Analyzing an OT network, what to expect

Saber Mhiri

***Abstract*—Due to the complication of the Operational Technology (OT) environment and the lack of documentation of the network field state, we provide this report that helps to understand the state of the current OT networks, the main protocols implemented and their problems as well as the approaches that can be used for monitoring these type of network.**

***Index Terms*—Operational Technology (OT), industrial network, network monitoring, industrial protocols.**

I. Introduction

When it comes to the OT environment, we usually imagine a set of production machines connected to Programmable Logic Controllers (PLCs), Supervisory Control and Data Acquisition (SCADA) systems, Distributed control system (DCS), computer numerical control (CNC) and various other control systems that oversee controlling the different production machines through industrial protocols often proprietorial to the PLC manufacturer. And based on this description, planning to monitor the network in the OT environment are expected to be a challenge due to the incoherence of its elements in terms of manufacturer and protocols of communication.

These issues are the reason behind the appearance of multiple standardization protocols in the market such as BACnet and OPC UA from OPC foundation.

Multiple solution providers offer products able to monitor the OT networks, collect logs as well as to detect vulnerabilities and attacks. These solutions usually specify the list of protocols they can support such as the Matrikon solution which leverages their standardized OPC protocol [1].

Although these products provide efficient solutions to the current OT networks, new more sophisticated solutions are appearing, where the network protocols have no effect on the performance of the solution [2].

In reality, when it comes to monitoring the industrial network, most of the previously mentioned problems can be resolved in most of the times without the need for implementing a standardization protocol.

Due to the introduction of industry 4.0, most of the network components in the IoT environment use protocols based on TCP/IP [3] such as Modbus, Profibus, s7comm, ...

This paper is composed of seven sections, in section 2 we describe the relation between the OT and IT environment and their characteristics. In section 3 we describe the components and architecture of a typical industrial network. In section 4, we focus on the protocols used in the industrial network and the reason behind it's diversity. In section 5, we present a solution to monitor the network traffic and collect the logs in the network. In section 6 we focus on analysing the traffic in the network, we present two main protocols that can be found in an industrial network, we discuss each of the protocols characteristics and architectures and we also present our testing environment, architecture and tools, as well as our results results. Finally in In section 7, we conclude this papers with our conclusions concerning monitoring OT networks.

II. A typical OT network vs. the reality

When it comes to OT and IT networks, we usually expect the networks to be segregated and protected as Figure 1 shows. But in reality, the two networks are increasingly merging together and the boundaries between OT and IT are getting blurry.

IT technologies are being implemented in OT environment rapidly. These IT technologies bring with them threats that the OT environment pieces of equipment are simply not ready for.

Due to this situation, the OT environment is in increasing need of protection and monitoring [4].

Fig. 1. IT vs. OT environment expectation

III. Industrial Network architecture

Typically, the OT architecture starts from the PLC, since PLCs represent the brains of the OT network. The northbound of the PLC is usually connected to IT hardware and serve as a source of information to the SCADA systems. The southbound of the PLC is used for communicating with the different sensors and industrial equipment in the network as described in [5].

As Figure 2 shows, in a typical industrial network, it's possible to have multiple different networks, where each network use a completely different communication model and protocols.

It's also worth mentioning that machine to machine communications are common in OT network, meaning that a PLC, for

example, can communicate directly with another PLC without the need to pass through a middle switch neither router.
Another example of these communications is the communications between PLCs and industrial material such as robotic arms.
Due to the use of different industrial communication protocols, these machine to machine communications is possible.

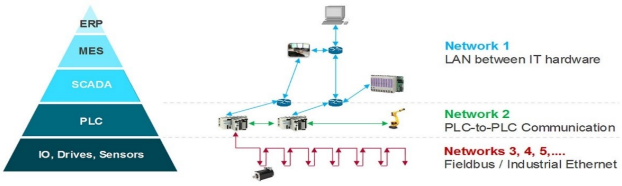

Fig. 2. Industrial network hierarchy

IV. PLC AND PROTOCOLS

Although there are multiple PLC providers in the market such as Siemens, Schneider, Allen Bradley ... these PLC solutions offer similar functionalities with a difference in term of supported protocols.
[6]discuss the PLC communication protocols and their different architectures.
The protocols supported by these PLCs can vary depending on the version of the PLC and provider, but in some cases, PLC providers may use a proprietary protocol such as the case of Siemens with their s7comm and s7comm plus protocols. Figure 3 shows the most commonly used protocols in the industry.

