Industrial Network Topology Analysis with Episode Mining

Ankush Meshram

Vision and Fusion Laboratory
Institute for Anthropomatics
Karlsruhe Institute of Technology (KIT), Germany
ankush.meshram@kit.edu

Technical Report IES-2018-04

Abstract

Industrial network communication is highly deterministic as result of availability requirement of control systems in automated industrial production systems. This deterministic character helps with initial step of self-learning anomaly detection systems to detect periodic production cycle in industrial network communication. The methods for frequent episode mining in event sequences fits well to solve the challenge of production cycle detection for self-learning system. We encode the network communication events to serial and parallel episodes. Methods for discovery of frequent episodes in event sequences are briefly explained. These methods would be further adapted in future to our encoded network communication traffic to extract production cycle comprised of serial and parallel episodes.

1 Introduction

Industrial network communication enforces high availability requirement for automated industrial production systems. This results into deterministic and discrete communication behaviour between network components, and hence industrial control system components [Mes17]. The foremost task for self-learning anomaly detection in industrial networks is detecting the periodic synchronous communication resulting from the production cycle of the industrial

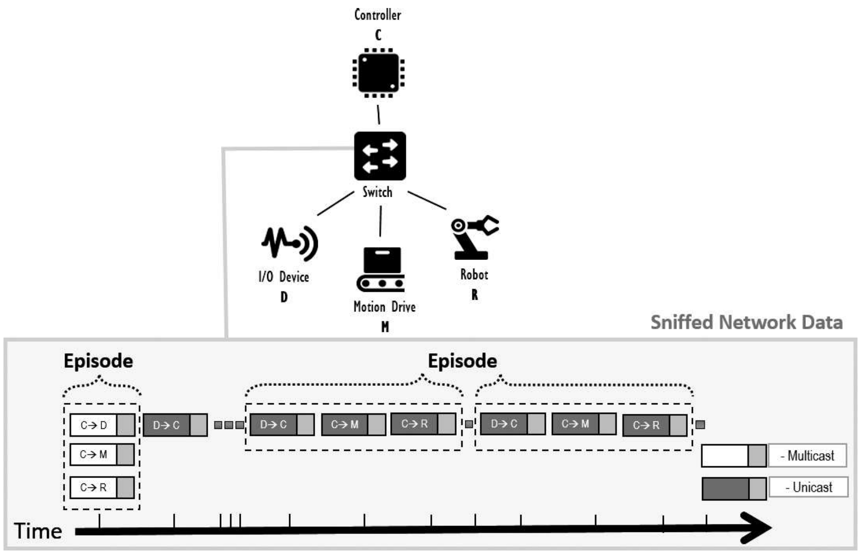


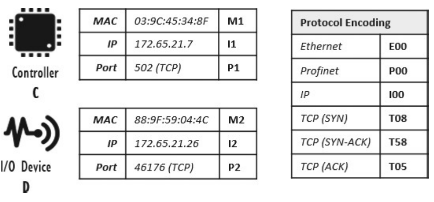
Figure 1.1: Example Drilling System.

system[MH16]. In order to do so the network communication traffic is sniffed and analyzed to detect the production cycle.

In this report, we propose the usage of Episode Mining paradigm to discover periodic patterns in the network traffic sequence [MTV97]. In the following sections, we begin with requirements for production cycle detection in industrial networks with a sample scenario in Section 2. Then, we map the industrial communication events to Episode and its variants in Section 3. We briefly explain the different methodologies for frequent episode mining.

2 Production Cycle Detection in Industrial Networks

For production cycle detection, we analyze the continuous sequential network communication traffic data. Our goal is to extract periodic patterns from the analyzed traffic while preserving the time order of communication events. Network


Controller
C
I/O Device
D

MAC	*03:9C:45:34:8F*	**M1**
IP	*172.65.21.7*	**I1**
Port	*502 (TCP)*	**P1**

MAC	*88:9F:59:04:4C*	**M2**
IP	*172.65.21.26*	**I2**
Port	*46176 (TCP)*	**P2**

Protocol Encoding	
Ethernet	**E00**
Profinet	**P00**
IP	**I00**
TCP (SYN)	**T08**
TCP (SYN-ACK)	**T58**
TCP (ACK)	**T05**

Figure 2.1: Mapping of component properties (MAC Address, IP Address and Port Number) and Protocol Flags to text variables.

communication at higher levels does not happen in isolation. For example, when OSI Layer 4 protocol TCP is used for communicating information from one node to another, the underlying Layer 3 and Layer 2 protocols also participate simultaneously. Hence, we need pattern extraction method which considers strict or partial ordering of simultaneous events.

We consider an example scenario of simple drilling system as depicted in Figure 1.1. The I/O Device (D) notifies PLC controller (C) that a metal block has arrived on conveyor belt driven by a Motor (M). The Controller then signals the Motor to run for a fixed time duration and move metal block under the Drilling Robot (R). After the fixed time has passed, Controller signals Robot to drill a hole in the metal block. This whole process repeats itself and is the production cycle we want to detect from network traffic, sniffed at Switch (S) without any further information about industrial setup.

