Preprints.org

Article

Not peer-reviewed version

Reading between the Lines: Process Mining on OPC UA Network Data

Markus Hornsteiner * , Philip Empl , Timo Bunghardt , Stefan Schönig

Posted Date: 24 May 2024

doi: 10.20944/preprints202405.1589.v1

Keywords: Process Mining; Industrial Internet of Things; Business Process Management; Industry 4.0

Preprints.org is a free multidiscipline platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Reading between the Lines: Process Mining on OPC UA Network Data

Hornsteiner Markus *, Empl Philip, Bunghardt Timo and Schönig Stefan

University of Regensburg, Regensburg, Germany; philip.empl@ur.de (E.P.); Timo-Bunghardt@t-online.de (B.T.); stefan.schoenig@ur.de (S.S.)

* Correspondence: markus.hornsteiner@ur.de

Abstract: The introduction of the Industrial Internet of Things (IIoT) has led to major changes in the industry. Thanks to machine data, business process management methods and techniques could also be applied to them. However, one data source has so far remained untouched: The network data of the machines. In the business environment, process mining, for example, has already been carried out based on network data, but the IIoT, with its particular protocols such as OPC UA, has yet to be investigated. With the help of design science research and on the shoulders of CRISP-DM, we first develop a framework for process mining in the IIoT in this paper. We then apply the framework to real-world IIoT network traffic data and evaluate the outcome and performance of our approach in detail. We find tremendous potential in network traffic data but also limitations. Among other things, due to the dependence on process experts and the existence of case IDs.

Keywords: process mining; industrial Internet of Things; business process management; Industry 4.0

1. Introduction

Industrial Internet of Things (IIoT) technologies have ushered in a new era of manufacturing and industrial processes, offering unprecedented levels of connectivity, automation, and data-driven decision-making. In the heart of these dynamic ecosystems lies the seamless exchange of information among interconnected devices, sensors, and control systems [1]. This intricate web of interactions, facilitated by standard industrial communication protocols such as OPC UA (Open Platform Communications Unified Architecture)[2] and MQTT (Message Queue Telemetry Transport), generates vast volumes of network data, which, until recently, remained an untapped resource for unraveling the underlying operational intricacies [3,4].

In this paper, we delve into the realm of process mining as a transformative approach to extract invaluable insights from collected network data in Industrial IoT environments. Process mining, a field at the confluence of data science, machine learning, and process management, refers to the automated discovery, monitoring, and improvement of process models from event data of IT systems [5]. Event data is used in the research area of process mining to generate and compare process models automatically with the help of process mining algorithms. Event information can be generated by classical IT systems as well as by employees using smart devices, (production) machines, and sensors [6–8]. IT systems within an organization create, for example, records of activities performed, messages sent, or transaction data. These event data are compiled into event logs and form the starting point for process mining algorithms.

The application of process mining in the context of IIoT network data opens up a wealth of opportunities for industries seeking to enhance operational efficiency, reduce downtime, and improve overall performance [9]. Our research focuses on harnessing the power of process mining to analyze the communication patterns, information flows, and transactional sequences embedded within the data streams of the OPC UA protocol. These protocols are the backbone of communication in many industrial settings, facilitating data exchange between devices, sensors, and supervisory control systems. By scrutinizing the recorded network data with advanced process mining techniques, we aim to uncover hidden process models, identify bottlenecks, and optimize data flows, ultimately enabling industries to make informed decisions for improved productivity and reliability.

© 2024 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

Using network data to discover (business) processes is an emerging research area, gaining increasing attention recently [10–12]. Our approach differs from previous approaches in that it operates on network data in the IIoT and thus requires even more process steps to transfer the raw network data into a processable format for process mining. Besides, to the best of our knowledge, we are the first to use real-world network traffic data instead of simulated ones. This paper outlines our method for collecting, preprocessing, and analyzing network data from IIoT environments, shedding light on the challenges and intricacies of handling large-scale, heterogeneous data sources. We present a real-world use case illustrating the practical application of process mining on OPC UA data, showcasing how this approach can yield actionable insights that translate into tangible operational benefits. In summary, integrating process mining with network data in IIoT environments promises to revolutionize the way industries operate, offering a data-driven lens through which to optimize processes, enhance decision-making, and unlock the full potential of IIoT technologies. This paper comprehensively explores this innovative intersection, shedding light on its theoretical foundations, practical implementation, and the transformative impact it can have on industrial operations.

The paper is structured as follows: in Section 2, we present essential basics and related literature on process mining and network traffic data. This is followed in Section 3 by our method to discover business processes in the IIoT. In Section 4, we present the implementation of the method, which we apply to a real-world use case in Section 5. We evaluate and discuss our method in Section 6 regarding performance and quality and conclude the paper in Section 7.

2. Background and Related Work

2.1. Process Mining and Network Event Data

Event data, generated during business process execution, includes details on activities, their sequence, timestamps, and contextual information. Event data is derived from systems like databases, software applications, or sensors and is the foundation for process mining. By combining data mining, machine learning, and process management techniques, process mining analyzes and visualizes event data to reconstruct and model organizational processes. It identifies inefficiencies, bottlenecks, compliance issues, deviations, and improvement opportunities. Process mining relies on event logs as its core input, forming the basis for analyzing and optimizing processes. Network data is precious for process mining due to various reasons:

- *Rich Information Source:* Network data contains information generated by interconnected devices and systems. It captures interactions and communications between entities, providing a detailed record of activities and their sequence.
- *Granularity and Detail:* Network data often offer granular insights into the flow of activities and dependencies among different elements within a system. This information can be valuable for reconstructing processes accurately.
- *Real-Time and Continuous Data:* Networks continuously generate data in real-time as activities occur, providing an up-to-date and comprehensive view of ongoing processes. This real-time nature enables immediate analysis of process deviations or inefficiencies.
- *Comprehensive Coverage:* Network data often covers various activities, including structured and unstructured data, allowing for a holistic view of processes.
- *Interconnection of Systems:* In many cases, processes are interconnected across various systems or devices. Analyzing network data helps understand the interactions and dependencies among these systems, offering insights into end-to-end processes.

