytes sent in *'Force Attack'*. The NIDS is a *rule*-based system which utilizes *protocol*-based and *traffic mining* information to detect anomalies. Its anomaly detection capability on *process* information could be enhanced with integration of the PBP framework.

Through successful detection of *'Force Attack'*, the PBP framework satisfies the BSI's "*Category D - unusual changes in process data*" requirement, outlined in section 1.1.1, for detecting anomalies in an industrial process.

4.6.3 Review

For a string of length n and alphabet $\mathcal{A}$, the space complexity of its suffix tree is $\mathcal{O}(n|\mathcal{A}|\log n)$. In 1993, Udi Manber and Gene Myers [Man93], introduced Suffix Array as a data structure to store suffixes of a string in a lexicographically increasing order. It has $\mathcal{O}(n \log n)$ space complexity and is independent of alphabet size. The longest repeated string is found by comparing neighbouring suffixes for 'Longest Common Prefix (LCP)' and the LCP with largest size is returned. This has overall runtime complexity of $\mathcal{O}(n \log n)$ for a string of length n [Sun09]. When there is no restriction on runtime and string is quite large in size, a suffix array could be substituted for a suffix tree in section 4.3.

One-Class Support Vector Machines are better at detecting *global outliers* in data than *local outliers*. This gap is filled by Local Outlier Factor and Isolation Forest, and together all three methods cover the whole range of *outliers*. However, the quality of data used in their training is the decisive factor on their *anomaly*/*outlier* detection performance. If it contains noise then the methods would learn *false positives* instances as *normal*. In section 4.4.3, the data is assumed to be devoid of *unwanted* noise. In industrial networks, an application-specific acceptable *jitter*/noise in transition intervals is allowed, which is included for the analysis as part of normal operations.

The PBP framework has been conceptualized with *PROFINET* as an instance of industrial network protocol. Nevertheless, it is extensible to any industrial protocol in discrete process as the requirements (refer section 4.2.1) and *spatio-temporal* characteristics remain same. For an industrial protocol $\mathcal{E}$, a protocol dissector is required that extracts payload bytes containing process parameters from process data exchanges utilizing $\mathcal{E}$. The rest of the procedure to extract the *interval* and *transition profiles* of underlying process follows as described in section 4.3 and section 4.4.

Bunte et al. [Bun20] presented an automaton based approach to detect *production cycle* in production plants. The employed automaton, Online Timed Automaton Learning Algorithm (OTALA), keeps track of number of passes through different sets of state transitions. Each state consists of all discreet value the system, and transitions are triggered by change in discrete signals. The PBP framework also considers change in signals (process parameters), captured in alphabet-encode process payload bytes. It detects the *production cycle* in linear time and doesn't keep a history of different transition matrices as the OTALA-based algorithm. Both the methods operate on similar information i.e. change in signals with different approaches to arrive at same goal of *production cycle* detection.

The *'Force Attack'* on the *Festo Demonstrator* sends new process parameters in the output data exchange, triggering the change in *spatial* characteristics of sub-process *PLC-2~BK-2*, hence, detected easily using its *transition profile*. Korkmaz et al. [Kor17] presented time delay injection attack in ICS, where an attacker injects extra extra time delays in the underlying process to cause anomalous operational regime, leading to system crash. The time delays are introduced incrementally at lower rate to avoid detection by system devices. This gradual increase in the time delays would be detected as invalid transition intervals using the *interval profile* of the underlying process. Implementing such attacks need precision in intercepting the traffic at right moment and injecting right amount of time delays. Collection of these configurations on the *Festo Demonstrator* is ongoing and a similar attack will be added for evaluating PBP framework's capability in its detection.

4.6.4 Conclusion

In this chapter, the extraction of *spatio-temporal* characteristics of an industrial process from its network traffic for process behaviour analysis in a self-learning setup is addressed. A framework, Payload Bytes Profiling (PBP), capturing the *spatial* and *temporal* characteristics of an industrial process, and its constituent sub-processes, through alphabet-encoding of process payload bytes is introduced. The *transition profile* representing the *spatial* characteristic is extracted with RegEx-couple Suffix Tree based algorithm in section 4.3. The temporal characteristic is the interval between transitions of transition profile, and is modeled as set of *classifiers*, one for each transition pair, in section 4.4. The *classifier* inherently constitutes of density-based Local Outlier

Factor, boundary-based One-Class Support Vector Machines and tree-based Isolation Forest *anomaly/outlier* detection classifiers in combination for detecting *valid/invalid* transition interval.

The PBP framework in section 4.6.1 has been used to detect the *production cycle* of the *Festo Demonstrator*, which has been verified by the *Engineer* of the *Demonstrator*'s automation project. In section 4.6.2, the PBP framework's role as a process analysis-based IDS in successfully detecting a process-targeted attack, *'Force Attack'*, on the *Demonstrator* is outlined. Thus satisfying the BSI's "*Category D - unusual changes in process data*" requirement of an anomaly detection system to detect anomalies in an industrial process, as outlined in section 1.1.1. In conclusion, the PBP framework is extensible to any industrial protocol in discrete process as the requirements and *spatio-temporal* characteristics remain same.

In the next chapter, the research problems associated with analysing the *PNIO* frames for anomaly detection of industrial process behaviour through *Graph Representation Learning* is being addressed.

5 Graph-based Self-learning Process Behaviour from Network Traffic

A *Graph* is an ubiquitous representation of systems where the interactions between the constituent objects reflect functionality of the system. The objects are represented as *nodes/vertices* and the directional or mutual interactions/relations are represented by the *directed* or *undirected edges* between components. Social networks interactions [Qiu18], telecommunication interactions [Cor01], recommender systems [Wu19], molecular interactions in chemistry [Gil17], grammatical relations between words of a sentence [Mar17], etc. are some of the graphically represented systems.

Anomaly detection on graph data has been an active research area [Ako15]. Especially, graph-based anomaly detection finds its application in Cybersecurity for spam and malware detection [Cas07, Ben05] in Web network, socware (malware in social networks) detection in Social networks [Rah12], and cyber-attacks and intrusion detection in computer networks [Xia20, Ili11].

An industrial communication network can be represented as a *graph*, where industrial components and networking assets constitute nodes and the communication relationships between them are represented as edges of the graph. The communication relationship modeled on the edges represent the nature of the graph as *static* or *dynamic*. The connectivity information embedded in the graph are exploited for network optimization [Alm19], network planning [Zhu21] and deployment of network security countermeasures.

The process data communication relationships between industrial components could be modeled in a graph and used for process data analysis to detect anomalies in the industrial process. The network topology remains *static* however the values on the edges are *dynamic*. The structural information and temporally changing edge information offers insights into *spatio-temporal* characteristics of industrial networks.

In this chapter, the challenge of representing the *spatio-temporal* characteristics of an industrial process as a *Graph* and self-learning the process behaviour for anomaly detection is addressed. The research problems concerning the representation and learning are formulated in section 5.1 followed with the foundational information on *Graph Representation Learning* in section 5.2. The proposed solutions for the representation and learning of an Graph-represented industrial process are described in section 5.3 - section 5.5. The chapter is reviewed and concluded in section 5.6.

5.1 Research Problem

Threat actors target the *spatial* and *temporal* characteristics (the periodic and deterministic requirements) of an industrial process through violating *integrity* and *availability* requirements of industrial communication [Rub17]. The threat behaviours in the traffic could be detected efficiently through Graph-based analysis *only* when the industrial network's *Graph* representation aptly *represents* the *spatio-temporal* characteristics of its communication. After an appropriate *spatio-temporal* characteristics representation on a graph is available, a Machine Learning (ML) model can *learn* the *normal* process behaviours and detect the deviations as *anomalies*.

In a *self-learning* process behaviour analysis of industrial networks, the *spatio-temporal* characteristics are extracted from the industrial traffic and represented as a graph. A ML model learns the normal process behaviour from the normal operations of industrial process, and the model is utilized for anomaly detection in process data exchange. Graph Neural Networks (GNN) [Mer05, Sca08] are the ML models exclusively developed for graph data analysis. In recent years, GNN have gained attention for analysis of graph-structured data and have shown to perform better in graph classification, node classification, edge classification and link prediction tasks [Zho20b]. Its applications in Cybersecurity for botnet detection [Zho20a], malware detection [Bus21] and network intrusion detection [Puj21] have been reported.

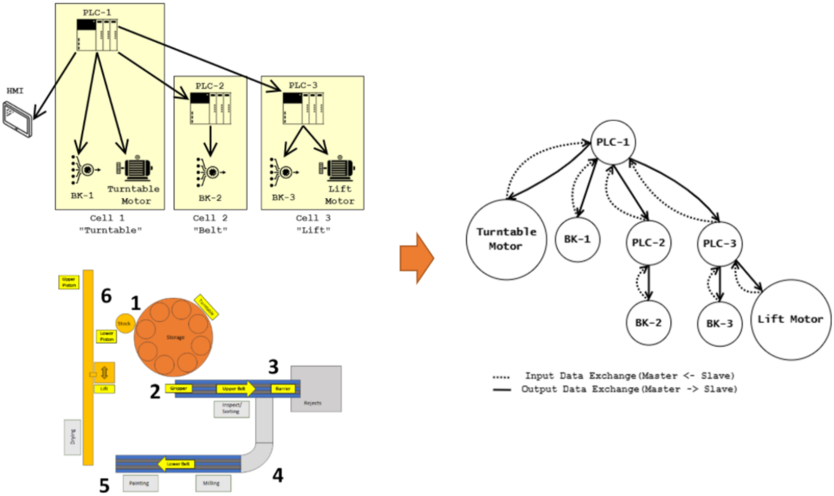

Figure 5.1: Graph representation of the *Festo Demonstrator*'s industrial network communication.

In the presented work, for a given *PROFINET*-based industrial system (the *Festo Demonstrator*, refer section 2.5), its process data exchange is represented as a graph, as shown in fig. 5.1. The edges represent *Connections* between industrial components and changing edge values represent process data exchange. A *spatio-temporal* graph representation is required to capture the *spatio-temporal* characteristics extracted from *PROFINET* traffic. Feasibility of GNN to model process behaviour based on the *spatio-temporal* graph representation for anomaly detection needs an evaluation. The research problem of *self-learning* the process behaviour in Graph represented data is formulated as follows:

1 How to model *spatio-temporal* characteristic of industrial communication in Graph representation? [*Addressed in section 5.4*]

2 How to learn Graph-represented process behaviour? [*Addressed in section 5.5*]

3 How to classify Graph-represented process data as *normal* or *anomalous*? [*Addressed in section 5.5*]

5.2 Foundations

Formally, a graph **G**=($\boldsymbol{\mathcal{V}}$, $\boldsymbol{\mathcal{E}}$) defines set of nodes $\boldsymbol{\mathcal{V}}$ and interactions/relations between nodes as set of edges $\boldsymbol{\mathcal{E}}$. An edge $e_{ij} \in \mathcal{E}$ is defined as directed/undirected link between nodes v_i and v_j, $v_i, v_j \in \mathcal{E}$. In case of directed relations, $e_{ij} \neq e_{ji}$. The structural information of a graph or its topology is represented by an adjacency matrix $\mathcal{A}$. Each element of the matrix $a_{ij} \in \mathcal{A}$ represents the edge information between nodes v_i and v_j i.e. $a_{ij} = 1$ if $e_{ij} \in \mathcal{E}$, otherwise $a_{ij} = 0$. If there are no self-loops on the nodes, the diagonal elements are 0, and for undirected graphs $\mathcal{A}$ is diagonally symmetrical. A *complete graph* contains an unique edge between every pair of nodes. If the graph is not a *complete graph*, the $\mathcal{A}$ is a sparse matrix and utilizing it for graph analysis task is computationally expensive. To circumvent the computational overload, the edge connectivity is represented as *coordinate list* (COO) of node indices tuples, such that tuple (i,j) represents edge between nodes v_i and v_j.

In *homogeneous* graphs, the nodes and edges have same types, whereas nodes and edges have different types in *heterogeneous* graphs. Social networks are homogeneous graphs where nodes are persons and edges represent relationship between them. In heterogeneous biomedical graphs, the nodes could be either proteins, drugs or diseases and edges between protein-drug and drug-disease represent different information such as inhibitor and treatment relations, respectively.

Any domain's data represented as graphs are analysed with ML methods to extract problem-driven solutions to its underlying problem use cases. The foremost requirement for any ML pipeline is defining features that captures the information embedded in graph data. Node-level features of graphs are represented by matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times m}$, where each node is represented as m-dimensional real-valued vector. When edge-level features are available, the elements of adjacency matrix $\mathcal{A}$ are represented as $a_{ij} \in \mathbb{R}^k$ for k-dimensional real-valued vector.

The ML tasks in the context of graph analysis can be summarized as follows:

1 Node-level tasks either classify nodes into categories (*node classification*), predict real-value of nodes (*node regression*) or group similar nodes together (*node clustering*). In a citation graph, classification of node-represented documents to a topic is an example of node classification [Kip16].