Popular Industrial Protocols		
Wired		Wireless
Fieldbus	**Industrial Ethernet**	802.15.4
Profibus	Profinet	6LoWPAN
Modbus	Ethernet/IP	Bluetooth/LE
DeviceNET	Ethernet/CAT	Cellular
CANOpen	Modbus TCP	LoRA
CC-Link		Wi-Fi
AS-I		WirelessHART
Interbus		ZigBee
ControlNet		

Fig. 3. List of commonly used industrial protocols

V. COLLECTING LOGS

In order to monitor the industrial network, we should start by collecting the logs of the machines in the network. To this end, several industrial networks may have historians implemented in their network. A Data Historian (also known as a Process Historian or Operational Historian) is a software program that records and retrieves production and process data by time; it stores the information in a time series database that can efficiently store data with minimal disk space and fast retrieval.
But in reality, the use of historian isn't a common practice, especially that SME factories are still struggling with updates needed in order to get included in the industry 4.0 revolution. On the other hand, the PLCs in the network do not save the logs unless programmed to do so. In the case of collecting the logs from the PLCs, we need to program each and every PLC in the network to save the logs and send it to our monitoring system. Most of the log collection solutions in the market although presented as solutions that collect logs from the machines, they actually monitor the traffic in the network and generate their own logs.
The process of generating logs through traffic is implemented in the Check Point and Nozomi SCADAguardian solutions. The process is straight forward, the product gets connected to a mirroring port of a switch in the network and start spoofing all the traffic passing through that network section.
As figure 4 shows all the blue boxes (log collectors) are directly connected to the switches in the network. These solutions monitor the traffic coming from the PLCs northbound and the rest of the IT network. Although these solutions are not placed in the lower level of the network, between the PLC and the different sensors of the factory, they do monitor and collect logs related to the PLC communication with the lower network level because most of the PLC communications go through the switches.

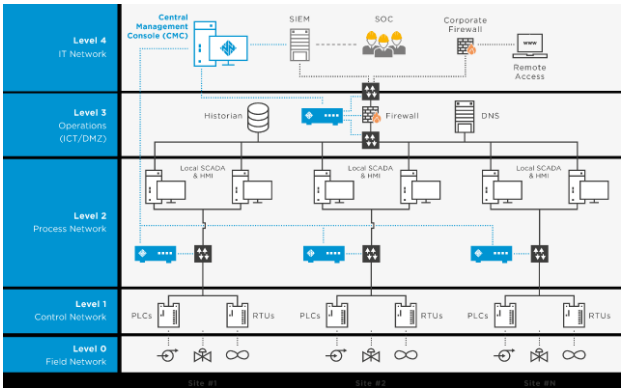

Fig. 4. Log collection through spoofing network mirror switches [2]

VI. ANALYZING THE TRAFFIC

When it comes to analyzing the traffic in an OT environment, We can face any of the previously mentioned protocols in Figure 3.
These protocols can also be classified into protocols over

TPC/IP and protocols that do not use the TCP/IP. Due to the emergence of the IT and OT environment, most of the protocols used in relatively modern factories are built over TCP/IP.
When it comes to TCP/IP protocols, there are mainly 2 classes of protocols, the open protocols such as Modbus and the proprietary protocols such as s7comm and s7comm plus.
In this work, we will focus on monitoring s7comm and s7comm plus, as well as the Modbus protocol.

A. Modbus

Modbus [7] is a serial communication protocol developed by Modicon published by Modicon® in 1979 for use with its programmable logic controllers (PLCs). In simple terms, it is a method used for transmitting information over serial lines between electronic devices. The device requesting the information is called the Modbus Master and the devices supplying information are Modbus Slaves. In a standard Modbus network, there is one Master and up to 247 Slaves, each with a unique Slave Address from 1 to 247. The Master can also write information to the Slaves [8].
Modbus is an open protocol, meaning that it's free for manufacturers to build into their equipment without having to pay royalties. It has become a standard communications protocol in industry and is now the most commonly available means of connecting industrial electronic devices. It is used widely by many manufacturers throughout many industries.
Modbus is typically used to transmit signals from instrumentation and control devices back to the main controller or data gathering system, for example, a system that measures temperature and humidity and communicates the results to a computer.
Modbus is often used to connect a supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems. Versions of the Modbus protocol exist for serial lines (Modbus RTU and Modbus ASCII) and for Ethernet (Modbus TCP).
Information is stored in the Slave device in four different tables. Two tables store on/off discrete values (coils) and two store numerical values (registers). The coils and registers each have a read-only table and read-write table.
Each table has 9999 values. Each coil or contact is 1 bit and assigned a data address between 0000 and 270E. Each register is 1 word = 16 bits = 2 bytes and has data address between 0000 and 270E.
Figure 5 shows the different fields a Modbus TCP packet has:

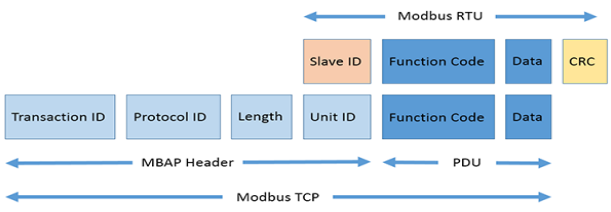

Fig. 5. Modbus TCP packet structure

- **Slave ID:**
  Each slave in a network is assigned a unique unit address from 1 to 247. When the master requests data, the seventh byte it sends is the Slave address. This way each slave knows after the first bite whether or not to ignore the message.
- **Function code:**
  The byte 8 sent by the Master is the Function code. This number tells the slave which table to access and whether to read from or write to the table.

Function Code	Action	Table Name
01 (01 hex)	Read	Discrete Output Coils
05 (05 hex)	Write single	Discrete Output Coil
15 (0F hex)	Write multiple	Discrete Output Coils
02 (02 hex)	Read	Discrete Input Contacts
04 (04 hex)	Read	Analog Input Registers
03 (03 hex)	Read	Analog Output Holding Registers
06 (06 hex)	Write single	Analog Output Holding Register
16 (10 hex)	Write multiple	Analog Output Holding Registers

Fig. 6. Function code

1) Simulations environment: Modbus protocol can be simulated by using a master and slave simulators. In our tests, we used the ModbusPal simulator and the Mbtget simulators.

- **ModbusPal:**
  ModbusPal is a project to develop a PC-based Modbus simulator. Its goal is to reproduce a realistic environment, with many slaves and animated register values. Almost everything in ModbusPal can be customized and controlled by scripts.
- **Mbtget:**
  A simple Perl script to make some Modbus transaction from the command line. This module gives you access to TCP and RTU version of this protocol, through the MBclient object.

2) Results: As shown in figure 7,the resulting packets of running the modbusPal with some slaves, and using the mbtget tool show Transaction identifier, Protocol identifier, Length, Unit Identifier, Function code, Reference number, and Bit count fields. Monitoring Modbus in the network proven to be an easy task since all the data are easily accessible. But on the other hand, using modbus in the industrial network is a Cybersecurity Risks that need to be dealt with.

B. S7comm

After searching for resources for monitoring and analyzing the s7 protocol [9], we came across a few projects that help to deal with this protocol. The open source communication library snap7 [10] which implement basic communication scenarios. This library also provides documentation of the basic structure of the s7 protocol.
Another project that helps to deal with the s7comm protocol is the s7 Wireshark dissector [11] which covers most of the protocol and its source code contains a lengthy list of protocol constants.
Due to the lack of official documentation of the protocol,

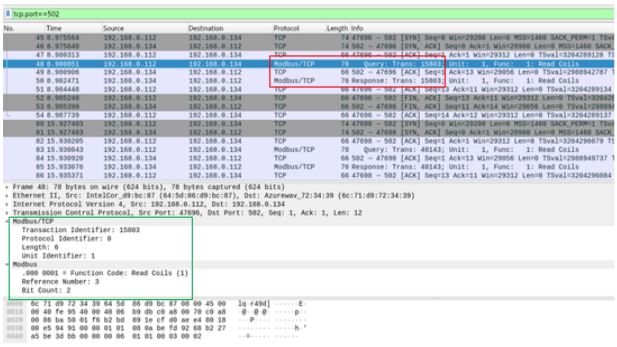

Fig. 7. Wireshark Modbus packet capture

official terminology does not exist when it comes to the S7 protocol.

When communicating with S7 devices there is a whole family of protocols, that can be used. In general, you can divide them into Profinet protocols and S7 Comm protocols. The latter is far simpler in structure, but also far less documented. The S7 Comm protocols are generally split up into two: The classic S7 Comm and a newer version unofficially called S7 Comm Plus. As figure 8 shows.