Network data, though rich, can be complex and varied, requiring specialized expertise for effective preprocessing, analysis, and interpretation in process mining. Regarding the IIoT, OPC UA is a widely used communication protocol in industrial automation, ensuring secure data exchange among devices in a networked environment. It supports various encoding formats such as binary, JSON, and XML, with binary favored for performance-critical applications and JSON/XML for web-based

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

systems. In OPC UA, packets follow a request-response pattern, ensuring secure data transmission channels. The packet structure includes a message header with crucial information, a message body, and defined services for client-server interactions, covering operations like data reading/writing and event subscription. OPC UA handshakes involve a request handle in each packet, matching responses to original requests, supporting asynchronous communication, and aiding in error responses for processing issues.

2.2. OPC UA Protocol

OPC UA (*Open Platform Communications Unified Architecture*) is a machine-to-machine communication protocol that is widely used in industrial automation systems. It provides a framework for secure and reliable data exchange between various devices and applications in a networked environment. OPC UA supports multiple data encoding formats to represent information during communication. These formats include binary, JSON (*JavaScript Object Notation*), and XML (*eXtensible Markup Language*). Each format has its own characteristics and usage scenarios. Binary encoding is often preferred for performance-critical applications with limited bandwidth, while JSON and XML are more commonly used in web-based and interoperable systems where human readability and compatibility are important factors.

Table 1 provides an overview of the OPC UA packet structure. OPC UA allows establishing secure channels to ensure the confidentiality and integrity of data transmission (**A**). OPC UA messages are encapsulated within the secure channel (if used) or directly transmitted over the network. The message header contains essential information about the message, such as the message type, size, and encoding (**B**). The message body contains the actual content of the message, which can vary depending on the type of message (**C**). OPC UA defines a set of services that allow clients and servers to interact with each other (**D**). These services are transmitted via the message body and provide functionality for various operations, such as reading and writing data, subscribing to events, browsing the server's address space, and managing sessions.

Table 1. OPC UA Packet Structure.

Secure Channel Layer (A)	optional
Message Header (C)	fixed size
Message Type	4 bytes
Message Size	4 bytes
Secure Channel ID	4 bytes
Security Flag	4 bytes
Additional Header	variable size
Message Body (D)	variable size
ReadRequest/ReadResponse (E)	variable size

Every OPC UA handshake follows the request and response pattern, as shown by the read operation in Table 2. Besides the requested (e.g., NodesToRead) or transmitted data (e.g., Results), OPC UA packets carry the request handle located in the header, a unique identifier assigned to a client's request message when communicating with a server. It correlates a request (see Table 2a) and a response (see Table 2b) within a session. The request handle serves three main purposes. First, it allows the client to match the response received from the server to the original request. Second, OPC UA supports asynchronous communication, where a client can send multiple requests to a server without waiting for responses. Third, in case of errors or exceptions during processing, the server includes the request handle in the error response.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

Table 2. Request and Response Headers.

(a) OPC UA Read Request

REQUEST HEADER	
Type ID	4 BYTES
Request Handle	4 BYTES
Timestamp	8 BYTES
NODESTOREAD	VARIABLE

(b) OPC UA Read Response

RESPONSE HEADER	
Type ID	4 BYTES
Request Handle	4 BYTES
Timestamp	8 BYTES
RESULTS	VARIABLE

2.3. Related Work

Network data for process mining is a burgeoning research area gaining significant attention (Table 3). Existing studies predominantly use simulated network data from tools like Wireshark[1] and can be categorized into two event log generation techniques: rule-based and model-based. Using network data as input for process mining is a burgeoning research area gaining significant attention (Table 3). Existing papers predominantly use simulated network data and capture the network traffic with tools like Wireshark. However, we find two distinct event log generation techniques: rule-based and model-based.

Table 3. Related works on network data-based process mining.

Reference	Input data	Log generation	Automation	Model
Wakup & Desel [12]	Simulated	Rule-based	◑	Petri net
Engelberg et al. [10]	Simulated	Rule-based	◑	BPMN
Hadad et al. [11]	Simulated	Model-based	◑	Event log
Apolinário et al. [13]	Simulated	Model/rule-based	●	BPMN
Lange & Möller [14]	Simulated	Model-based	●	BPMN
Lange et al. [15]	Simulated	Model-based	●	BPMN
Empl et al. [16]	Simulated	Rule-based	◑	Petri net
Our paper	Real world	Rule-based	◑	BPMN

◑ semi-automated ● fully automated

Rule-Based Techniques

Rule-based techniques transform captured network traffic into structured event logs through predefined rules, necessitating manual rule definition beforehand. For instance, Wakup and Desel [12] employ filter TCP dumps with predefined rules and use TCPLog2Eventlog for extraction. Engelberg et al.[10] focus on HR recruitment, applying the heuristic miner to capture network data for business processes. Apolinário et al.[13] introduce FingerCI, combining both techniques for ICS model construction.