2 Edge-level tasks aim to either classify edge types (*edge classification*) or predict a relation/link between nodes (*link prediction*). Predicting a drug's side-effects in biomedical graphs is an example of link prediction [Zit18].

3 Graph-level tasks consist of predicting real-valued properties of a graph (*graph regression*), classify graphs into categories (*graph classification*) and grouping similar graphs together (*graph clustering*). Predicting toxicity of a molecule is an example of graph regression [Gil17]. For graph-level tasks, the node features (and edge features) are used in the model for analysis.

In a *transductive learning* setting, the ML model evaluates on nodes provided in training phase. For new unseen nodes, the training phase is performed again. In contrast, the models learnt in *inductive* setting evaluates new unseen nodes without the need of retraining.

The ***Traditional ML*** approaches on graphs utilize *graph statistics* and *kernel methods* for node and graph classification tasks. In particular, node degree, node centrality, clustering coefficient for node's local neighbourhood are node-level features used for node classification tasks. For graph classification tasks, graph-level features are extracted through graph kernel methods such as bag of nodes, the Weisfeiler-Lehman kernel, graphlet (subgraph structures) kernel, path-based random walk and shortest-path kernels. The node-level and graph-level features used for classification tasks do not take the relationships between nodes into consideration. For edge prediction tasks, a new measure quantifying the neighbourhood overlap between nodes have been used. For the local overlap statistics, Sorensen index, Salton index, Resource Allocation index, etc. have been proposed. When two nodes do not share a local neighbourhood but belong to same community in the graph, global overlap statistics have been proposed for such use cases. Katz index, LHN similarity and random walk method such as PageRank approaches are path-based similarity measures developed for quantifying global neighbourhood relationships. For details on the mentioned graph statistics, refer to Chapter 2 of Hamilton's *Graph Representation Learning* [Ham20].

The traditional ML approaches on graphs are limited due to hand-engineered statistics and measures which are inflexible to adapt to a learning process and expensively time-consuming design process. As an alternative to limiting hand-engineered feature extraction, the *graph representation learning* approach learns the representations embedded with structural information of a graph. As outlined in [Ham20], the graph representation learning is viewed as the framework of encoding and decoding graphs to learn low-dimensional vectors or *embeddings*. An encoder model encodes the graph's positional and local neighbourhood structural information of nodes in low-dimensional vectors. The decoder model takes the node embeddings to reconstruct node neighbourhood and other graph statistics in the original graph.

Formally, for a pair of nodes $u,v \in \mathcal{V}$, an encoder function ENC, a decoder function DEC, a similarity measure between two nodes in a graph S, $z_u,z_v \in \mathbb{R}^d$ as d-dimensional node embeddings, the encoder-decoder framework is summarized as follows:

$$ENC : \mathcal{V} \rightarrow \mathbb{R}^d$$
$$DEC(ENC(u), ENC(v)) = DEC(z_u,z_v) \approx S[u,v]$$

The reconstruction objective is to minimize the reconstruction loss $\mathcal{L}$ over a set of node pairs $\mathcal{D}$:

$$\mathcal{L} = \sum_{u,v \in \mathcal{D}} l(DEC(z_u, z_v), S[u,v])$$

where l is the reconstruction loss function, usually mean-squared error or cross entropy loss.

The ***Shallow Embedding*** approach defines the encoder function as a lookup of a node based on its *ID* in a node embedding matrix $\boldsymbol{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$. The well known shallow embedding methods are motivated by matrix factorization and random walks. In matrix factorization, the decoder reconstructs the entries into the adjacency matrix of a graph from low-dimensional node embeddings with approximation on node-node similarity matrix S. Graph Factorization approaches, GraRep and HOPE are examples of shallow node embedding algorithms using matrix factorization. DeepWalk, node2vec and large-scale information network embeddings are random walk based shallow node embedding algorithms. The node embeddings are optimized such that the two nodes co-occurring on short random walks over graph have similar embeddings. There are drawbacks to shallow embedding approaches: *(a)* no parameters are shared between nodes in the encoder, hence, the number of parameters grow linearly as number of nodes increases (computationally inefficient), *(b)* the node features are not utilized in the encoder, and *(c)* transductive learning setting i.e. doesn't generalize to unseen nodes.

5.3 Graph Neural Networks

Graph Neural Networks (GNN) are a neural network framework for graph data with encoders defined over the graph structure and its attributes. The motivation of utilizing neural networks for graph data is found in the works of Bruna et al. [Bru13] to generalize convolutions to non-Euclidean data, Dai et al. [Dai16] defining a differentiable variant of belief propagation and Hamilton et al. [Ham17a] outlining graph isomorphism tests. At its core, a GNN employs a form of *neural message passing*, where for a given graph $\mathbf{G}$=($\boldsymbol{\mathcal{V}}$, $\boldsymbol{\mathcal{E}}$) and its node features $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$, the information as vector messages are exchanged between nodes and updated through neural networks to generate node embeddings $z_u, \forall u \in \mathcal{V}$ [Gil17]. In message-passing iteration of GNN, a *hidden embedding* h_u for each node $u \in \mathcal{V}$ is updated with information aggregated from u's graph neighbourhood $\mathcal{N}(u)$, as shown in fig. 5.2. In fig. 5.2, the message-passing computations of GNN over the graph for its full neighbourhood is shown on the right.

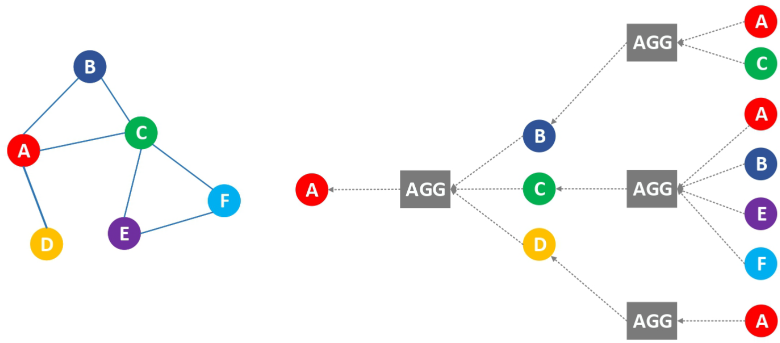

Figure 5.2: The demonstration of GNN computation [Ham17b].

The message-passing update process [Ham20] can be summarized as:

$$\begin{aligned} h_u^{(k+1)} &= UPDATE^{(k)}(h_u^{(k)}, AGGREGATE^{(k)}(\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\})) \\ &= UPDATE^{(k)}(h_u^{(k)}, m_{\mathcal{N}(u)}^{(k)}) \end{aligned} \tag{5.3}$$

where, $UPDATE$ and $AGGREGATE$ are differentiable functions and $m_{\mathcal{N}(u)}$ is the "*message*" containing aggregated information from node u's graph neighbourhood $\mathcal{N}(u)$. The superscripts represent embeddings and functions at different iterations of message passing, or different layers of GNN.

5.3.1 Basic Intuition

At each iteration/layer k of GNN, every node aggregates embeddings of its local neighbouring nodes in the graph and updates its own embeddings. As the iterations progresses, the information of the further reaches in the graph are aggregated into node's embeddings. The initial embedding for a node u is its input features i.e. at $k = 0, h_u = x_u, \forall u \in \mathcal{V}$. At $k = 1$, node u's 1-hop neighbourhood is aggregated in $h_u^{(1)}$ i.e. all the nodes that are immediate neighbours with graph path length 1. At $k = 2$, $h_u^{(2)}$ contains 2-hop neighbourhood information, and so on. After K iterations of message-passing and updates, the final layer outputs the embeddings of each node, $z_u = h_u^{(K)}, \forall u \in \mathcal{V}$. It summarizes *structural information* such as node degrees and *feature-based information* of nodes in its K-hop neighbourhood. The local neighbourhood information aggregation is analogous to convolutional operation in Convolutional Neural Network, where feature information is aggregated over local graph neighbourhood instead of spatially-defined patches in an image [Dai21].

5.3.2 Basic GNN

The abstract definition of message-passing in a GNN described so far can be translated to a general instantiations in two classes as follows:

(a) *isotropic GNN*, where every neighbouring edge is treated equally in the update equation:

$$h_u^{(k)} = \sigma(W_{self}\, h_u^{(k-1)} + W_{neigh} \sum_{v \in \mathcal{N}(u)} h_v^{(k-1)})$$

(b) *anisotropic GNN*, where every neighbouring edge is weighted different:

$$h_u^{(k)} = \sigma(W_{self}\, h_u^{(k-1)} + W_{neigh} \sum_{v \in \mathcal{N}(u)} \eta_{uv}\, h_v^{(k-1)})$$

$$\eta_{uv} = f^{(k-1)}(h_u^{(k-1)}, h_v^{(k-1)})$$

where $W_{self}, W_{neigh} \in \mathbb{R}^{d^{(k)} \times d^{(k-1)}}$ are trainable parameter matrices, σ is a non-linear element-wise activation such as *ReLU* or *tanh*, and f is a parametrized function on edge between nodes u and v.

Different GNN instantiations of *isotropic* and *anisotropic* GNN, which are evaluated for process data analysis, are outlined in section 5.5.

5.3.3 Graph Pooling

After K-iterations, the node embeddings $z_u, \forall u \in \mathcal{V}$, for a graph are computed. However, in a graph-level task, an embedding over the graph $\mathbf{G}$, z_G is required. The graph embedding can be learnt through pooling the node embeddings over final GNN layer together to calculate their sum (or mean) [Ham20]:

$$z_G = \frac{\sum_{u \in \mathcal{V}} z_u}{f_n(|\mathcal{V}|)},$$

where f_n is the normalizing function.

5.4 Spatio-temporal Graph Representation

In order to capture the *spatio-temporal* characteristics of an industrial system's network communication, Payload Bytes Profiling (PBP) framework's sequential steps from *Payload Byte Extraction* to *Transition Interval Computation* are employed, as shown in fig. 5.3.

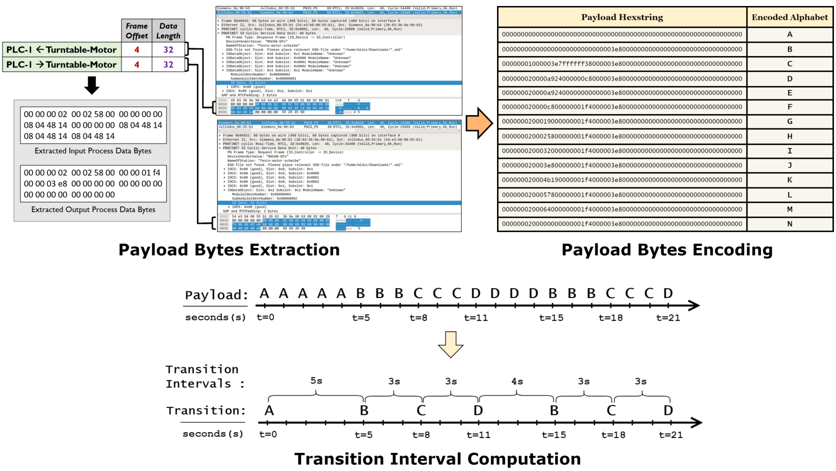

Figure 5.3: The Payload Byte Encoding steps from PBP.

The transition order of the alphabet-encoded process payload bytes corresponds to the *spatial* characteristics and the transition interval represents the *temporal* characteristics. In the current case of graph representation of industrial network communication, we encode the *spatio-temporal* characteristics of process data exchange over the edges of *Connection* between industrial components (IOController, IODevice), as shown in fig. 5.4. Additionally, the network transmission characteristics are added as the *Cycle Counter* value differences between transitioning process payload bytes. The cycle counter value in a *PNIO* frame serves the purpose of up-to-dateness determination of frame and detect duplicate frames. Based on the deterministic time interval in process data transmission, the cycle counter values are incremented in fixed iterations and reset after the cycle is complete. The difference in cycle counter values of transition process payload bytes encapsulates either transition from higher counter valued frames to lower counter frames (-ve) or vice-versa (+ve). In a nutshell, as shown in fig. 5.4, the edge features of the *Connection* between components is defined as tuple `(previous payload byte, current payload byte, transition interval, cycle counter difference)`.

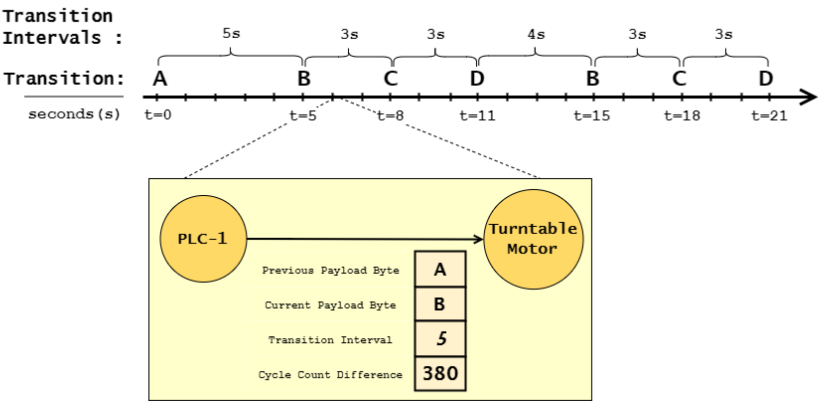

Figure 5.4: Spatio-temporal industrial network characteristics modeling on Graph.