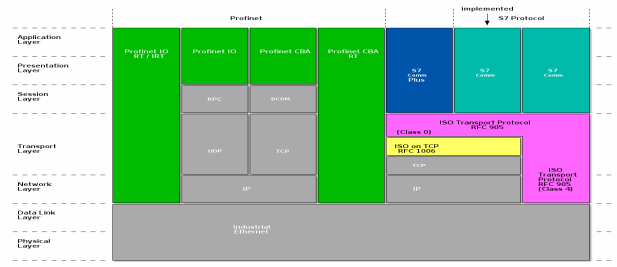

Fig. 8. Profinet vs. s7comm vs. s7comm plus protocols

- **The Siemens Communication Scenario:**
  When we talk about the ”S7 protocol” I refer to the Ethernet S7 communication that is mainly used to connect the PLCs to the PC stations (PG/PC - PLC communication) [12]. This is not to be confused with the different fieldbus protocols that the Siemens equipment use, such as MPI, Profibus, IE, and Profinet (which is an Ethernet-based protocol used to connect PLCs to IO modules, not the management protocol of the devices).
  Usually, the Siemens communication follows the traditional master-slave or client-server model, where the PC (master/client) sends S7 requests to the field device (slave/server). These requests are used to query from or send data to the device or issue certain commands. There are a few exceptions when a PLC can be the communication master, with $FB14/FB15$ the device can initiate GET and PUT requests to other devices.
- **The S7 PDU:**
  The S7 protocol TCP/IP implementation relies on the block-oriented ISO transport service. The S7 protocol is wrapped in the TPKT and ISO-COTP protocols, which allows the PDU (Protocol Data Unit) to be carried over TCP. Figure 8 shows the basic s7 communication packet fields. The S7 protocol is function/command-oriented

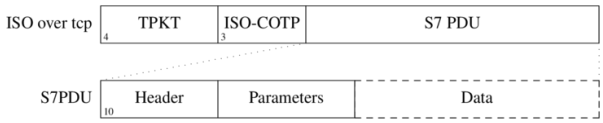

Fig. 9. s7 PDU packet

which means a transmission consist of an S7 request and an appropriate reply (with very few exceptions). The number of parallel transmissions and the maximum length of a PDU is negotiated during the connection setup. The S7 PDU consists of three main parts:

  - **Header:** contains length information, PDU reference and message type constant
  - **Parameters:** the content and structure greatly vary based on the message and function type of the PDU.
  - **Data:** it is an optional field to carry the data if there is any, e.g. memory values, block code, firmware data . . . etc
- **The problem with the s7comm plus:** This new version of the protocol is highly encrypted, making it impossible to parse the packets. There is no official documentation of the protocol and little to no information online about it.
  After searching for solutions, we found the s7comm plus Wireshark project which is a dissector for the s7comm plus protocol packets. But after testing it on our experimental environment, we found out that this dissector does not provide any useful information about the packets and the functionalities.
  Another source was the thesis published by Maik Brueggemann [13], but after analyzing it, we realized that the Information seems to be invalid or incorrect.

1) Simulations environment: Due to the lack of s7comm simulators, we created a small testing environment consisting of a real s7-1200 PLC, Wireshark, a control PC using TIA Portal v.15 and a network switch with the mirroring port enabled. We choice the s7-1200 PLC because of its capability of generating the s7comm plus traffic.

- **TIA Portal v.15**
  Totally Integrated Automation Portal (TIA Portal) provides you with unrestricted access to our complete range of digitalized automation services, from digital planning and integrated engineering to transparent operation. Through this tool, we managed to program the siemens s7 1200 PLC through the ladder logic, upload the program to the PLC, and monitor the activity of the PLC.
- **S7-1200 PLC**
  SIMATIC S7-1200 Basic Controllers are the ideal choice when it comes to flexibly and efficiently performing automation tasks in the lower to medium performance range. They feature a comprehensive range of technolog-

ical functions and integrated IOs as well as an especially compact and space-saving design.