Model-Based Techniques

Model-based techniques generate event logs or process models from network traffic data, typically requiring no human intervention through unsupervised learning. For instance, Hadad et al.[11]

[1] https://www.wireshark.org/

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

propose unsupervised learning for event log generation, addressing challenges in activity recognition from network data. Lange et al.[15] introduce MONA, deriving workflows directly from network data without generating event logs.

In contrast, our paper contributes to explainable rule-based event log generation and process discovery, focusing on real-world OPC UA network data captured from a manufacturing company's end-of-line business process. Moreover, we generate event logs without isolating processes and derive processes without relying on inexplicable machine learning techniques.

3. OPC UA Process Discovery Method

To address the lack of a structured approach for process discovery from OPC UA network data, we develop this method in this paper. Process discovery typically involves obtaining data from running processes, generating event logs, and mining processes from these logs [17]. The challenge lies in abstracting multiple low-level network events into a high-level event log [18]. Following the design science research approach of Hevner et al. [19], we develop an IIoT-specific artifact based on CRISP-DM (Cross Industry Standard Process for Data Mining) [20], as illustrated in Figure 1. Further details on the individual phases follow. Before starting, it is crucial to determine the scope: the target systems or components (*which?*), the technique, frequency, and timing (*how?*), the stakeholders (*who?*), and the desired outcome of the discovery (*what for?*). In addition, suitable metrics must be defined to measure whether the scope has been achieved. For example, which data should be used for the event log, or is there already a process model to be compared against the output? However, the stakeholders involved should document and agree on these metrics to ensure the success [21].

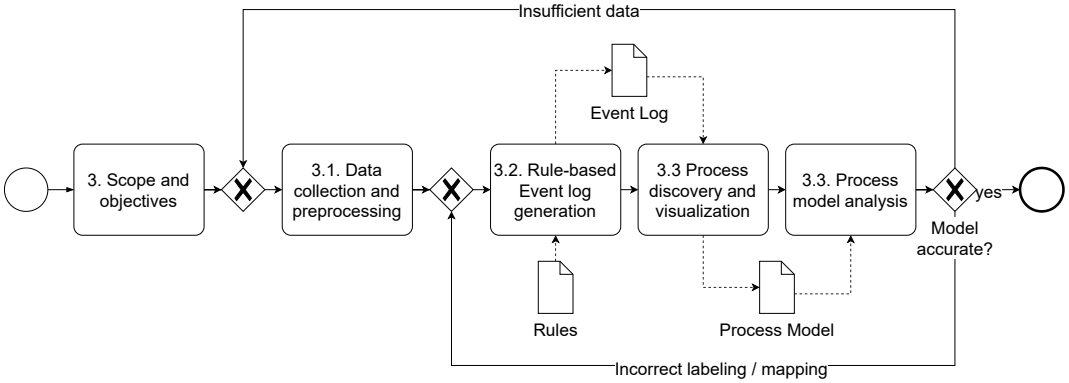


Figure 1. Generic process discovery approach in the IIoT.

3.1. Data Collection and Pre-Processing

Collect

Once the scope and objectives have been defined, we recommend using a passive data collection technique (network sniffing) instead of an active one, as it does not affect the operational processes and aligns with IIoT's high availability requirements. Passive recording is feasible using appropriate hardware (e.g., a switch with port mirroring) and software (e.g., Wireshark). Regardless of the hardware and software in use, the collected data's quality (e.g., completeness or encryption) is crucial. Competing the large data volume, filtering rules (e.g., on ports) ensure alignment with the predefined scope, but when recording an initial snapshot, a full capture is recommended, pushing the understanding of the network further. Last, as the PCAP format might be difficult to handle, it can be transformed into human-readable formats (e.g., XML or JSON).

Understand

Before pre-processing the data, the data analyst must understand the data's context, e.g., by collecting additional information, such as existing process models, descriptions, expert interviews,

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

asset inventories, or site visits. Afterward, it is crucial to understand the collected data [20]. Within the IIoT, this includes gaining insight into the network topology, IP addresses, ports, or protocols. After resolving duplicates, the data analyst can deeply dive into the packets' structures to identify data of interest, such as the case ID for subsequent event logs. Different paths lead to Rome, so visualizing information (e.g., social network diagram) may also assist before heading to the data pre-processing.

Pre-Processing

Network data is selected based on scope and objectives. Iteratively approaching the scope and objectives will lead to the desired outcome. Data analysts can assess the model's quality at each iteration by filtering less data and iteratively refining the selected data for the event log. Event log generation may involve aggregating multiple packets to form activities, especially in client-server architectures. In OPC UA, requests and responses can be matched using the so-called requestHandle (see Algorithm 1). The algorithm generates activities from low-level request-response events. Enriching activity names with human-readable labels ensures understandable process models. For example, if information on the function of a machine is available, replace the IP address and port with this information to increase readability.

Algorithm 1 Activity generation.

```
Require: opcua_packets
Ensure: matches
procedure MATCH_PACKETS(opcua_packets)
    req ← empty list
    res ← empty list
    for all packet in opcua_packets do
        ip ← packet.ipdst
        port ← packet.portdst
        time ← packet.time
        if packet.ttype == "MSG" then
            FIND_CONNECTION_TYPE(obj)
            if header then
                nodes ← GET_NODE_STRINGS
                if "RequestHeader" then
                    req.app(time,ip,port,nodes)
                else
                    res.app(time,ip,port,nodes)
                end if
            end if
        end if
    end for
    matches ← MATCH_REQUESTS(req, res)
    return SORT_BY_TIME(matches)
end procedure
```