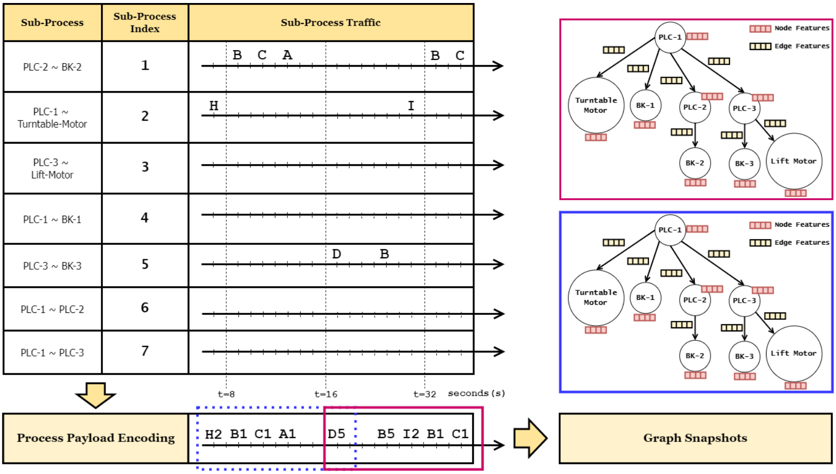

Figure 5.5: Graph Snapshots over the encoded traffic of industrial process.

In order to represent the *spatio-temporal* characteristics of the process, a *graph snapshot* is defined on a fixed time window over the traffic, shown in fig. 5.5. The *graph snapshot* contains the *homogeneous* graph representation of the industrial system's network communication, where features of each *Connection* edge are updated as per transition information of the window. The window size is defined as number of *Output Connections* of the industrial network.

Significance of encoding: Within the *graph snapshot*, the correlation of process payload bytes between *Connections* are encoded. It encodes following implicit information:

1 the relation between the current payloads of all *Connections*, and

2 the relation between *intra-Connection* transitions, along with their *temporal* transition intervals.

Especially, when the transition of payload bytes in one of the *Connection* remains persistent for long time, its effect on the behaviour of other *Connections* can be easily captured. In theory, stealthy attacks on industrial processes through manipulation of *Cycle Counters* in *PROFINET* traffic outlined by Ferrari et al. [Fer20] can be detected. The cycle counter differences encoded over the edge will have inconsistent values compared to normal behaviour.

Binary Encoding. For representing numerically the encoded process payload byte and devices, one-hot encoding is incorporated. For encoding the values of intervals and counter differences, each *float* value is *32-bit* encoded. For example, A1 and B1 alphabet-encoded transitioning payload bytes for *Connection PLC-2~BK-2* are one-hot encoded over the edges, as shown in fig. 5.6. The interval values 32.0 *seconds* and counter differences 2600 are binary encoded. The edge feature for every *Connection* in the *graph snapshot* has length 76 (6 + 6 + 32 + 32), and node feature size is 8.

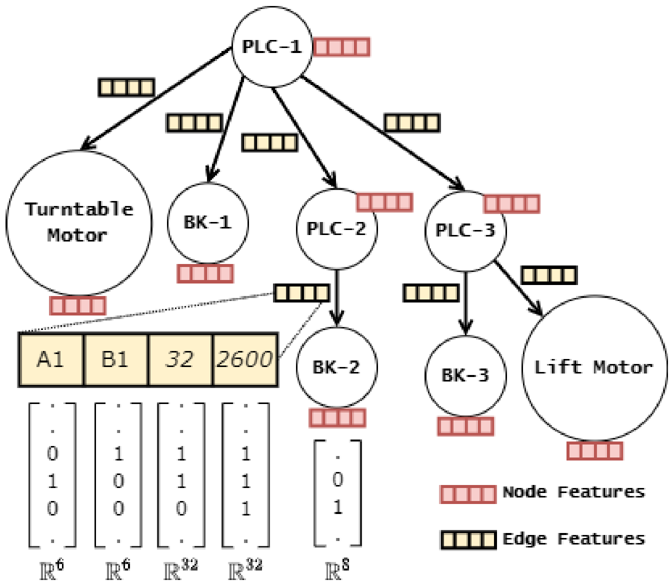

Figure 5.6: Binary Encoding of node and edge features.

5.5 Graph-based Process Data Analysis

The industrial network communication characteristics are represented as *Graphs*, where edges represent logical network connection between industrial components for process data exchange. GNN are the ML models being employed to learn the characteristics. They learn the structural information of logical network topologies and process exchange traffic characteristics encoded over the edges in edge features, as outlined before. The ML task for GNN is to classify *graph snapshots* for *normal* or *abnormal* industrial process behaviour (*graph classification*). Different GNN architectures are evaluated for their performances on learning the normal behaviour, and capability to distinguish anomalies triggered by process data targeted *'Force Attack'* (refer fig. 2.13).

5.5.1 GNN Variants

There have been multiple GNN variants developed in recent years varying in their $UPDATE$ and $AGGREGATE$ functions. In this work, *Graph Convolutional Network* (GCN) for an isotropic GNN, and *Message Passing Neural Network* (MPNN), *Gated Graph ConvNet* (GatedGCN) and *Graph Transformer* (GT) anisotropic models have been employed for proposed solution. The selection of anisotropic models is driven by their usage of edge attributes for GNN computation and implementation availability amongst the published GNN architectures.

(a) Graph Convolutional Network (GCN) is the baseline GNN model, outlined by Kipf et al. [Kip16], that employs symmetric normalization for isotropic aggregation of neighbourhood information of a node along with its own information.

$$h_u^{(k+1)} = \mathrm{ReLU}\left(W^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{h_v^{(k)}}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}} \right),$$

where W is the learnable parameter matrix, h_u,$h_v \in \mathbb{R}^d$ are node embeddings and $\mathcal{N}(u)$ is the neighbourhood of node u.

(b) Message Passing Neural Network (MPNN) is a GNN framework, outlined by Gilmer et al. [Gil17], that operates on node features $x_u, u \in \mathcal{V}$ and edge features $e_{uv}, e_{uv} \in \mathcal{E}$ of a given graph to generate node embeddings.

$$h_u^{(k+1)} = \mathrm{ReLU}(W^{(k)} h_u^{(k)} + \sum_{v \in \mathcal{N}(u)} h_v^{(k)} h_\theta(e_{uv})),$$

where, h_θ is a multilayer perceptron to embed edge features, W is the learnable parameter matrix, and h_u,$h_v \in \mathbb{R}^d$ are node embeddings.

(c) Graph Transformer (GT) outlined in Shi et al. [Shi20] adopts multi-head attention mechanism from *Transformer* architecture of Vaswani et. al. [Vas17], with inclusion of edge features for graph learning. An "*attention*" defines the weight of an edge in a node's neighbourhood as per its influence in aggregation computation.

$$\begin{aligned} h_u^{(k+1)} &= \mathrm{ReLU}(W_1^{(k)} h_u^{(k)} + \sum_{v \in \mathcal{N}(u)} \alpha_{uv}(W_2^{(k)} h_v^{(k)} + W_6^{(k)} e_{uv})) \\ \alpha_{uv} &= \mathrm{softmax}\left(\frac{(W_3^{(k)} h_u^{(k)})(W_4^{(k)} h_v^{(k)} + W_6^{(k)} e_{uv})}{\sqrt{d}}\right), \end{aligned}$$

where, W_1, W_2, W_3, W_4, W_5 and W_6 are the learnable parameter matrices, e_{uv} is the edge feature, α_{uv} is the attention-coefficient for edge e_{uv}, and d is the size of head.

(d) Gated Graph ConvNet (GatedGCN) is the GNN architecture proposed by Bresson et al. [Bre17] leveraging the vanilla graph ConvNet architecture of Sukhbaatar et al. [Suk16] and the edge gating mechanism of Marcheggiani et al. [Mar17]. Unlike other anisotropic GNN models, GatedGCN explicitly update edge features along with node features at each iteration:

$$h_u^{(k+1)} = \mathrm{ReLU}(W_1^{(k)} h_u^{(k)} + \sum_{v \in \mathcal{N}(u)} \eta_{uv} \odot W_2^{(k)} h_v^{(k)}),$$

where, $W_1$and W_2 are learnable parameter matrices, h_u,h_v are node embeddings, $\odot$ is Hadamard product, and η_{uv} are the edge gates computed as:

$$\eta_{uv} = \frac{\sigma(e_{uv}^{(k)})}{\sum_{v \in \mathcal{N}(u)} \sigma(e_{uv}^{(k)}) + \epsilon}$$

$$e_{uv}^{(k)} = \mathrm{ReLU}(A^{(k)} h_u^{(k-1)} + B^{(k)} h_v^{(k-1)} + C^{(k)} e_{uv}^{(k-1)}),$$

where, σ is the *sigmoid* function, e_{uv} are edge features, ϵ is the fixed constant for numerical stability, and A, B and C are learnable parameter matrices.

5.5.2 Dataset

Negative Sampling. The *spatio-temporal* modeling of network traffic contains only the normal behaviour of industrial network communication. For a ML model to self-learn the normality in absence of any known abnormality, the data with abnormal characteristics are generated artificially. They can be generated in multiple ways for a given domain, which is the case for current work. The following amongst the many other ways are chosen to generate abnormal characteristics data in the presented study: (1) abnormal transition order of payload bytes for *inter-Connection* transitions, (2) abnormal transition intervals outside the observed interval ranges of normal transitions, and (3) abnormal cycle counter differences outside the observed ranges of normal transitions.

Training Dataset. The training dataset consists of *graph snapshots* extracted over data captured from normal behaviour, refer section 2.5.4. Every *graph snapshot* is a graph consisting of 8 nodes and 7 edges with feature size 8 for a node and 76 for an edge. The negative sampled data was added to weigh in abnormalities as per the selected methodologies in section 5.5.2. The dataset was split into 80:20 ratio for train and validation for training of GNN models:

- Training Dataset: 1139 normal snapshots, 261 negative-sampled abnormal snapshots

- Validation Dataset: 489 normal snapshots, 113 negative-sampled abnormal snapshots

Test Dataset. To evaluate the anomaly detection performance, the traffic captured while executing network attacks is used for extracting snapshots. The anomaly use case is outlined in fig. 5.7. The process parameters on the output data exchange for *Connection PLC-2~BK-2* are modified, which causes *Lower Belt* to move in reverse direction. The faulty process payload byte transition is encoded over the edge between *PLC-2* and *BK-2* of the *graph snapshot*. All the *graph snapshots* containing the faulty payload byte transition are considered *abnormal*, otherwise *normal*. The test dataset contains 41 normal snapshots and 47 abnormal/anomalous snapshots. Every *graph snapshot* is a graph consisting of 8 nodes and 7 edges with feature size 8 for a node and 76 for an edge.

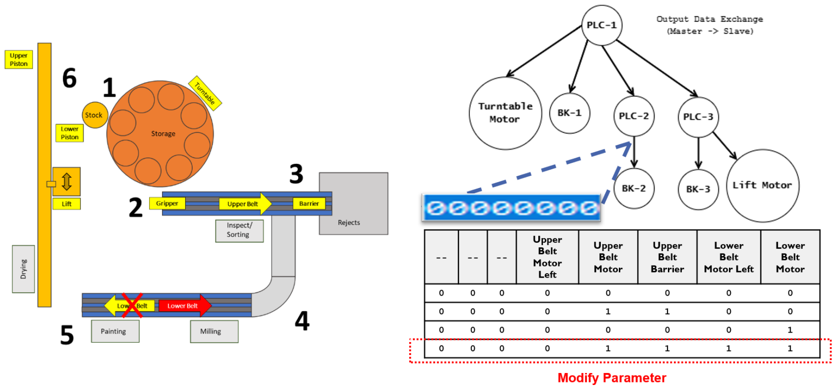

Figure 5.7: The anomaly use case of *modifying parameter*.

5.5.3 Analysis Pipeline

Dwivedi et al. [Dwi20] outlined an experimental pipeline for benchmarking of GNN, shown in fig. 5.8, and it has been incorporated in this work. The graph and its features are given as input to the *Input Layer*, followed by convolutional computations in *GNN Layer* and graph classification task in the *Prediction Layer*.