We found a solution in the Episode Mining framework where our periodic patterns are called episodes. An episode is a partially ordered collection of events occurring together as defined in [MTV97]. We explain further in the next section discovery of Episode as the appropriate periodic pattern extraction method for production cycle detection in industrial network communication.

3 Episode Mining for Production Cycle Detection

To understand Episodes further we consider the communication between components Controller (C) and I/O Device (D) with respective information of MAC-address, IP-address and Port with type. For easier explanation, we encode the device information with text variables. MAC addresses are mapped to variables beginning with '**M**' and appropriate indexing. Similarly, IP addresses and Port are mapped to variables beginning with '**I**' and '**P**' followed by appropriate indexing respectively. Also we follow an encoding scheme for protocols through which these devices might communicate. The last two digit represents the state of stateful protocol. In particular, successful TCP-Handshake communication follows SYN, SYN-ACK and ACK. Figure 2.1 lists up the mappings for Controller and I/O Device with protocol encoding for Ethernet, Profinet and TCP communication protocols.

The observed network traffic is complex. Each packet contains information on communication occurring at multiple OSI Layers. For explanation we consider here only L2 to L4 communication and encode the communication contained in the packet. The higher layer communication doesn't happen in isolation as mentioned earlier. For example, in Figure 3.1, the third packet communicated between Controller and I/O Device, the encoding '**P1P2T08**' represents Port P1 of Controller initiated TCP-Handshake with Port P2 of Device, along with communication at L3 and L2 encoded as '**I1I2I00**' and '**M1M2P00**'. Events following a total order are called Serial Episode whereas these ones following trivial order are called Parallel Episodes. In Figure 3.1, { **M1M2P00**, **M2M1P00** } is a Serial Episode and { **P1P2T08 I1I2I00 M1M2P00** } is a Parallel Episode. In particular two events marked with thick-edged rectangle could be TCP SYN flooding attack as the TCP Handshake is initiated again.

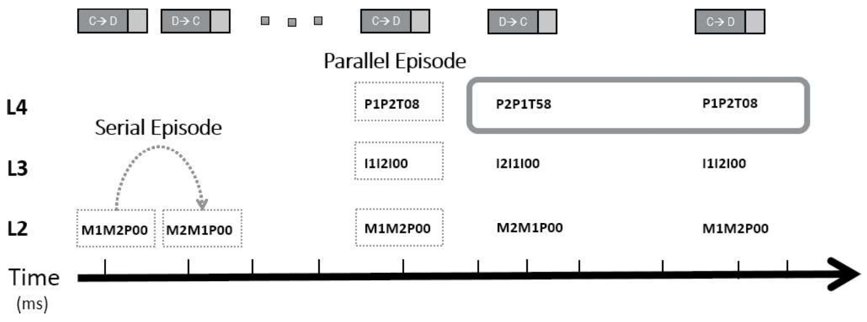


Figure 3.1: Mapped network traffic to Episodes.

More formally, episode mining can be described as follows with basic definitions. Given a set of event types E, an event is a pair (A,t) where $A \in E$ is an event type and t is an integer representing time of occurrence. An Event Sequence **s** is a triple (s, T_s, T_e) with ordered sequence of events A_i and T_s representing starting time, T_e representing ending time. The event occurring at ending time isn't included in the sequence.

$$s = \langle (A_1, T_1), (A_2, T_2), .., (A_n, T_n) \rangle$$
$$\text{s.t. } A_i \in E \forall i = 1, .., n \text{ and } t_i \leq t_{i+1} \forall i = 1, .., n-1$$
$$T_s \leq t_i < T_e \forall i = 1, .., n$$

The window of Figure 3.2 is a slice of event sequence and an event sequence can be considered as a sequence of partially overlapping windows. A window on an event sequence $\mathbf{s} = (s, T_s, T_e)$ is an event sequence $\mathbf{w} = (w, t_s, t_e)$ such that $t_s < T^e$ and $t_e > T_s$. The width of window **w** is defined by $width(\mathbf{w}) = win = t_e - t_s$. $W(\mathbf{s}, w)$ is a set of all windows **w** on event sequence **s** given window width win. The number of windows in $W(\mathbf{s}, win)$ is calculated as $T_e - T_s - win + 1$.

An episode α is a triple $(V, \leq, g)$ such that V is set of nodes, $\leq$ is partial order on V, and g is mapping associating each node with an event type i.e. $g : V \to E$. The size of episode α is defined by $|V|$. Episode α is parallel if relation $\leq$ is trivial i.e. $x \nleq y, \forall x, y \in V$s.t.$x \neq y$. Episode α is serial if relation $\leq$ is total order i.e. $x \leq y$ or $y \leq x, \forall x, y \in V$.

An episode $\beta = (V', \leq', g')$ is a *subepisode* of $\alpha = (V, \leq, g)$ i.e. $\beta \preceq \alpha$ if $\exists f : V' \to V$ s.t. $g'(v) = g(f(v)) \forall v \in V'$ and $\forall v, w \in V'$ with $v \leq' w$ also $f(v) \leq f(w)$. An episode α is *superepisode* of β i.e. $\beta \prec \alpha$ **if and only if** $\beta \preceq \alpha$.