3.2. Rule-Based Event Log Generation

After generating activities from filtered, aggregated, and labeled network packets, the next step is identifying each activity's process instance and generating an event log. Mandatory information of an event log includes the 1) case ID, 2) timestamp, and 3) activity name. The case ID is a unique identifier that identifies a process instance or a run and is assigned to all activities involved. Timestamps indicate the event's occurrence and provide information on sequential or parallel activities. While an event can have different activity names, non-uniqueness within the same process run is permitted. Optional information complements an event log, including information about the resource, e.g., the name of the actuator executing an activity. In the IIoT, we find physical processes and machines handing over products. We can refer to each product traveling through this process as a process instance, while a new process instance is created when it first appears in the network traffic. Each product has a unique identifier, ideal as a case ID. As not every packet carries the product identifier, pseudocode in Algorithm 2 details the event log generation based on the case ID assignment. Automatically assigning activities ensures consistency over the process and the event logs. Experimenting with different case IDs (in the case of appropriate candidates) further allows the comparison throughout the event logs.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

Algorithm 2 Event log generation.

```
Require: matches
Ensure: cases
procedure ADD_CASE_ID(matches)
    prod_id ← ITEM_ID
    ip_to_case_id ← empty dictionary
    cases ← empty list
    for all match in matches do
        time ← match.time
        ip ← match.ip
        port ← match.port
        nodes ← match.nodes
        case_id ← None
        for all node in nodes do
            if prod_id in node.keys() then
                case_id ← node(prod_id)
                ip_to_case_id[ip] ← case_id
                break
            end if
        end for
        case_id ← ip_to_case_id.get(ip)
        cases.app(time, ip, port, case_id)
    end for
    return cases
end procedure
```

3.3. Process Discovery, Visualization and Analysis

The derived event log is the basis for applying process mining techniques and enables identifying and visualizing processes and process instances. For example, process mining discovery techniques include heuristic, alpha, and inductive miners, which produce different outcomes (e.g., BPMN or Petri net). Each outcome, when visualized, shows different process perspectives. A direct follows graph creates an overview of process instances and dimensions (e.g., frequency or performance). The BPMN notation (and notably extended options with context-specific variables) focuses more on business processes [22].

Data analysts can interpret the results regardless of the notation or process mining technique used. This way, deviations between the discovered and target processes can be identified, e.g., bottlenecks. Visualizations also help to uncover optimization potential. For informed decision-making, stakeholders can enrich the process models with expert knowledge if required. An inaccurate model (e.g., inadequate data or pre-processing) may result in returning to an earlier phase.

4. OPC UA Mining Implementation

This section introduces the Python implementation details of the event log generation using the OPCUAPACKETANALYZER class. This analyzes OPC UA network packets, extracting relevant information and generating event logs. The implementation is available on GitHub[2]. It loads OPC UA data from a JSON file, extracts data from packets at various ISO/OSI layers, matches request/response handles in OPC UA packets, and generates CSV event logs. These event logs serve as the foundation for subsequent analysis.

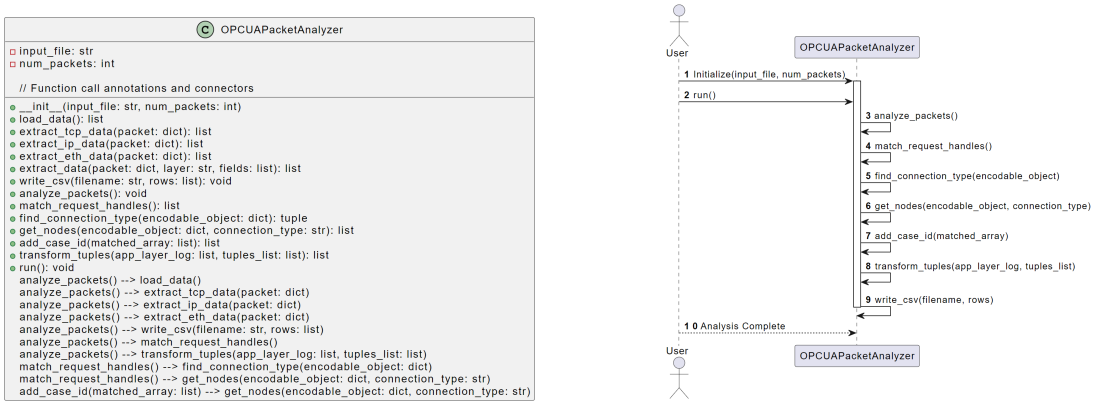
4.1. Software Design

In Figure 2, we present a visual representation of the OPCUAPACKETANALYZER class structure and relationships (see Figure 2a). In the class diagram, we can derive the structure of the OPCUAPACKETANALYZER class, including its attributes and methods. The relationships between methods are depicted to provide a high-level overview of how they interact. A sequence diagram depicts the interactions and flow of control between objects and actors. In our case, we use a sequence diagram to illustrate how the OPCUAPACKETANALYZER class is invoked and how its methods interact (see Figure 2b). The sequence diagram shows the actions when users interact with the OPCUAPACKET-

[2] https://github.com/philipempl/opcua-mining

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

ANALYZER. The user initializes the class, runs the analysis, and triggers various internal methods to perform specific tasks, which we detail in the following.



(a) Class diagram. **(b)** Sequence diagram.

Figure 2. Implementation design of the OPCUAPACKETANALYZER class.

4.2. Implementation Details

We provide detailed explanations of key methods and functionalities of the OPCUAPACKETANALYZER class in the following:

Entrypoint. The analyze_packets() method is the entry point for event log generation, orchestrating data extraction, request handle matching, case ID assignment, and event log generation. It structures OPC UA communication for process mining and analysis.

Data Loading. The load_data() method loads OPC UA communication data from a Wireshark JSON file, ensuring availability for subsequent methods.