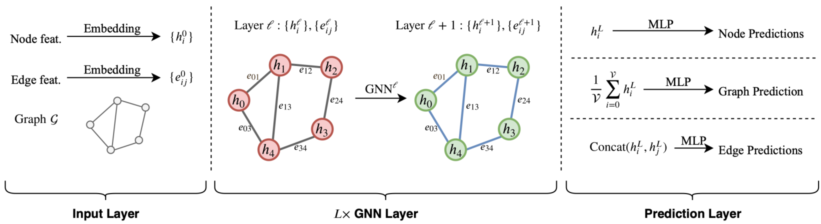

Figure 5.8: The GNN analysis pipeline adapted from Dwivedi et al. [Dwi20].

Input Layer. The node features $\alpha_i \in \mathbb{R}^{a\times 1}$ for every node i and edge features $\beta_{ij} \in \mathbb{R}^{b\times 1}$ for every edge between nodes i and j are linearly projected to d-dimensional *hidden features* $h_i^{l=0}$ and $e_{ij}^{l=0}$, respectively:

$$h_i^0 = W_1^0\alpha_i\,;\; e_{ij}^0 = W_2^0\beta_{ij} \text{ where, } W_1^0 \in \mathbb{R}^{d\times a},\, W_2 \in \mathbb{R}^{d\times b}$$

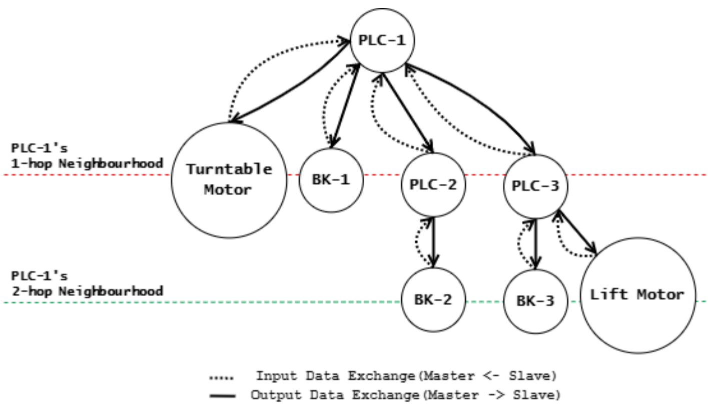

Figure 5.9: 2-hop neighbourhood of the *Festo Demonstrator*'s logical network structure.

GNN Layer. Each GNN Layer computes hidden features for each node and edge of the graph through recursive message passing in the local neighbourhood of each node. $\mathcal{L}$ GNN layers are stacked for $\mathcal{L}$-hop neighbourhood message passing computations. For the *graph snapshots*, $\mathcal{L}$ is defined by the *depth* of *graph snapshot* to allow message diffusion throughout the network. In the presented work and the industrial system under consideration, we employ 2 GNN layers for 2-hop neighbourhood computation, refer fig. 5.9.

Prediction Layer. We pool final node embeddings of all the nodes to embed the graph's features, refer section 5.3.3. A Multi-Layer Perceptron (MLP) is employed over the graph embeddings z_G for the graph classification task.

5.5.4 Implementation Frameworks

The analysis pipeline was realized in *Python* utilizing PyTorch Geometric (PyG) [Fey19] and Deep Graph Library (DGL) [Wan19] libraries for GNN architecture layers. Particularly, PyG implementations of *GCN*, *MPNN*, *GT* and DGL implementation for *GatedGCN* are utilized. Both the libraries require their own representation of datasets, hence, train and test snapshot datasets were converted into respective library formats. In addition, the libraries work over the *batches* of dataset for training and evaluation of GNN models.

In order to regularize the model parameters over different batches of dataset, *batch normalization* (BN) is augmented to each GNN layer. For every iteration of message passing,

$$h_u^{(k+1)} = UPDATE^{(k)}(BN(h_u^{(k)}, AGGREGATE^{(k)}(\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\}))$$

where, $UPDATE$ and $AGGREGATE$ are differentiable functions, $\mathcal{N}(u)$ is node u's graph neighbourhood and h_u,h_v are node embeddings.

Binary Cross Entropy (BCE) is utilized as the loss function to train GNN models for the graph classification task, which is defined as:

$$BCE = -t_1 \log(\sigma(s_1)) - (1 - t_1) \log(1 - \sigma(s_1))$$

where, t_1 and s_1 are ground truth and classification score of the sample for class C_1 of binary classes $\{C_1, C_2\}$, and σ is the sigmoid function.

Performance Metrics. *Specificity* is used to measure the GNN model's performance on detecting anomalies, whereas *Sensitivity* measures recall performance on normal *graph snapshots*, refer section 4.4.2.

System Configuration. The GNN models are computed in the *Colab Pro* environment with a *Tesla P100-PCIE-16GB* GPU offered by GOOGLE RESEARCH[1].

5.5.5 Hyperparameter Optimization

The parameters of the learning process and model parameters for different configurations of every GNN architecture are evaluated with the Training Dataset, to get the best model settings for each GNN architecture. The table 5.1 summarizes all the parameters which were optimized to get the best model configurations for each GNN architecture. For different configuration instances of each GNN architecture, the one with best performance on the Validation Dataset with high *Sensitivity* measure is chosen as the best model of the GNN architecture.

The number of *epochs* is the number of times the training dataset is passed forward and backward through the GNN. When the epochs are very low, the model is *underfitting* as the GNN has not learnt enough. On the other hand, the model might *overfit* the learning for too high epochs. The *batch size* defines the number of sub-samples from training data as input to the GNN before the weights are updated. A bigger batch size slows the learning process, whereas the lower batch size fastens the learning process. The *learning rate* controls the frequency of updating the weights at the end of each batch. A very small learning rate results in slow converging of the model, and the too large learning rate would diverge the model.

Table 5.1: Summary of hyperparamter optimization of GNN models.

Parameters	Value Options	Best Model Parameter Value			
		GCN	MPNN	GT	GatedGCN
Epochs	[*300*]	60	200	100	150
Batch Size	[*16, 32, 64*]	16	32	16	32
Learning Rate	[*0.1, 0.01, 0.001, 0.0001*]	0.1	0.01	0.001	0.001

[1] https://colab.research.google.com/

5.5.6 Evaluation Performance

The GNN architecture models with optimized parameter are evaluated on the Test Dataset, for their performances on detecting *anomalies* and recalling the learnt *normal* behaviour of the industrial process. The dataset contains 41 normal snapshots and 47 abnormal/anomalous snapshots, refer section 5.5.2. The anomalous snapshots contains the faulty process payload byte transition encoded over the edge between *PLC-2* and *BK-2*. The time taken for each hyperparameter-optimized GNN architecture with number of model parameters are summarized in table 5.3. The performance of GNN architectures for anomaly detection are summarized in table 5.2.

Table 5.2: Comparison of different GNN architectures' performance on anomaly detection and *spatio-temporal* modeling.

GNN Model	Specificity	Sensitivity
GCN (Baseline)	0.0	1.0
MPNN	1.0	1.0
GT	1.0	1.0
GatedGCN	1.0	1.0

The isotropic GCN model couldn't distinguish between *normal* and *anomalous graph snapshots* as the information is encoded on the edge features. Since, the network structure remains constant in either evaluation case and is similar to learnt *normal* behavioural graph structure, GCN recalls all the *normal graph snapshots* successfully.

The anisotropic GNN models detected the anomalies completely with complete recall on *normal graph snapshots*. For the proposed spatio-temporal characteristics encoding on the edges (refer section 5.4), the anisotropic GNN models could be incorporated for anomaly detection. The information on the edges of different connections could be enriched for analysis in more complex use cases than *'Force Attack'*, such as *Stealthy Attack* outlined in the work of Ferrari et al. [Fer20].

The evaluation result concludes the correctness of encoding mechanism outlined in section 5.4 for modeling *spatio-temporal* graph representation. The *Sensitivity* score indicates that the *anisotropic* GNN are able to learn process behaviour of the industrial network based on the *spatio-temporal* encodings on the edges. The *Specificity* score conveys the applicability of Graph pooling for the *graph classification* task as process data anomaly detection method. The detection of *'Force Attack'* by the *anisotropic* GNN based on the *spatio-temporal* encodings on the edges satisfies the BSI's "*Category D - unusual changes in process data*" requirement to detect anomalies in an industrial process, as outlined in section 1.1.1.

Table 5.3: Execution time of GNN architectures.

GNN Model	Model Parameter Count	Computation Time per 10 Epoch (s)
GCN (Baseline)	3,201	7.6
MPNN	75,361	4.2
GT	29,089	11.0
GatedGCN	187,521	6.9

The time taken to execute GNN architectures are low as the size of every evaluated *graph snapshot*, normal and anomalous, is small - 8 nodes and 7 edges. For a large industrial system, when the process complexity increases, the number of nodes and edges of the networking graph increases. The larger the size of graph, more the number of model parameters. Consequently, the GNN computation time would grow.

5.6 Discussion and Summary

5.6.1 Review

In the recent years, analogous to presented work, applications of GNN for network cybersecurity based on features extracted from network traffic have been published.

Zhou et al. [Zho20a] incorporated GCN for botnet detection. A synthetic botnet topology is overlayed on a large-scale real networks and topological information is extracted from network traffic flows. The graph representation of the network does not contain any flow features on the edges or nodes of the graph. Based solely on the connectivity information, GCN classifies nodes as part of botnet or normal infrastructure (*node classification* task).

Pujol-Perich et al. [Puj21] presented GNN's potential as robust intrusion detection methodology. The network flows between network assets are represented as a *heterogeneous host-connection* graph to model the structural relationships between different network flows. The *host* nodes represent the network assets, flows between assets are represented by the *connection* nodes and the directed edges connect the flows to source and target hosts. A GNN model based on the heterogeneous graph representation with Gated Recurrent Units [Chu14] as $UPDATE$ functions (refer eq. (5.3)) for embedding flow relationships is proposed. The embeddings of *connection* nodes are classified for specific attacks (DDoS, port scan or network scans) or benign traffic (*node classification* task).

The usage of Spatio-Temporal Graph Convolutional Network (STGCN) [Yu17] for Distributed Denial-of-Service (DDoS) attack detection is outlined in the work of Xie et al. [Xie20]. The network flow features extracted from windows over the traffic are utilized to represent characteristics of all the nodes in the topology graph of the monitored network: *total source packets, total source bytes, total destination packets, total destination bytes and unknown TCP traffic*. STGCN is shown to effectively correlate spatio and temporal features between network nodes for the underlying *node classification* task of DDoS attack detection.

GCN SCOPE [Oba20] is a graph convolutional network-based suspicious communication pair estimation framework, which learns the embeddings of *IP* addresses of a network from its communicaiton triplets using relational graph convolutional networks (R-GCNs) [Sch18]. It quantifies anomalous unobserved communication triplets, each triplet is a tuple of server *IP* address, *TCP*/*UDP* port and client *IP* address. A *multigraph* is constructed from communication triplets, and R-GCN is used for *link prediction* task to score unobserved communication triplet.

E-GraphSAGE [Lo21] is a GNN-based intrusion detection system inspired from GraphSAGE algorithm [Ham17b] to capture edge features in addition to topological information and node features. The network flow features are represented over the edges of graph representing network communication for graph representation learning. E-GraphSAGE samples its neighbouring nodes and corresponding edges of all the nodes to compute node and edge embeddings. The edge embeddings are used for *edge classification* task of network flow classification.

NF-GNN [Bus21] is a network flow graph-based approach for traffic-based malware detection. A network flow graph extracted from traffic consists of nodes corresponding to network endpoints and edges represent the network flow features of connection between endpoints. A GNN model tailored for network flow graph representation is proposed to compute graph embeddings for supervised/unsupervised *graph classification* task of malware detection.

The presented work in this chapter differs from the published literature in terms of graph representation of network traffic and the ML task. The industrial network communication is represented through *directed homogeneous* graph (refer section 5.4), which differs from the work of GCN SCOPE and Pujol-Perich et al. E-GraphSAGE samples the neighbourhood information, whereas complete neighbourhood information is required for process data analysis (refer section 5.5). The task of NF-GNN is similar to this chapter's proposed solution of utilizing GNN for process data analysis in industrial networks, hence, its usage for evaluation is scheduled as component of future work.

5.6.2 Conclusion

In this chapter, different Graph Neural Networks (GNN) architectures are evaluated for modeling *spatio-temporal* characteristics of *PROFINET* networks to detect process data anomalies. A graph representation to encode the *spatio-temporal* characteristics on the edges is presented. *Isotropic* Graph Convolutional Network (GCN) and *anisotropic* Message Passing Neural Network (MPNN), Graph Transformer (GT) and Gated Graph Convolutional Network (GatedGCN) instances are utilized to learn process behaviour from the normal operations of industrial process. Negative sampling is utilized for the learning process. The anomaly use case of 'Force Attack' on the *Festo Demonstrator* is employed for anomaly detection performance evaluation. Graph pooling for the *graph classification* task is defined as the anomaly detection methodology. The evaluation results show that GNN are capable to learn *spatio-temporal* characteristics of process data exchange in industrial networks for anomaly detection. Thus satisfying the BSI's "*Category D - unusual changes in process data*" requirement to detect anomalies in an industrial process, as outlined in section 1.1.1.