- **Wireshark s7comm plus dissector**
  A plugin dedicated to s7comm plus packet analysis, built with Wireshark sources for 2.6.1.
- **Objectives**
  Monitor the traffic and create our own Logs in place of collecting the logs from the different TIA Portals in the factory network.
- **Topology**
  we're using the following topology:

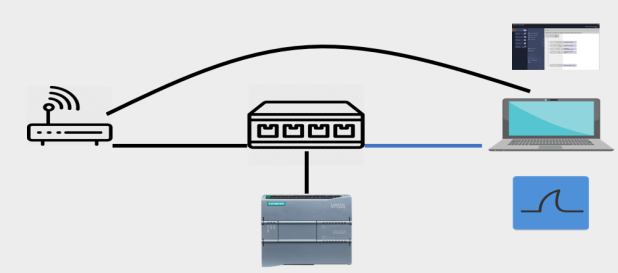

Fig. 10. The s7comm plus testing environment

  - The switch in the middle is connected to the PLC and the modem through ports 1 and 2.
  - The switch in the middle is connected to the PC through port 3 which is mirroring enabled (to monitor the traffic in the ports 1 and 2).
  - The PC is also connected through WIFI directly to the modem.
  - The PC is using Wireshark to monitor the traffic coming through the ethernet port connected to the mirroring port of the switch.
  - The PC is using TIA Portal to communicate with the PLC.
- **Scenarios**
  After setting up the topology and configuring the switch, we started testing with TIA Portal the following scenarios:
  - Scenario 1
    * Start TIA Portal.
    * Start capturing packets with Wireshark.
    * Connect the TIA Portal to the PLC.
    * Put the PLC to RUN.
    * Disconnect the TIA Portal.
    * Stop capturing packets with Wireshark and saving them.
  - Scenario 2
    * Start TIA Portal.
    * Start capturing packets with Wireshark.
    * Connect the TIA Portal to the PLC.
    * Put the PLC to Stop.
    * Disconnect the TIA Portal.
    * Stop capturing packets with Wireshark and saving them.
  - Scenario 3
    * Start TIA Portal.
    * Start capturing packets with Wireshark.
    * Connect the TIA Portal to the PLC.
    * Upload from the PLC the software and hardware configuration to the TIA Portal.
    * Disconnect the TIA Portal.
    * Stop capturing packets with Wireshark and saving them.
  - Scenario 4
    * Start TIA Portal
    * Start capturing packets with Wireshark
    * Connect the TIA Portal to the PLC.
    * Compile and download to the PLC the software and hardware configuration from the TIA Portal.
    * Disconnect the TIA Portal
    * Stop capturing packets with Wireshark and saving them.

2) Results and analysis's: Using the s7comm plus Wireshark dissector and following the ReadMe file, we incorporated the dissector into Wireshark v2.6.1.
The resulting package analysis is shown in Figure 10. The results show that the dissector is unable to provide useful information for monitoring the network.
The results show:

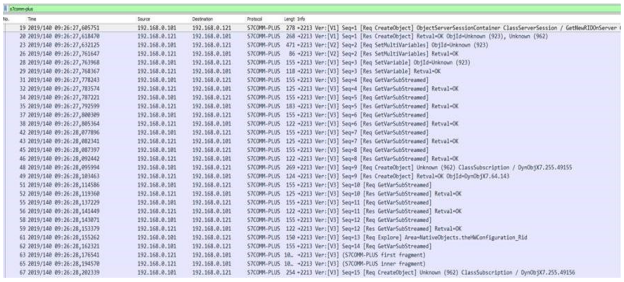

Fig. 11. s7comm plus capture with a Wireshark dissector

- First, the dissector results have no clear description of the nature of the action happening in each packet.
- Second, there is a lack of material describing the protocol s7comm plus.
- Finally, S7comm protocol has a better packet description because it's not encrypted.

VII. Conclusions

The OT environment is far more complicated to scan than what is expected mainly due to an abundant amount of protocols implemented. Although a lot of the protocols used are well documented and can be dissected, still some protocols are being left secrets, which make the process of monitoring the OT networks harder. Focusing on the IP level of the packet in order to map and monitor the networks seems to be the tendency of the different OT network monitoring solution in the market.

References

[1] https://www.matrikonopc.com/opc-drivers/opc-modbus/base-driver-details.aspx
[2] https://www.nozominetworks.com/

[3] https://www.rs-online.com/designspark/iot-industry-40-and-the-need-for-industrial-ethernet
[4] https://link.springer.com/chapter/10.1007/978-3-030-00024-011
[5] https://automationforum.co/basics-of-industrial-networking-architecture/
[6] https://library.automationdirect.com/plc-communications-coming-of-age/
[7] http://www.modbus.org/
[8] https://www.schneider-electric.com/en/faqs/FA168406/
[9] https://new.siemens.com/global/en.html
[10] http://snap7.sourceforge.net/
[11] https://sourceforge.net/projects/s7commWireshark/
[12] http://snap7.sourceforge.net/siemens$comm.html$