Data Extraction. Utilizing extract_tcp_data(), extract_ip_data(), and extract_eth_data(), this step extracts relevant data from packets at various ISO/OSI layers.

Request Handle Matching. The match_request_handles() method matches request handles in OPC UA packets, establishing relationships between requests and responses and creating activities.

Event Log Generation. The write_csv() method generates CSV event logs from extracted data for process mining or visualization.

Case ID Assignment. The add_case_id() method assigns case IDs to matched arrays of OPC UA packets based on specific keys, facilitating subsequent process mining techniques.

5. Use Case: End-Of-Line Process

In this section, we apply the method from Section 3 to a real industrial use case to showcase its application and relevance to real-world OPC UA network data. We thereby rely on the implemented OPCUAPACKETANALYZER class. The scenario includes an automotive supplier's end-of-line process, which includes robotic inspections, laser engraving, and cleaning processes. We travel through each phase of the method and discuss corresponding actions. The dataset includes activities from four machines and a central process control system and provides a real-time snapshot of the process. Through this, we aim to investigate the feasibility of identifying IIoT business processes by mining OPC UA data.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

5.1. Data Collection and Pre-Processing

Collect

We collect the data in real-time using a Raspberry Pi[3] connected to the network switch responsible for internal network communication. Using port mirroring, the Raspberry Pi captures and stores 30 minutes of network traffic in plain text on a USB hard disk. This created a snapshot of the network communication during live operation in PCAP format.

Understand

Initially, we attempt to directly read the PCAP file using the pyshark library[4] but face limitations, e.g., no support for OPC UA. We then export the network traffic with Wireshark to JSON, which requires us to specify relevant OPC UA service ports. The Wireshark OPC UA extension aids packet interpretation, enabling the creation of the network structure (see Figure 3a) for an overview. We identify the central network's IP address as .31 for the Process Control System (PCS) server. Among 24,445 OPC UA packets, we find 24,421 OPC UA message packets and 24 OpenSecureChannelRequest packets, which we do not further analyze. In total, 9,244 packets have been sent by the PCS, the PCS has received 9,247 packets, and 2,965 were sent to the protocol server. The network data reveals that the PCS requests machine information through read and write requests. Publish and response packets lack production-relevant content, possibly due to an OPC PubSub-based notification system. Publish request packets originate from the PCS or the log system, addressed to the cleaning, conveyor, and test systems, resulting in publish response packets. While those packet timestamps lack unique polling information, we exclude publish and subscribe packets from the event generation process.

Pre-Processing

Following the contextual analysis of the packets, we initiate the pre-processing. First, we exclude communication relationships containing the protocol server and focus on OPC UA packets between machines and the PCS. We apply the request handle matching algorithm to create activities by aggregating OPC UA packets having the same request handle. To ensure better human reading, we assign labels using IP addresses with device type mapping (*IP address:Label*). In Figure 3b, these labels, like *.31:PCS server*, serve as activity names in event logs, enhancing process model readability.

(a) Network communication frequency based on IP addresses (green) and ports (grey).

(b) Human-readable machine labeling of IP addresses.

Figure 3. Representation of visualizations.

[3] https://www.onlogic.com/eu-en/computers/industrial/fanless/factor-200/
[4] https://github.com/KimiNewt/pyshark

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

5.2. Rule-Based Event Log Generation and Process Mining

Next, we generate the event logs for our use case. We find a product identifier (CANPRODUCE.ITEM_ID) in the network traffic, allowing us to run the event log generation algorithm. This algorithm assigns the product identifier to each process that has not yet been attributed a case ID. It is important to note that the choice of case ID depends highly on the specific use case, which emphasizes the need for a thorough understanding of the available data. The event log is then written to CSV files. After creating the event log, we run process mining discovery techniques on our event log. For visualization purposes, Appendix A shows the directly follows graph of the event log. As the Alpha miner, Heuristic miner, and Inductive miner rely on different algorithms, the results vary. We discuss each process model with the process experts and assess whether it reflects the reality. However, all of them somehow reflect the reality. We observe, that process experts have problems to assess low-level network events.

6. Evaluation

As already shown that mining processes from OPC UA network data is feasible, we aim to assess the scalability and quality of our approach. To assess mining capabilities and model quality in the OPC UA context, we implement experiments within a Jupyter notebook, available on GitHub[5]. Using a MacBook Pro 2021 with an Apple M1 Pro chip, 8 cores, and 16 GB of memory, we employ experiments on the OPCUAPACKETANALYZER class, analyzing OPC UA packets to generate event logs. This class extracts data from different ISO/OSI stack layers, generating logs for process mining algorithms. Performance evaluation involves generating event logs of varying sizes to understand scalability. Quality metrics such as replay fitness, precision, generalization, and simplicity gauge model performance. Collaboration with a process expert validates real-world accuracy and relevance, enriching results and fortifying practical implications.

6.1. Results

Event Log Generation Performance

Our experimental setup explores OPC UA packet processing performance by incrementally analyzing varying loads. Key metrics include time, CPU, and RAM usage. We start with 1,000 packets, increasing by 1,000 in each run until dataset exhaustion. Visualizing the results in Figure 4, packet analysis time shows a quadratic relationship with packet count, confirmed by a polynomial regression (black line). As expected, processing time increases with more packets. CPU and RAM usage (green and magenta lines) remain consistent, with occasional RAM spikes and steady CPU usage. Results indicate a significant computational demand increase with rising packet count. The polynomial regression in the experimental setup is:

$$T(p) = 1.4840 \times 10^{-8} p^2 - 1.1514 \times 10^{-6} p + 0.0728$$

The polynomial regression trendline offers a predictive insight, where $T(p)$ is the time taken, and p is the number of packets, suggesting that for larger data sets, resource allocation should be planned judiciously to ensure optimal performance. For instance, generating an event log for 1,000,000 OPC UA packets requires approximately four hours, which is appropriate as it is the initial step towards process mining and deriving process models, which is relatively fast. The observed CPU/RAM usage trends further emphasize the importance of efficient resource management, mainly when dealing with substantial packet loads.