In comparison to published GNN literature for Cybersecurity, NF-GNN [Bus21] is the closest fit as additional GNN architecture to be considered in future evaluations compared to E-GraphSAGE [Lo21], GCN SCOPE [Oba20] and GNN architecture of Pujol-Perich et al. [Puj21]. The size of an industrial system's network and the complexity of its industrial process affects the size of node and edge feature encoding, consequently affecting the execution of the chosen GNN architecture computations. In conclusion, for an industrial system with deterministic industrial process, the process behaviour can be modeled as Graph with *spatio-temporal* characteristics encoded on the edges from its network traffic in a *self-learning* environment.

6 Conclusion

With the advent of 21st Century, the *fourth* industrial revolution interconnected physical industrial components with networking technology advancements to reduce production cost through efficient process control systems in the industrial production. The introduction of Ethernet-based protocols blurred the distinction between the IT office/enterprise networks and the OT ICS/industrial networks. The *sensor* in the production floor of an industrial production is made accessible to the *personnel* in the office/enterprise network. The traditional notion of *Security by Obscurity* to protect OT network from adversaries have been challenged by the cyber threats targeted at industrial systems in the last few decades (*ref. table 1.2*). The cybersecurity of industrial networks has become paramount with the adoption of new layered security design *Security in Depth.* Network-based intrusion detection system (NIDS) are one of the key countermeasure component in the layered security concept to detect cyber threats through monitoring the network traffic of an industrial communication network. The *anomaly-based* methods are effective in detecting unknown cyber threats in comparison to the *signature-based* methods for NIDS in industrial networks. Germany's Federal Office for Information Security (BSI) in its cybersecurity recommendations on production networks [BSI19] outlined anomalies in industrial networks and related categories of feature requirements for the passively monitoring anomaly detection system (*refer section 1.1.1*). The anomaly detection system learns the industrial communication network's topology, the communication links, time-related behaviour and communication content from the monitored network infrastructure information to flag the deviations as anomalies.

6.1 The Quest

When the access to the network infrastructure information of an industrial production is unavailable, configuring the NIDS for anomaly detection is challenging. An empirical solution is to self-learn the network infrastructure information (topology, assets and communication links) and the *spatio-temporal* behaviour from passive monitoring of industrial network traffic, in conjunction with anomaly detection systems to detect anomalies. However, in the absence of industrial process information, learning the process behaviour from the network traffic for anomaly detection is a challenge. Thus, the *quest* to find solutions for the research questions emerging from the challenge of self-learning anomaly detection in industrial production ensued the dissertation.

A self-learning framework for passive network monitoring and anomaly detection in industrial networks consisting of different analytical components is conceptualized as the structural skeleton to systematically explore the solutions for the research questions (*ref. section 1.2*). These analytical components are (a) *data collection and network information extraction*; (b) *network information representation*; and (c) *network information analysis*.

On the quest, the industrial system employed to develop and evaluate the solutions is the *PROFINET*-based *Festo Demonstrator* built at the IT-Security Laboratory, Fraunhofer IOSB (*ref. section 2.5*). It is a miniaturized manufacturing process mimicking a simplified painting closed-loop process scenario with *PROFINET* protocol for communication between real hardware components such as *controllers* and *devices*. In addition, two network-based cyber threats were executed during the process operations and the network traffic is passively captured from the network switch for the evaluation.

The *contributions* of the dissertation transpiring from the reported solutions for the research objectives and *future prospects* to better the results or resolve open questions are outlined in this chapter.

6.2 Industrial Network Transparency

Research Question 1: *How to extract information of an industrial system from its network traffic?*

Contributions. A *Python*-based systematic framework is conceptualized and developed to passively capture the network traffic from the *PROFINET*-based industrial system and extract the relevant network information to make the industrial network transparent for analysis. It dissects the network packets using *Scapy* framework to detect different networking operations of the industrial system. These networking operations reflect the multiple industrial operations an industrial system undergoes from start-up to the process data exchange. The network information that makes the industrial system transparent are *network topology*, *assets and their attributes*, *communication links established between master and slave* (referred as *PROFINET Connection*), *time-stamped process data payload bytes*. Thus satisfying the BSI's "*Category A - general requirements*" for network transparency.

Discussion. The framework is modular to adapt different output information formats. However, the extracted network information that are relevant for further downstream data analyses are fixed. It contains the network topology, the process communication data and assets information for topology analysis, process data analysis and asset inventory, respectively. Additionally, the framework is not restricted to one industrial protocol either. To extract the information from another industrial communication protocol such as *EtherNet/IP*, the corresponding network packet dissector library need to be employed or developed when not available.

Outlook. The captured data is analysed *offline* in a batch process and the memory requirement is quadratic to the size of captured packets (PCAP). An *online* memory-efficient packet capture and simultaneous near to real-time information extraction module with parallelization capability is the next logical enhancement step. An open question with the research objective is, *how to standardize the network information format* to maximize its compatibility for open source analysis frameworks such as *Security Onion* [22c], *Malcolm* [22b].

6.3 Industrial Operations Behaviour

Research Question 2: *How to enumerate and track an industrial system's operations from network traffic?*

Contributions. Finite State Machine (FSM) models are conceptualized and developed to represent the industrial operations and their transitions for devices, connections between master and slaves, and the industrial system. These FSMs model the valid industrial operations interpreted from PROFINET protocol specifications to detect different operation stages from the extracted network information and flag the invalid transitions as anomalies (*ref. section 3.4*). A *Python*-based framework *PROFINET Operations Enumeration and Tracking (POET)* is developed to realize FSMs and instantiate FSM Device, FSM Connection and FSM System for *PROFINET* devices, connections and the industrial system detected in the extracted network information (*ref. section 3.4.4*). POET successfully detected the implemented *PN-DCP* based network attack on the *Festo Demonstrator* mimicking an adversarial action to change the parameters of a configured *PROFINET* device during the process data exchange operation (*ref. section 3.5.1*). Consequently, POET satisfies the BSI's "*Category B - unusual or exceptional activities in an ICS network*" requirements for detecting the change in protocol communication activity.

Discussion. POET is the first-of-its-kind Protocol-analysis based NIDS for PROFINET to incorporate empirical behaviour of PROFINET system collected from real-world systems. The conceptualized FSM models at the device-, connection- and system-level granularity can be applied to other Industrial Ethernet protocol based on its specification. Consequently, the stages and the transitions between the stages of industrial operations could be enumerated and tracked from the extracted network information. The industrial system's operation behaviour could be learnt through FSMs and anomalies are detected as the invalid operations.

Outlook. Integrating POET to an event-based IDS such as *Snort* would trigger alarms when unwarranted protocol-based events are detected in the network traffic. The current version of POET uses empirical information collected from

Siemens PLC, however, the information could vary with other PLC environments such as CODESYS. Evaluating POET's performance in CODESYS environment is the future work for robustness testing. Additional network attacks violating the valid industrial operations at all the granularities are needed to evaluate the anomaly detection performance of POET.

6.4 Spatio-temporal Process Behaviour

Research Question 3: *How to represent the spatio-temporal characteristics of an industrial system's process?*

Contributions. In the absence of information on the programmed industrial process, there are no insights into the semantics of process values being exchanged. The *spatio-temporal* characteristics of the finite set of process payload bytes extracted from the process data frames are the representative of the embedded process values. The industrial process is conceptualized as the constituent of multiple sub-processes. Each sub-process represents the process data communication link established between a pair of master and slave devices exchanging a particular process parameter of the overall process.

In chapter 4, the *String* representation of ordered payload bytes where each process payload byte is alphabet-encoded has been outlined. The temporal information associated with payload bytes is utilized to define the transition intervals between pairs of adjacent encoded payload bytes, thus, capturing the *spatio-temporal* characteristics. A *Python*-based systematic framework *Payload Bytes Profiling (PBP)* has been developed to alphabet-encode the extracted process payload bytes and represent the *spatio-temporal* characteristics of an industrial system's process and its constituent sub-processes into spatial (*transition profile*) and temporal (*interval profile*) profiles from passively monitored network traffic (*ref. section 4.2.3*).

In chapter 5, the ubiquitous *Graph* representation of an industrial network is additionally explored. The *graph snapshot* representation to capture the *spatio-temporal* characteristics of industrial process has been conceptualized and developed with *nodes* representing the industrial components and the *edges* being the communication link between the components. The *graph snapshot* is created from the alphabet-encoded industrial process traffic and the transition interval information in conjunction with transitioning payload bytes and the differences in their *Cycle Counter* values are encoded on the edges over a fixed time window on the traffic (*ref. section 5.4*).

Discussion. The encoding of process data exchange as either *String* or *Graph* representation is applicable to any discrete and deterministic industrial process of an industrial production. The encoding scheme is also non-restrictive as the sequential order and the transition interval between the encoded payload bytes can be easily extracted with the knowledge of encoding scheme.

Outlook. The *graph snapshot* representation flattens out the temporal information over the edges. Another graph representation such as *discrete temporal graphs with static nodes and evolving edges* [Ska21] or *spatial–temporal graph* [Yu17] could be used to represent the *spatio-temporal* characteristics of an industrial process. In addition, the survey from Kazemi et al. [Kaz20] comprehends the representation learning for *dynamic graphs* which can act as preliminary reference to explore *dynamic graph* representations. Representation of the *spatio-temporal* characteristics of an industrial process as *bitmaps/images* is an interesting future work to lean on the advancements in *Image Processing*.

Research Question 4: *How to learn the spatio-temporal characteristics of an industrial system's process for anomaly detection from the network traffic?*

Contributions. With two different conceptualized representations of industrial process traffic capturing its *spatio-temporal* characteristics, the process behaviour is learnt to distinguish *anomalous* behaviour from the *normal*.

In chapter 4, the Regular Expression-coupled Suffix Tree algorithm is conceptualized and developed to extract the *transition profiles* at the process and sub-process granularities from the alphabet-encoded process traffic. The *transition profile* for the process and its constituent sub-processes captures the periodically recurring alphabet-encoded process payload bytes sequence, thus, modeling the *spatial* characteristics of an industrial process/sub-processes (*ref. section 4.3*). For *temporal* characteristics modeling, the transition intervals between the transitions of process payload bytes in the *transition profile* of the process/sub-process are modeled through non-parametric Machine Learning models Local Outlier Factor (LOF), Isolation Forest (IF) and One-Class Support Vector Machines (OC-SVM), and represents the *temporal* characteristics as the *interval profile* in PBP of a sub-process or the overall process, (*ref. section 4.4*). The PBP for *Festo Demonstrator*'s painting process is evaluated for anomaly detection performance when the network-based attack, *'Force Attack'* (*ref. section 2.5.3*), changes the process values in a sub-process to move the *Lower Belt* component in the opposite direction. The anomalies triggered by the attack are successfully reported utilizing the *transition profile* of the sub-process *PLC-2~BK-2* (*ref. section 2.5.3*).

In order to learn the *spatio-temporal* characteristics of an industrial process represented in *graph snapshots*, the anisotropic Graph Neural Network (GNN) architectures (Message Passing Neural Network (MPNN), Gated Graph Convolutional Network (GatedGCN) and Graph Transformer (GT)) are employed and evaluated for anomaly detection in chapter 5. The anisotropic GNN architectures learns the process behaviour with *graph classification* as the Machine Learning task to classify *graph snapshots* as *normal* or *anomalous* (*ref. section 5.5*). The anomalies triggered by the *'Force Attack'* are successfully detected by the anisotropic GNN architectures, but the isotropic Graph Convolutional Network (GCN) model failed. It demonstrated the capability of anisotropic GNN architectures to utilize the edge features encoding the *spatio-temporal* characteristics in their learning.

As the PBP framework and anisotropic GNN architectures successfully detect the change in process values of an industrial process, they satisfy the BSI's "*Category D - unusual changes in process data*" requirement of process-based anomaly detection.

Discussion. The PBP framework can be used for production cycle detection of an industrial process as demonstrated in section 4.6.1. To utilize the PBP framework in another Industrial Ethernet protocol based discrete ad deterministic industrial process environment, the corresponding protocol dissectors are required to represent the process traffic as alphabet-encoded string and collect the transition intervals. The *transition profile* and *interval profile* modeling are unaffected and can be successfully extracted. As for the GNN architectures for process behaviour analysis, the size of the industrial network could impact the computational performance.

Outlook. The Suffix Tree algorithm can be substituted with better memory-efficient Suffix Array to extract the *transition profile.* The GNN architectures that are capable of learning from the *dynamic graphs* or *spatial–temporal graph* such as Spatio-Temporal Graph Convolutional Networks (STGCN) [Yu17], Temporal Graph Network [Ros20] can be employed for Graph-based process data analysis and anomaly detection in industrial production.

Bibliography

[17] CODESYS PROFINET. Oct. 2017. URL: https://www.codesys.com/products/codesys-fieldbus/industrial-ethernet/profinet.html (visited on 07/10/2022) (cit. on p. 76).