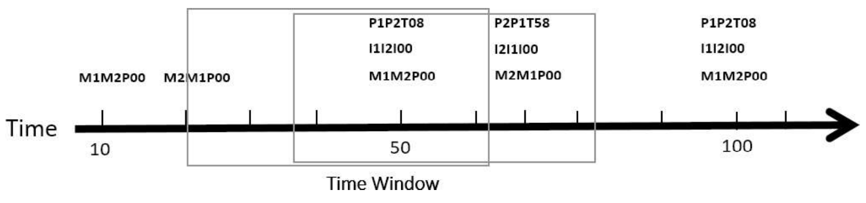


Figure 3.2: Windows for Episode Mining.

Algorithm 3.1 WINEPI Algorithm

Input: A set E of event types, an event sequence s over E,
a set ε of episodes, a window width win, and a frequency threshold min_fr
Output: The collection $\mathfrak{F}(s, win, min\_fr)$ of frequent episodes.
Method:

```
1:  $C_1 := \{\alpha \in \varepsilon \,|\, |\alpha| = 1\}$
2:  $l := 1$
3:  while $C_l \neq \varnothing$ do
4:      compute $\mathfrak{F}_l := \{\alpha \in C_l \,|\, fr(\alpha, s, win) \geq min\_fr\}$;
5:      $l := l + 1$;
6:      compute $C_l := \{\alpha \in \varepsilon \,|\, |\alpha| = l$ and for all $\beta \in \varepsilon$
7:                    such that $\beta \prec \alpha$ and $|\beta| < l$ we have $\beta \in \mathfrak{F}_{|\beta|}\}$
8:  end while
9:  for all $l$ do
10:     output $\mathfrak{F}_l$;
11: end for
```

An episode $\alpha = (V, \leq, g)$ occurs in an event sequence **s** = $(\langle (A_1, T_1), (A_2, T_2), .., (A_n, T_n) \rangle, T_s, T_e)$, if $\exists h : V \rightarrow \{1, .., n\}$ from nodes of α to events of **s** s.t. $g(s) = A_{h(x)} \forall x, y \in V$ with $x \neq y$ and $x \leq y$ we have $t_{h(x)} < t_{h(y)}$.

Frequency of an episode given an event sequence and window width is ratio of number of windows of the set of all windows on sequence in which it occurs to total number of windows on sequence. Frequency of an episode α in an event sequence **s** with window width win,

$$fr(\alpha, \mathbf{s}, win) = \frac{|\{w \in W(s, win)\} \,|\alpha \; occurs \; in \; w|}{|W(\mathbf{s}, win))|}$$

For a given frequency threshold min_fr, α is **frequent** if $fr(\alpha, \mathbf{s}, win) \geq min\_fr$.

The mining task is given an event sequence **s**, a window width win and a frequency threshold min_fr, discover all frequent episodes from a class ε of episodes - $\mathfrak{F}(\mathbf{s}, win, min\_fr)$. WINEPI algorithm 3.1 listed below discovers frequent episodes from an event sequence. Alternate approach to sliding window based

WINEPI is MINEPI algorithm. MINEPI is based on minimal occurrences of episodes where for each potentially interesting episode we find out the exact occurrences of the episode. One disadvantage to WINEPI is that MINEPI consumes significant amount of space. WINEPI and MINEPI work on offline dataset of sequences, MESELO [ALL$^+$15] is an algorithm for online frequent episode mining. MESELO stands for 'Mining frEquent Serial Episode via Last Occurrence' where authors introduced sophisticated data structure, *episode trie*, to store minimal occurrences of episodes compactly.

4 Summary

This report outlines using Episode Mining for Production Cycle Detection in Industrial Network Communication. A mapping for communication events to text variables is explained which are used later on to detect Serial and Parallel Episodes. Different methods for frequent episode mining are available. Implementation of MESELO algorithm modified for our goal is underway.

Bibliography

[ALL$^+$15] Xiang Ao, Ping Luo, Chengkai Li, Fuzhen Zhuang, and Qing He. Online frequent episode mining. In *2015 IEEE 31st International Conference on Data Engineering (ICDE)*, pages 891–902. IEEE, 2015.

[Mes17] Ankush Meshram. Deterministic industrial network communication: Fundamentals. In Jürgen Beyerer, Alexey Pak, and Miro Taphanel, editors, *Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory 2017. Proceedings*, pages 45–61, Triberg-Nussbach, July 2017. Karlsruhe: KIT Scientific Publishing, 2018 (Karlsruher Schriften zur Anthropomatik 34).

[MH16] Ankush Meshram and Christian Haas. Anomaly detection in industrial networks using machine learning: A roadmap. In *Machine Learning for Cyber Physical Systems 2016*, pages 65–72, Karlsruhe, Germany, September 2016.

[MTV97] Heikki Mannila, Hannu Toivonen, and A Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data mining and knowledge discovery*, 1(3):259–289, 1997.