[5] https://github.com/philipempl/opcua-mining

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

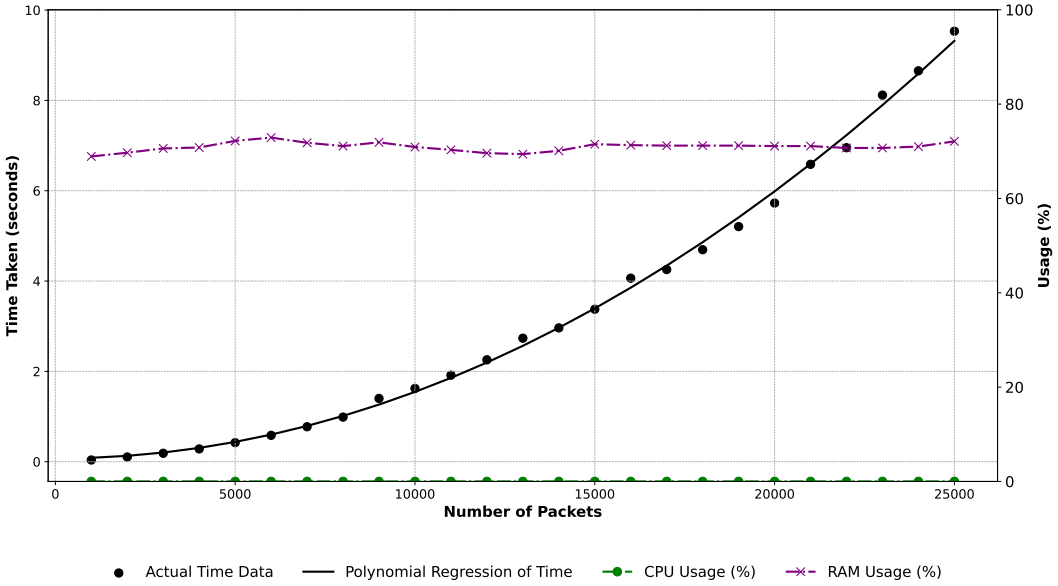


Figure 4. Performance analysis: time taken for analysis, CPU and RAM usage.

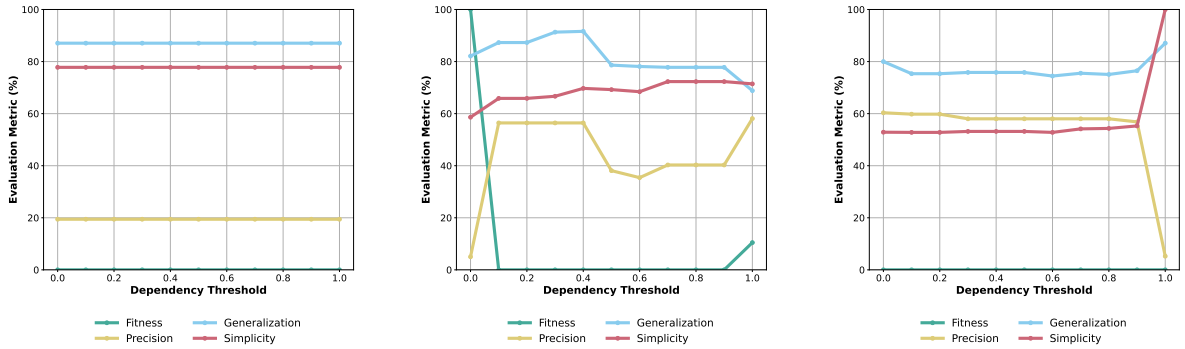
Process Model Quality

Within our setting, we compare the quality of three process discovery algorithms, the Alpha miner, Heuristic miner, and Inductive miner, across varying dependency thresholds on our OPC UA data (see Figure 5). This threshold ranges from 0 to 1 and represents the minimum required dependency between activities to establish a causal relationship. Note that the Alpha miner does not rely on dependency thresholds, resulting in horizontal lines. For the *Alpha miner* (Figure 5a), it consistently shows 0% fitness, indicating poor alignment with the event log. Precision, generalization, and simplicity metrics remain stable but at low values (19.4%, 87.1%, and 77.8%, respectively). This consistency indicates its limited adaptability. The *Heuristic miner* (Figure 5b) exhibits varied performance. At a threshold of 0, it achieves 100% fitness, declining sharply at higher thresholds. Precision peaks at 56.4%, with an upward trend in generalization. Simplicity fluctuates but remains within the mid-60% to mid-70% range. The *Inductive miner* (Figure 5c) shows intriguing results. At lower thresholds, it has 0% fitness, comparable to the Alpha miner. Precision starts at 60.3% and declines with higher thresholds. Generalization and simplicity fluctuate but within a tight range. In summary, the Heuristic miner is highly adaptable but the Inductive Miner offers a balanced performance in precision, generalization, and simplicity. The Alpha miner, while stable, lacks alignment with the log. Considering these nuances is vital for selecting an optimal miner in practical applications.