[20] IT Security Laboratory of Fraunhofer IOSB. 2020. URL: https://www.iosb.fraunhofer.de/en/business-units/automation-digitalization/fields-of-application/it-security-industry.html (visited on 07/10/2022) (cit. on p. 15).

[21a] Confusion matrix. Dec. 2021. URL: https://en.wikipedia.org/wiki/Confusion_matrix (cit. on p. 103).

[21b] Ethercat and TSN. 2021. URL: https://www.ethercat.org/en/ethercat_and_tsn.htm (visited on 07/10/2022) (cit. on p. 28).

[21c] PROFINET over TSN. 2021. URL: https://www.profibus.com/technology/industrie-40/profinet-over-tsn (visited on 07/10/2022) (cit. on p. 28).

[21d] Time-Sensitive Networking (TSN) Task Group. 2021. URL: https://1.ieee802.org/tsn/ (visited on 07/10/2022) (cit. on p. 28).

[22a] Industroyer MITRE ICS mapping. 2022. URL: https://attack.mitre.org/software/S0604/ (visited on 07/10/2022) (cit. on p. 30).

[22b] Malcolm. 2022. URL: https://github.com/cisagov/Malcolm (cit. on p. 151).

[22c] Security Onion Solutions. 2022. URL: https://securityonionsolutions.com/ (cit. on p. 151).

[22d] Stuxnet MITRE ICS mapping. 2022. URL: https://attack.mitre.org/software/S0603/ (visited on 07/10/2022) (cit. on p. 30).

[22e] Triton MITRE ICS mapping. 2022. URL: https://attack.mitre.org/software/S1009/ (visited on 07/10/2022) (cit. on p. 30).

[Abo09] Abouelhoda, Mohamed and Ghanem, Moustafa: “String mining in bioinformatics”. In: *Scientific data mining and knowledge discovery*. Springer, 2009, pp. 207–247 (cit. on p. 91).

[Agh16] Aghdam, Mehdi Hosseinzadeh; Kabiri, Peyman et al.: “Feature selection for intrusion detection system using ant colony optimization.” In: *Int. J. Netw. Secur.* 18.3 (2016), pp. 420–432 (cit. on p. 34).

[Ahm17] Ahmed, Chuadhry Mujeeb; Palleti, Venkata Reddy and Mathur, Aditya P: “WADI: a water distribution testbed for research in the design of secure cyber physical systems”. In: *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*. 2017, pp. 25–28 (cit. on pp. 36, 37).

[Ako15] Akoglu, Leman; Tong, Hanghang and Koutra, Danai: “Graph based anomaly detection and description: a survey”. In: *Data mining and knowledge discovery* 29.3 (2015), pp. 626–688 (cit. on p. 123).

[Ale20] Alexander, Otis; Belisle, Misha and Steele, Jacob: “MITRE ATT&CK® for industrial control systems: Design and philosophy”. In: *The MITRE Corporation: Bedford, MA, USA* (2020) (cit. on p. 29).

[Alm19] Almasan, Paul; Suárez-Varela, José; Badia-Sampera, Arnau; Rusek, Krzysztof; Barlet-Ros, Pere and Cabellos-Aparicio, Albert: “Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case”. In: *arXiv preprint arXiv:1910.07421* (2019) (cit. on p. 123).

[Ant19] Anton, Simon D Duque; Lohfink, Anna Pia; Garth, Christoph and Schotten, Hans Dieter: “Security in process: Detecting attacks in industrial process data”. In: *Proceedings of the Third Central European Cybersecurity Conference*. 2019, pp. 1–6 (cit. on p. 81).

[Ash17] Ashfaq, Rana Aamir Raza; Wang, Xi-Zhao; Huang, Joshua Zhexue; Abbas, Haider and He, Yu-Lin: "Fuzziness based semi-supervised learning approach for intrusion detection system". In: *Information sciences* 378 (2017), pp. 484–497 (cit. on p. 34).

[Ben05] Benczur, Andras A; Csalogany, Karoly; Sarlos, Tamas and Uher, Mate: "Spamrank–fully automatic link spam detection work in progress". In: *Proceedings of the first international workshop on adversarial information retrieval on the web.* 2005, pp. 1–14 (cit. on p. 123).

[Bio22] Biondi, Philippe and Scapy community., the: Scapy. 2022. url: https://scapy.net/ (cit. on p. 16).

[Bol06] Bolzoni, Damiano; Etalle, Sandro and Hartel, Pieter: "Poseidon: a 2-tier anomaly-based network intrusion detection system". In: *Fourth IEEE International Workshop on Information Assurance (IWIA'06).* IEEE. 2006, 10–pp (cit. on p. 84).

[Bos92] Boser, Bernhard E; Guyon, Isabelle M and Vapnik, Vladimir N: "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory.* 1992, pp. 144–152 (cit. on p. 102).

[Bra21] Brand, Thomas: Time Sensitive Networks: Real-time ethernet. 2021. url: https://www.analog.com/en/technical-articles/a87000-time-sensitive-networks-real-time-ethernet.html (visited on 07/10/2022) (cit. on p. 28).

[Bre00] Breunig, Markus M; Kriegel, Hans-Peter; Ng, Raymond T and Sander, Jörg: "LOF: identifying density-based local outliers". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data.* 2000, pp. 93–104 (cit. on p. 100).

[Bre17] Bresson, Xavier and Laurent, Thomas: "Residual gated graph convnets". In: *arXiv preprint arXiv:1711.07553* (2017) (cit. on p. 137).

[Bru13] Bruna, Joan; Zaremba, Wojciech; Szlam, Arthur and LeCun, Yann: "Spectral networks and locally connected networks on graphs". In: *arXiv preprint arXiv:1312.6203* (2013) (cit. on p. 129).

[BSI19] BSI: Monitoring and Anomaly Detection in Production Networks. Tech. rep. Federal Office for Information Security (BSI), 2019 (cit. on pp. 9, 149).

[Bun20] Bunte, Andreas; Ressler, Henrik and Moriz, Natalia: "Automated Detection of Production Cycles in Production Plants using Machine Learning". In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA).* Vol. 1. IEEE. 2020, pp. 1423–1426 (cit. on pp. 113, 120).

[Bus21] Busch, Julian; Kocheturov, Anton; Tresp, Volker and Seidl, Thomas: "NF-GNN: Network Flow Graph Neural Networks for Malware Detection and Classification". In: *arXiv preprint arXiv:2103.03939* (2021) (cit. on pp. 12, 125, 146, 147).

[Car09] Carcano, Andrea; Fovino, Igor Nai; Masera, Marcelo and Trombetta, Alberto: "State-based network intrusion detection systems for SCADA protocols: a proof of concept". In: *International Workshop on Critical Information Infrastructures Security.* Springer. 2009, pp. 138–150 (cit. on p. 34).

[Cár11] Cárdenas, Alvaro A; Amin, Saurabh; Lin, Zong-Syun; Huang, Yu-Lun; Huang, Chi-Yen and Sastry, Shankar: "Attacks against process control systems: risk assessment, detection, and response". In: *Proceedings of the 6th ACM symposium on information, computer and communications security.* 2011, pp. 355–366 (cit. on p. 34).

[Cas07] Castillo, Carlos; Donato, Debora; Gionis, Aristides; Murdock, Vanessa and Silvestri, Fabrizio: "Know your neighbors: Web spam detection using the web topology". In: *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval.* 2007, pp. 423–430 (cit. on p. 123).

[Cas15] Caselli, Marco; Zambon, Emmanuele and Kargl, Frank: "Sequence-aware intrusion detection in industrial control systems". In: *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. 2015, pp. 13–24 (cit. on pp. 34, 81).

[Che07] Cheung, Steven; Dutertre, Bruno; Fong, Martin; Lindqvist, Ulf; Skinner, Keith and Valdes, Alfonso: "Using model-based intrusion detection for SCADA networks". In: *Proceedings of the SCADA security scientific symposium*. Vol. 46. Citeseer. 2007, pp. 1–12 (cit. on p. 34).

[Chu14] Chung, Junyoung; Gulcehre, Caglar; Cho, KyungHyun and Bengio, Yoshua: "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014) (cit. on p. 145).

[COD13] CODE, PRICE: "Industrial communication networks–Network and system security–Part 3-3: System security requirements and security levels". In: (2013) (cit. on pp. 3, 5).

[Col03] Collins, Francis S; Morgan, Michael and Patrinos, Aristides: "The Human Genome Project: lessons from large-scale biology". In: *Science* 300.5617 (2003), pp. 286–290 (cit. on p. 91).

[Cor01] Cortes, Corinna; Pregibon, Daryl and Volinsky, Chris: "Communities of interest". In: *International Symposium on Intelligent Data Analysis*. Springer. 2001, pp. 105–114 (cit. on p. 123).

[Cox07] Cox, Russ: "Regular expression matching can be simple and fast (but is slow in java, perl, php, python, ruby,...)" In: *URL: http://swtch. com/rsc/regexp/regexp1. html* (2007) (cit. on p. 94).

[Dai16] Dai, Hanjun; Dai, Bo and Song, Le: "Discriminative embeddings of latent variable models for structured data". In: *International conference on machine learning*. PMLR. 2016, pp. 2702–2711 (cit. on p. 129).

[Dai21] Daigavane, Ameya; Ravindran, Balaraman and Aggarwal, Gaurav: “Understanding Convolutions on Graphs”. In: *Distill* (2021). https://distill.pub/2021/understanding-gnns. doi: 10.23915/distill.00032 (cit. on p. 131).

[Dan14] Danielis, Peter; Skodzik, Jan; Altmann, Vlado; Schweissguth, Eike Bjoern; Golatowski, Frank; Timmermann, Dirk and Schacht, Joerg: “Survey on real-time communication via ethernet in industrial automation environments”. In: *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE. 2014, pp. 1–8 (cit. on p. 27).

[Dec05] Decotignie, J-D: “Ethernet-based real-time and industrial communications”. In: *Proceedings of the IEEE* 93.6 (2005), pp. 1102–1117 (cit. on p. 27).

[Dwi20] Dwivedi, Vijay Prakash; Joshi, Chaitanya K; Laurent, Thomas; Bengio, Yoshua and Bresson, Xavier: “Benchmarking graph neural networks”. In: *arXiv preprint arXiv:2003.00982* (2020) (cit. on pp. 139, 140).

[Dzu05] Dzung, Dacfey; Naedele, Martin; Von Hoff, Thomas P and Crevatin, Mario: “Security for industrial communication systems”. In: *Proceedings of the IEEE* 93.6 (2005), pp. 1152–1177 (cit. on p. 7).

[Fer20] Ferrari, Paolo; Sisinni, Emiliano; Bellagente, Paolo; Rinaldi, Stefano; Pasetti, Marco; Sá, Alan Oliveira de; Machado, Raphael CS; Carmo, Luiz FR da C and Casimiro, António: “Model-Based Stealth Attack to Networked Control System Based on Real-Time Ethernet”. In: *IEEE Transactions on Industrial Electronics* 68.8 (2020), pp. 7672–7683 (cit. on pp. 135, 143).

[Fey19] Fey, Matthias and Lenssen, Jan E.: “Fast Graph Representation Learning with PyTorch Geometric”. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019 (cit. on p. 141).

[Gal12] Galloway, Brendan and Hancke, Gerhard P: “Introduction to industrial control networks”. In: *IEEE Communications surveys & tutorials* 15.2 (2012), pp. 860–880 (cit. on pp. 3, 25).

[Gao10] Gao, Wei; Morris, Thomas; Reaves, Bradley and Richey, Drew: “On SCADA control system command and response injection and intrusion detection”. In: *2010 eCrime Researchers Summit.* IEEE. 2010, pp. 1–9 (cit. on p. 34).

[Gil17] Gilmer, Justin; Schoenholz, Samuel S; Riley, Patrick F; Vinyals, Oriol and Dahl, George E: “Neural message passing for quantum chemistry”. In: *International conference on machine learning.* PMLR. 2017, pp. 1263–1272 (cit. on pp. 123, 127, 129, 137).

[Gus97] Gusfield, Dan: “Algorithms on stings, trees, and sequences: Computer science and computational biology”. In: *Acm Sigact News* 28.4 (1997), pp. 41–60 (cit. on p. 91).

[Had14] Hadžiosmanović, Dina; Sommer, Robin; Zambon, Emmanuele and Hartel, Pieter H: “Through the eye of the PLC: semantic security monitoring for industrial processes”. In: *Proceedings of the 30th Annual Computer Security Applications Conference.* 2014, pp. 126–135 (cit. on pp. 34, 81, 82).

[Ham17a] Hamilton, William L; Ying, Rex and Leskovec, Jure: “Inductive representation learning on large graphs”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems.* 2017, pp. 1025–1035 (cit. on p. 129).

[Ham17b] Hamilton, William L; Ying, Rex and Leskovec, Jure: “Inductive representation learning on large graphs”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems.* 2017, pp. 1025–1035 (cit. on pp. 130, 146).

[Ham20] Hamilton, William L: “Graph representation learning”. In: *Synthesis Lectures on Artifical Intelligence and Machine Learning* 14.3 (2020), pp. 1–159 (cit. on pp. 128, 130, 132).

[Hin18] Hindy, Hanan; Brosset, David; Bayne, Ethan; Seeam, Amar; Tachtatzis, Christos; Atkinson, Robert and Bellekens, Xavier: "A taxonomy and survey of intrusion detection system design techniques, network threats and datasets". In: (2018) (cit. on pp. 13, 14).

[Hol15] Holm, Hannes; Karresand, Martin; Vidström, Arne and Westring, Erik: "A survey of industrial control system testbeds". In: *Nordic Conference on Secure IT Systems*. Springer. 2015, pp. 11–26 (cit. on p. 35).

[Hol21] Holland, Jordan; Schmitt, Paul; Feamster, Nick and Mittal, Prateek: "New directions in automated traffic analysis". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 3366–3383 (cit. on p. 13).

[Hol93] Holzmann, Gerard J: "Design and validation of protocols: a tutorial". In: *Computer networks and ISDN systems* 25.9 (1993), pp. 981–1017 (cit. on p. 66).

[Hon14] Hong, Junho; Liu, Chen-Ching and Govindarasu, Manimaran: "Detection of cyber intrusions using network-based multicast messages for substation automation". In: *ISGT 2014*. IEEE. 2014, pp. 1–5 (cit. on p. 34).

[Hou12] Hou, Chongyuan; Jiang, Hanhong; Rui, Wanzhi and Liu, Liang: "A probabilistic principal component analysis approach for detecting traffic anomaly in industrial networks". In: *Journal of Xi'an Jiaotong University* 46.2 (2012), pp. 70–75 (cit. on p. 34).

[Hu18] Hu, Yan; Yang, An; Li, Hong; Sun, Yuyan and Sun, Limin: "A survey of intrusion detection on industrial control systems". In: *International Journal of Distributed Sensor Networks* 14.8 (2018), p. 1550147718794615 (cit. on pp. 33, 76, 81).

[Hum22] Humphrey, David: What is OPC UA TSN for industrial networking? 2022. url: https://www.arcweb.com/industry-best-practices/what-opc-ua-tsn-industrial-networking (cit. on p. 28).

[Ili11] Iliofotou, Marios; Kim, Hyun-chul; Faloutsos, Michalis; Mitzenmacher, Michael; Pappu, Prashanth and Varghese, George: "Graption: A graph-based P2P traffic classification framework for the internet backbone". In: *Computer Networks* 55.8 (2011), pp. 1909–1920 (cit. on p. 123).

[Kaz20] Kazemi, Seyed Mehran; Goel, Rishab; Jain, Kshitij; Kobyzev, Ivan; Sethi, Akshay; Forsyth, Peter and Poupart, Pascal: "Representation Learning for Dynamic Graphs: A Survey." In: *J. Mach. Learn. Res.* 21.70 (2020), pp. 1–73 (cit. on p. 154).

[Kim05] Kim, Seong Soo and Reddy, AL Narasimha: "Modeling network traffic as images". In: *IEEE International Conference on Communications, 2005. ICC 2005. 2005.* Vol. 1. IEEE. 2005, pp. 168–172 (cit. on p. 13).

[Kip16] Kipf, Thomas N and Welling, Max: "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016) (cit. on pp. 127, 136).

[Kis15] Kiss, Istvan; Genge, Bela and Haller, Piroska: "A clustering-based approach to detect cyber attacks in process control systems". In: *2015 IEEE 13th international conference on industrial informatics (INDIN).* IEEE. 2015, pp. 142–148 (cit. on p. 34).

[Kle56] Kleene, Stephen C et al.: "Representation of events in nerve nets and finite automata". In: *Automata studies* 34 (1956), pp. 3–41 (cit. on p. 66).

[Kor17] Korkmaz, Emrah; Davis, Matthew; Dolgikh, Andrey and Skormin, Victor: "Detection and mitigation of time delay injection attacks on industrial control systems with PLCs". In: *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security.* Springer. 2017, pp. 62–74 (cit. on p. 121).

[Lee17] Lee, Robert M; Assante, MJ and Conway, T: "Crashoverride: Analysis of the threat to electric grid operations". In: *Dragos Inc., March* (2017) (cit. on p. 7).

[Lem16] Lemay, Antoine and Fernandez, José M: “Providing {SCADA} network data sets for intrusion detection research”. In: *9th Workshop on Cyber Security Experimentation and Test (CSET 16)*. 2016 (cit. on pp. 36, 37).

[Lin13a] Lin, Hui; Slagell, Adam; Di Martino, Catello; Kalbarczyk, Zbigniew and Iyer, Ravishankar K: “Adapting bro into scada: building a specification-based intrusion detection system for the dnp3 protocol”. In: *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*. 2013, pp. 1–4 (cit. on p. 34).

[Lin13b] Lin, Hui; Slagell, Adam; Kalbarczyk, Zbigniew; Sauer, Peter W and Iyer, Ravishankar K: “Semantic security analysis of SCADA networks to detect malicious control commands in power grids”. In: *Proceedings of the first ACM workshop on Smart energy grid security*. 2013, pp. 29–34 (cit. on p. 34).

[Liu08] Liu, Fei Tony; Ting, Kai Ming and Zhou, Zhi-Hua: “Isolation forest”. In: *2008 eighth ieee international conference on data mining*. IEEE. 2008, pp. 413–422 (cit. on p. 101).

[Lo21] Lo, Wai Weng; Layeghy, Siamak; Sarhan, Mohanad; Gallagher, Marcus and Portmann, Marius: “E-GraphSAGE: A Graph Neural Network based Intrusion Detection System”. In: *arXiv preprint arXiv:2103.16329* (2021) (cit. on pp. 146, 147).

[Mag14] Maglaras, Leandros A and Jiang, Jianmin: “Intrusion detection in SCADA systems using machine learning techniques”. In: *2014 Science and Information Conference*. IEEE. 2014, pp. 626–631 (cit. on p. 34).

[Mak21] Makrakis, Georgios Michail; Kolias, Constantinos; Kambourakis, Georgios; Rieger, Craig and Benjamin, Jacob: “Vulnerabilities and Attacks Against Industrial Control Systems and Critical Infrastructures”. In: *arXiv preprint arXiv:2109.03945* (2021) (cit. on p. 79).

[Man93] Manber, Udi and Myers, Gene: “Suffix arrays: a new method for on-line string searches”. In: *siam Journal on Computing* 22.5 (1993), pp. 935–948 (cit. on p. 120).

[Mar17] Marcheggiani, Diego and Titov, Ivan: “Encoding sentences with graph convolutional networks for semantic role labeling”. In: *arXiv preprint arXiv:1703.04826* (2017) (cit. on pp. 123, 137).

[Mat16] Mathur, Aditya P and Tippenhauer, Nils Ole: “SWaT: A water treatment testbed for research and training on ICS security”. In: *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*. IEEE. 2016, pp. 31–36 (cit. on pp. 36, 37).

[McC76] McCreight, Edward M: “A space-economical suffix tree construction algorithm”. In: *Journal of the ACM (JACM)* 23.2 (1976), pp. 262–272 (cit. on p. 91).

[Mer05] Merkwirth, Christian and Lengauer, Thomas: “Automatic generation of complementary descriptors with molecular graph networks”. In: *Journal of chemical information and modeling* 45.5 (2005), pp. 1159–1168 (cit. on p. 124).

[Mes18] Meshram, Ankush: Deterministic Industrial Network Communication: Fundamentals. KIT Scientific Publishing, 2018 (cit. on p. 47).

[Mor12] Morris, Thomas; Vaughn, Rayford and Dandass, Yoginder: “A retrofit network intrusion detection system for MODBUS RTU and ASCII industrial control systems”. In: *2012 45th Hawaii International Conference on System Sciences*. IEEE. 2012, pp. 2338–2345 (cit. on p. 34).

[Mor14] Morris, Thomas and Gao, Wei: “Industrial control system traffic data sets for intrusion detection research”. In: *International conference on critical infrastructure protection*. Springer. 2014, pp. 65–78 (cit. on pp. 36, 37).

[Mow05] Mower, Jeffrey P: “PREP-Mt: predictive RNA editor for plant mitochondrial genes”. In: *BMC bioinformatics* 6.1 (2005), pp. 1–15 (cit. on p. 104).

[Mye17] Myers, David; Radke, Kenneth; Suriadi, Suriadi and Foo, Ernest: “Process discovery for industrial control system cyber attack detection”. In: *IFIP International Conference on ICT Systems Security and Privacy Protection.* Springer. 2017, pp. 61–75 (cit. on p. 34).

[NID21] NIDEBORN, JOAKIM. 2021. url: https://www.hms-networks.com/news-and-insights/news-from-hms/2021/03/31/continued-growth-for-industrial-networks-despite-pandemic (visited on 07/10/2022) (cit. on p. 15).

[Oba20] Oba, Tatsumi and Taniguchi, Tadahiro: “Graph convolutional network-based suspicious communication pair estimation for industrial control systems”. In: *arXiv preprint arXiv:2007.10204* (2020) (cit. on pp. 145, 147).

[Pax99] Paxson, Vern: “Bro: a system for detecting network intruders in real-time”. In: *Computer networks* 31.23-24 (1999), pp. 2435–2463 (cit. on p. 13).

[Ped11] Pedregosa, F. et al.: “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 104).

[Per09] Perdisci, Roberto; Ariu, Davide; Fogla, Prahlad; Giacinto, Giorgio and Lee, Wenke: “McPAD: A multiple classifier system for accurate payload-based anomaly detection”. In: *Computer networks* 53.6 (2009), pp. 864–881 (cit. on p. 84).

[Pfr16] Pfrang, Steffen; Kippe, Jörg; Meier, David and Haas, Christian: “Design and architecture of an industrial it security lab”. In: *International Conference on Testbeds and Research Infrastructures.* Springer. 2016, pp. 114–123 (cit. on p. 15).

[Pfr17] Pfrang, Steffen and Meier, David: “On the Detection of Replay Attacks in Industrial Automation Networks Operated with Profinet IO.” In: *ICISSP*. 2017, pp. 683–693 (cit. on pp. 73, 76).

[Pol22] Polzer, Dominik: A Comprehensive Beginner’s Guide to the Diverse Field of Anomaly Detection. 2022. url: https://towardsdatascience.com/a-comprehensive-beginners-guide-to-the-diverse-field-of-anomaly-detection-8c818d153995 (visited on 07/10/2022) (cit. on p. 99).

[Pop14] Popp, Manfred: Industrial communication with PROFINET. Profibus Nutzerorganisation, 2014 (cit. on p. 66).

[Por21] Pordelkhaki, Moojan; Fouad, Shereen and Josephs, Mark: “Intrusion Detection for Industrial Control Systems by Machine Learning using Privileged Information”. In: *2021 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE. 2021, pp. 1–6 (cit. on p. 81).

[Puj21] Pujol-Perich, David; Suárez-Varela, José; Cabellos-Aparicio, Albert and Barlet-Ros, Pere: “Unveiling the potential of Graph Neural Networks for robust Intrusion Detection”. In: *arXiv preprint arXiv:2107.14756* (2021) (cit. on pp. 12, 125, 145–147).

[Pyt] Pytransitions: Pytransitions/Transitions: A lightweight, object-oriented finite state machine implementation in python with many extensions. url: https://github.com/pytransitions/transitions (visited on 07/10/2022) (cit. on p. 72).

[Qiu18] Qiu, Jiezhong; Tang, Jian; Ma, Hao; Dong, Yuxiao; Wang, Kuansan and Tang, Jie: “Deepinf: Social influence prediction with deep learning”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 2110–2119 (cit. on p. 123).

[Rah12] Rahman, Md Sazzadur; Huang, Ting-Kai; Madhyastha, Harsha V and Faloutsos, Michalis: “Efficient and scalable socware detection in online social networks”. In: *21st {USENIX} Security*

Symposium ({USENIX} Security 12). 2012, pp. 663–678 (cit. on p. 123).

[Reu02] Reussner, Ralf: Counter-constrained Finite State Machines: Modelling Component Protocols with Resource-dependencies. Univ., Fak. für Informatik, Bibliothek, 2002 (cit. on p. 66).

[Rie17] Rieth, Cory A.; Amsel, Ben D.; Tran, Randy and Cook, Maia B.: “Additional Tennessee Eastman Process Simulation Data for Anomaly Detection Evaluation”. In: 2017 (cit. on pp. 36, 37).

[Ros20] Rossi, Emanuele; Chamberlain, Ben; Frasca, Fabrizio; Eynard, Davide; Monti, Federico and Bronstein, Michael: “Temporal graph networks for deep learning on dynamic graphs”. In: *arXiv preprint arXiv:2006.10637* (2020) (cit. on p. 156).