Operational Insights

We also gain insights from the continuous evaluation of processes through our industrial collaboration. An initial statement from the process expert is that “he would never have believed that we could get so close to the real process using only network data”, which led to an internal rethink about the importance of network data for operational benefits. In addition, within the analysis of the network data, we identified further potential for process optimization. For example, in addition to OPC UA, we discovered that a server regularly searches for printers in the network, which reduced the performance of the network. There were also indications of typing errors in naming and variations in variables, which were rectified.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1



(a) Alpha miner. **(b)** Heuristic miner. **(c)** Inductive miner.

Figure 5. Process discovery algorithms' quality with varying thresholds.

6.2. Discussion

Limitations

While our research highlights the benefits of process mining in OPC UA network data, we acknowledge limitations that may impact the generalizability of our findings. First, our paper assumes the availability of network data in plain text. Encryption is sometimes used in real-life scenarios, which could conceal important information. Secondly, the method relies on a product ID for tracing and differentiating process instances. In cases of absent or inconsistent identifiers, mined process accuracy and completeness may be limited. Lastly, our dataset, covering only 32 unique process instances, may not represent the diversity of processes in more complex industrial settings, affecting the robustness and applicability of our insights.

Scientific Impact

In the evolving realm of process mining, our paper marks a paradigm shift, breaking away from conventional approaches. We pioneer the application of process mining to OPC UA data, showcasing its feasibility and effectiveness while highlighting key challenges, notably in data availability. This revelation emphasizes that datasets suitable for process mining are more extensive than previously believed. Our findings have broad applicability, such as in cybersecurity, where process models can enhance network intrusion detection or ensure compliance[22]. Additionally, our insights into OPC UA processes offer valuable nuances for future benchmarking studies.

Practical Impact

In process mining, decoding OPC UA network data holds transformative potential for gaining insights into operational processes. While our models currently exhibit qualitative limitations, they already reflect real-world end-of-line process behavior. Increased data volume would facilitate more meaningful models. Process experts, though adept at a macro-level understanding, may lack granularity in network-level events. Precisely determining process starting points is critical for aligning mapped processes with expert understanding. Our 30 minutes capture suggests that a one-week snapshot can reveal essential details for informed analysis. By bridging the gap between high-level process knowledge and complex network traffic patterns, organizations can fully leverage the potential of process mining.

7. Conclusion

Our research taps into the rich potential of network data in the IIoT, an area that has not been fully explored for generating event logs and uncovering business processes. To the best of our knowledge, we are the first to introduce a method that reveals IIoT processes based on (OPC UA) network traffic data. Our method not only advances academic research, allowing for more detailed comparisons and improvements (like benchmarking), but it also shows practitioners the real value of network traffic

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

data. We developed an open-source prototype that represents a significant shift in process mining, offering a transparent and understandable way of mining OPC UA network data. Despite facing challenges like network encryption and working with a relatively small dataset, our findings are promising. They reveal that our process models accurately reflects a real-world use case at a quite high quality with relatively good performance. In our discussion, we emphasize the importance of using larger datasets for more precise results. We are excited to follow future research in this area, confident that network traffic data is poised to unlock new opportunities in process mining and beyond.

Acknowledgments: This work is funded by the "Bavarian Ministry of Economic Affairs, Regional Development and Energy" within the project Security Iiot pRocEss Notation (SIREN).

Appendix A. Directly-Follows Graph of the End-of-Line Process

The directly follows graph (DFG) visualizes the sequence and frequency of events or activities in a given process. In this specific DFG, nodes represent distinct activities, such as "PublishRequest" and various "ReadRequest" operations with associated parameters. Directed edges between nodes signify the order in which these activities occur. For instance, an edge from "PublishRequest" to a "ReadRequest" indicates that the "PublishRequest" activity directly precedes the "ReadRequest" activity in the process sequence. Furthermore, numerical annotations on the edges, like 1588 or 1505, represent the sequence frequency, denoting how many times another directly followed one activity in the observed data. The nodes' parameters detail transferred data within the respective OPC UA tuples. This DFG provides insights into the common paths and patterns of the end-of-line process but is still cluttered.

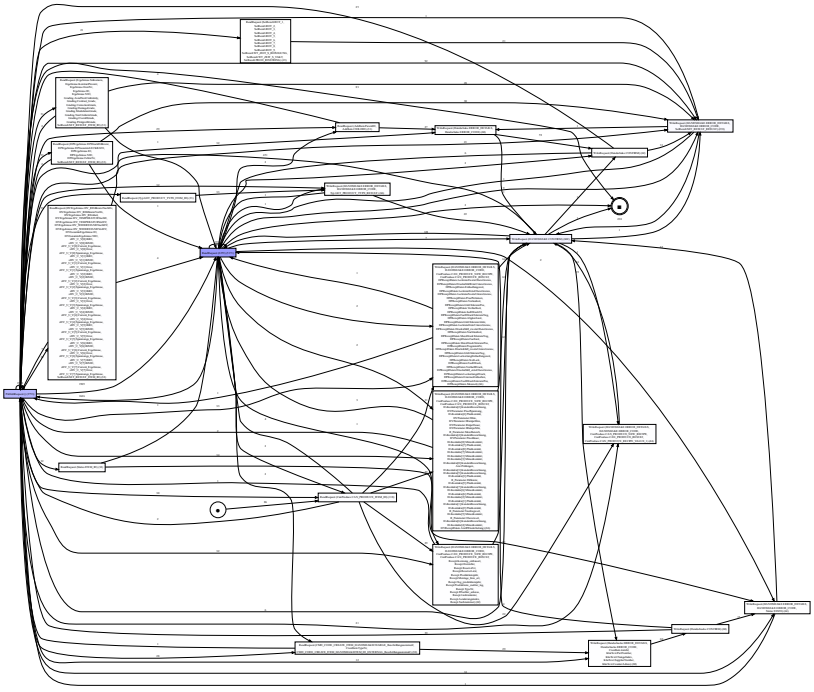


Figure A1. Cluttered directly-follows graph of the end-of-line process.