[Rub17] Rubio, Juan Enrique; Alcaraz, Cristina; Roman, Rodrigo and Lopez, Javier: “Analysis of Intrusion Detection Systems in Industrial Ecosystems.” In: *SECRYPT*. 2017, pp. 116–128 (cit. on p. 124).

[Sca08] Scarselli, Franco; Gori, Marco; Tsoi, Ah Chung; Hagenbuchner, Markus and Monfardini, Gabriele: “The graph neural network model”. In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80 (cit. on p. 124).

[Sch18] Schlichtkrull, Michael; Kipf, Thomas N; Bloem, Peter; Van Den Berg, Rianne; Titov, Ivan and Welling, Max: “Modeling relational data with graph convolutional networks”. In: *European semantic web conference*. Springer. 2018, pp. 593–607 (cit. on p. 145).

[Sch99] Schölkopf, Bernhard; Williamson, Robert C; Smola, Alexander J; Shawe-Taylor, John; Platt, John C et al.: “Support vector method for novelty detection.” In: *NIPS*. Vol. 12. Citeseer. 1999, pp. 582–588 (cit. on p. 102).

[Shi20] Shi, Yunsheng; Huang, Zhengjie; Feng, Shikun; Zhong, Hui; Wang, Wenjin and Sun, Yu: "Masked label prediction: Unified message passing model for semi-supervised classification". In: *arXiv preprint arXiv:2009.03509* (2020) (cit. on p. 137).

[Ska21] Skarding, Joakim; Gabrys, Bogdan and Musial, Katarzyna: "Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey". In: *IEEE Access* 9 (2021), pp. 79143–79168 (cit. on p. 154).

[Som10] Sommer, Robin and Paxson, Vern: "Outside the closed world: On using machine learning for network intrusion detection". In: *2010 IEEE symposium on security and privacy*. IEEE. 2010, pp. 305–316 (cit. on p. 35).

[Sri14] Sridhar, Siddharth and Govindarasu, Manimaran: "Model-based attack detection and mitigation for automatic generation control". In: *IEEE Transactions on Smart Grid* 5.2 (2014), pp. 580–591 (cit. on p. 34).

[Ste93] Stevens, W Richard: TCP/IP illustrated vol. I: the protocols. Pearson Education India, 1993 (cit. on p. 66).

[Sto08] Stouffer, Keith; Falco, Joseph and Scarfone, Karen: Guide to industrial control systems (ICS) security. Tech. rep. National Institute of Standards and Technology, 2008 (cit. on p. 3).

[Str18] Strom, Blake E; Applebaum, Andy; Miller, Doug P; Nickels, Kathryn C; Pennington, Adam G and Thomas, Cody B: "Mitre att&ck: Design and philosophy". In: *Technical report* (2018) (cit. on pp. 30, 73).

[Suk16] Sukhbaatar, Sainbayar; Fergus, Rob et al.: "Learning multi-agent communication with backpropagation". In: *Advances in neural information processing systems* 29 (2016), pp. 2244–2252 (cit. on p. 137).

[Sun09] Sung, Wing-Kin: Algorithms in bioinformatics: A practical introduction. CRC Press, 2009 (cit. on pp. 93, 120).

[Teu10] TEUMIM, David J: Industrial network security. Isa, 2010 (cit. on p. 35).

[Tri21] TRIFONOV, Hristo and HEFFERNAN, Donal: "OPC UA TSN: a next-generation network for Industry 4.0 and IIoT". In: *International Journal of Pervasive Computing and Communications* (2021) (cit. on p. 28).

[Ukk95] UKKONEN, Esko: "On-line construction of suffix trees". In: *Algorithmica* 14.3 (1995), pp. 249–260 (cit. on p. 91).

[V P18] V. (PNO), PROFIBUS Nutzerorganisation e.: PROFINET System Description. Tech. rep. PROFIBUS & PROFINET International (PI), 2018 (cit. on p. 47).

[Vas17] VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N; KAISER, Łukasz and POLOSUKHIN, Illia: "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008 (cit. on p. 137).

[Wan04] WANG, Ke and STOLFO, Salvatore J: "Anomalous payload-based network intrusion detection". In: *International workshop on recent advances in intrusion detection*. Springer. 2004, pp. 203–222 (cit. on p. 84).

[Wan06] WANG, Ke; PAREKH, Janak J and STOLFO, Salvatore J: "Anagram: A content anomaly detector resistant to mimicry attack". In: *International workshop on recent advances in intrusion detection*. Springer. 2006, pp. 226–248 (cit. on p. 84).

[Wan19] WANG, Minjie et al.: "Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks". In: *arXiv preprint arXiv:1909.01315* (2019) (cit. on p. 141).

[Wei73] WEINER, Peter: "Linear pattern matching algorithms". In: *14th Annual Symposium on Switching and Automata Theory (swat 1973)*. IEEE. 1973, pp. 1–11 (cit. on p. 91).

[Wil18] WILAMOWSKI, Bogdan M and IRWIN, J David: Industrial communication systems. CRC Press, 2018 (cit. on pp. 49, 79).

[Wre13] Wressnegger, Christian; Schwenk, Guido; Arp, Daniel and Rieck, Konrad: “A close look on n-grams in intrusion detection: anomaly detection vs. classification”. In: *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*. 2013, pp. 67–76 (cit. on p. 12).

[Wre18] Wressnegger, Christian; Kellner, Ansgar and Rieck, Konrad: “Zoe: Content-based anomaly detection for industrial control systems”. In: *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE. 2018, pp. 127–138 (cit. on p. 12).

[Wu19] Wu, Qitian; Zhang, Hengrui; Gao, Xiaofeng; He, Peng; Weng, Paul; Gao, Han and Chen, Guihai: “Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems”. In: *The World Wide Web Conference*. 2019, pp. 2091–2102 (cit. on p. 123).

[Xia20] Xiao, Qingsai; Liu, Jian; Wang, Quiyun; Jiang, Zhengwei; Wang, Xuren and Yao, Yepeng: “Towards network anomaly detection using graph embedding”. In: *International Conference on Computational Science*. Springer. 2020, pp. 156–169 (cit. on p. 123).

[Xie20] Xie, Qian; Huang, Zheng; Guo, Jie and Qiu, Weidong: “Spatio-Temporal Graph Convolutional Networks for DDoS Attack Detecting”. In: *International Conference on Machine Learning for Cyber Security*. Springer. 2020, pp. 153–159 (cit. on p. 145).

[Yu17] Yu, Bing; Yin, Haoteng and Zhu, Zhanxing: “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting”. In: *arXiv preprint arXiv:1709.04875* (2017) (cit. on pp. 145, 154, 156).

[Zho20a] Zhou, Jiawei; Xu, Zhiying; Rush, Alexander M and Yu, Minlan: “Automating botnet detection with graph neural networks”. In: *arXiv preprint arXiv:2003.06344* (2020) (cit. on pp. 125, 145).

[Zho20b] Zhou, Jie; Cui, Ganqu; Hu, Shengding; Zhang, Zhengyan; Yang, Cheng; Liu, Zhiyuan; Wang, Lifeng; Li, Changcheng and Sun, Maosong: "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81 (cit. on p. 125).

[Zhu21] Zhu, Hang; Gupta, Varun; Ahuja, Satyajeet Singh; Tian, Yuandong; Zhang, Ying and Jin, Xin: "Network planning with deep reinforcement learning". In: *Proceedings of the 2021 ACM SIGCOMM 2021 Conference.* 2021, pp. 258–271 (cit. on p. 123).

[Zit18] Zitnik, Marinka; Agrawal, Monica and Leskovec, Jure: "Modeling polypharmacy side effects with graph convolutional networks". In: *Bioinformatics* 34.13 (2018), pp. i457–i466 (cit. on p. 127).

Publications

[1] Meshram, Ankush and Haas, Christian: "Anomaly detection in industrial networks using machine learning: a roadmap". In: *Machine Learning for Cyber Physical Systems*. Springer, 2017, pp. 65–72.

[2] Meshram, Ankush: "Deterministic Industrial Network Communication: Fundamentals". In: *Proceedings of the 2017 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory. Ed.: J. Beyerer*. KIT Scientific Publishing, 2018, p. 45.

[3] Patzer, Florian; Meshram, Ankush; Birnstill, Pascal; Haas, Christian and Beyerer, Jürgen: "Towards computer-aided security life cycle management for critical industrial control systems". In: *International Conference on Critical Information Infrastructures Security*. Springer, 2018, pp. 45–56.

[4] Patzer, Florian; Meshram, Ankush and Heß, Maximilian: "Automated Incident Response for Industrial Control Systems Leveraging Software-defined Networking". In: *Proceedings of the 5th International Conference on Information Systems Security and Privacy - ICISSP*. SciTePress, 2019, pp. 319–327.

[5] Borcherding, Anne; Feldmann, Lukas; Karch, Markus; Meshram, Ankush and Beyerer, Jürgen: "Towards a Better Understanding of Machine Learning based Network Intrusion Detection Systems in Industrial Networks". In: *Proceedings of the 8th International Conference on Information Systems Security and Privacy - ICISSP*. SciTePress, 2022, pp. 314–325.

Supervised student theses

[1] Gehring, Ruben: "Testbench for Intrusion Detection in Industrial Networks". Master's Thesis. Karlsruhe Institute of Technology, 2018.

[2] Heß, Maximilian: "Regelbasiertes Incident Handling für Software-defined Networks in industriellen Umgebungen". Master's Thesis. Karlsruhe Institute of Technology, 2018.

[3] Feldmann, Lukas: "A Systematic Approach to Predict the Intrusion Detection Model in Industrial Networks". Master's Thesis. Karlsruhe Institute of Technology, 2020.

[4] Laly, Yorick: "Analyse der elektrischen Signale zur Angriffserkennung in industriellen Systemen basierend auf maschinellem Lernen". Master's Thesis. Karlsruhe Institute of Technology, 2021.

List of Figures

List of Tables

Listings

Acronyms

AC	Access Control
APT	Advanced Persistent Threats
AR	Application Relationship
ARP	Address Resolution Protocol
BSI	Federal Office for Information Security
CAN	Controller Area Network
CR	Communication Relationship
DC	Data Confidentiality
DCS	Distributed Control Systems
DDoS	Distributed Denial-of-Service
DGL	Deep Graph Library
DI	Data Integrity
DMZ	DeMilitarized Zone
DNA	Deoxyribonucleic acid

DTMC	Discrete-Time Markov Chain
ERP	Enterprise Resource Planning
EtherNet/IP	Ethernet Industrial Protocol
FF	Foundation Fieldbus
FSM	Finite State Machines
FTP	File Transfer Protocol
GatedGCN	Gated Graph Convolutional Network
GCN	Graph Convolutional Network
GMM	Gaussian Mixture Model
GNN	Graph Neural Networks
GSD	General System Description
GT	Graph Transformer
HIDS	Host-based Intrusion Detection Systems
HMI	Human-Machine Interfaces
HSE	Human, Societal and Environmental
I/O	Input/Output
ICS	Industrial Control Systems
IDS	Intrusion Detection Systems

IEC International Electrotechnical Commission

IF Isolation Forest

IOSB Fraunhofer Institute of Optronics, System Technologies and Image Exploitation, Karlsruhe

IP Internet Protocol

IRT Isochronous Real-Time

ISA Instrumentation Society of America

ISO International Standards Organisation

IT Information Technology

LLDP Link Layer Discovery Protocol

LOF Local Outlier Factor

LRS longest repeating and non-overlapping substring

MAC Media Access Control

MAP/EPA Manufacturing Automation Protocol/Enhanced Performance Architecture

MES Manufacturing Execution Systems

MitM Man-in-the-Middle

ML Machine Learning

MLP Multi-Layer Perceptron

Modbus/TCP Modbus Transmission Control Protocol

MPNN Message Passing Neural Network

NIDS Network-based Intrusion Detection Systems

NN Neural Networks

OC-SVM One-Class Support Vector Machines

OPC UA Open Platform Communications Unified Architecture

OSI Open System Interconnection

OT Operational Technology

PBP Payload Bytes Profiling

PCA Principal Component Analysis

PDev Physical Device Management

PLC Programmable Logic Controller

PN-CM PROFINET Context Manager

PN-DCP PROFINET Discovery and Configuration Protocol

PNIO PROFINET Input/Output

POET PROFINET Operations Enumeration and Tracking

PROFIBUS Process Field Bus

PROFINET Process Field Net

PyG	PyTorch Geometric
RA	Resource Availability
RDF	Restrict Data Flow
RNA	Ribonucleic acid
RT	Real-Time
RTE	Real-Time Ethernet
SCADA	Supervisory Control and Data Acquisition systems
SERCOS	Serial Real-time Communication System
SNMP	Simple Network Management Protocol
SSH	Secure Shell Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TLV	Type Length Value
TRE	Timely Response to Event
TSN	Time Sensitive Networking
UC	Use Control
UDP	User Datagram Protocol
VPN	Virtual Private Networks