References

1. Boyes, H.; Hallaq, B.; Cunningham, J.; Watson, T. The industrial internet of things (IIoT): An analysis framework. *Computers in industry* **2018**, *101*, 1–12. doi:10.1016/j.compind.2018.04.015.
2. Hästbacka, D.; Barna, L.; Karaila, M.; Liang, Y.; Tuominen, P.; Kuikka, S. Device status information service architecture for condition monitoring using OPC UA. ETFA, 2015, pp. 1–7. doi:10.1109/ETFA.2014.7005141.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 May 2024 doi:10.20944/preprints202405.1589.v1

3. Shin, S.J. An OPC UA-Compliant Interface of Data Analytics Models for Interoperable Manufacturing Intelligence. *IEEE Transactions on Ind. Inf.* **2021**, pp. 3588–3598. doi:10.1109/TII.2020.3024628.
4. Schönig, S.; Hornsteiner, M.; Stoiber, C. Towards Process-Oriented IIoT Security Management: Perspectives and Challenges. Enterprise, Business-Process and Information Systems Modeling; Springer International Publishing: Cham, 2022; pp. 18–26.
5. van der Aalst, W.M. *Process Mining: Data Science in Action*; Springer, 2016. doi:10.1007/978-3-662-49851-4.
6. Bertrand, Y.; De Weerdt, J.; Serral, E. A bridging model for process mining and IoT. ICPM. Springer, 2021, pp. 98–110. doi:10.1007/978-3-030-98581-3_8.
7. Seiger, R.; Franceschetti, M.; Weber, B. An interactive method for detection of process activity executions from iot data. *Future Internet* **2023**, *15*, 77. doi:10.3390/fi15020077.
8. Mangler, J.; Grüger, J.; Malburg, L.; Ehrendorfer, M.; Bertrand, Y.; Benzin, J.V.; Rinderle-Ma, S.; Serral Asensio, E.; Bergmann, R. DataStream XES extension: embedding IoT sensor data into extensible event stream logs. *Future Internet* **2023**, *15*, 109. doi:10.3390/fi15030109.
9. Dunzer, S.; Zilker, S.; Marx, E.; Grundler, V.; Matzner, M. The Status Quo of Process Mining in the Industrial Sector. WI. AISeL, 2021, pp. 629–644. doi:10.1007/978-3-030-86800-0_43.
10. Engelberg, G.; Hadad, M.; Soffer, P. From network traffic data to business activities: a process mining driven conceptualization. BPMDS. Springer, 2021, pp. 3–18. doi:10.1007/978-3-030-79186-5_1.
11. Hadad, M.; Engelberg, G.; Soffer, P. From Network Traffic Data to a Business-Level Event Log. BPMDS. Springer, 2023, pp. 60–75. doi:10.1007/978-3-031-34241-7_5.
12. Wakup, C.; Desel, J. Analyzing a TCP/IP-protocol with process mining techniques. International Conference on Business Process Management. Springer, 2014, pp. 353–364. doi:10.1007/978-3-319-15895-2_30.
13. Apolinário, F.; Escravana, N.; Hervé, É.; Pardal, M.L.; Correia, M. FingerCI: generating specifications for critical infrastructures. SIGAPP, 2022, pp. 183–186. doi:10.1145/3477314.3507323.
14. Lange, M.; Kuhr, F.; Möller, R. Using a deep understanding of network activities for workflow mining. KI. Springer, 2016, pp. 177–184. doi:10.1007/978-3-319-46073-4_17.
15. Lange, M.; Möller, R. Time series data mining for network service dependency analysis. SOCO'16-CISIS'16-ICEUTE'16. Springer, 2017, pp. 584–594. doi:10.1007/978-3-319-47364-2_57.
16. Empl, P.; Böhm, F.; Pernul, G. Process-Aware Intrusion Detection in MQTT Networks. Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy (CODASPY '24); ACM: New York, NY, USA, 2024; p. 12. doi:10.1145/3626232.3653271.
17. Cook, J.E.; Wolf, A.L. Automating Process Discovery Through Event-Data Analysis. ICSE. ACM, 1995, pp. 73–82. doi:10.1145/225014.225021.
18. Mannhardt, F.; de Leoni, M.; Reijers, H.A.; van der Aalst, W.M.P.; Toussaint, P.J. Guided Process Discovery - A pattern-based approach. *Inf. Syst.* **2018**, *76*, 1–18. doi:10.1016/j.is.2018.01.009.
19. Hevner, A.R.; March, S.T.; Park, J.; Ram, S. Design Science in Information Systems Research. *MIS Q.* **2004**, *28*, 75–105. doi:10.2307/25148625.
20. Wirth, R.; Hipp, J. CRISP-DM: Towards a standard process model for data mining. Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining. Manchester, 2000, Vol. 1, pp. 29–39.
21. Mirza, M.N.; Pourzolfaghar, Z.; Shahnazari, M. Significance of Scope in Project Success. *Procedia Technology* **2013**, *9*, 722–729. CENTERIS / ProjMAN / HCIST, doi:10.1016/j.protcy.2013.12.080.
22. Hornsteiner, M.; Schönig, S. SIREN: Designing Business Processes for Comprehensive Industrial IoT Security Management. DESRIST. Springer, 2023, Vol. 13873, *LNCS*, pp. 379–393.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.