# A Profinet Simulator for the Digital Twin of Networked Electrical Drive Systems

Alberto Morato
*University of Padova and CMZ Sistemi Elettronici s.r.l.*
alberto.morato.3@phd.unipd.it

Stefano Vitturi
*National Research Council of Italy, IEIIT–CNR*
stefano.vitturi@ieiit.cnr.it

Tommaso Fedullo
*University of Padova*
tommaso.fedullo@phd.unipd.it

Giovanni Peserico
*University of Padova and Autec s.r.l.*
giovanni.peserico@phd.unipd.it

Federico Tramarin
*University of Modena and Reggio Emilia*
federico.tramarin@unimore.it

***Abstract*—Modern industrial manufacturing plants, especially those using coordinated electrical drives with strict timing requirements, make extensive use of real-time communication networks. These systems, typically, are based on various topologies, include diverse protocols, and connect devices from different manufacturers, which may make them difficult to study, plan and optimize. As a solution, the adoption of digital twins allows to simulate such systems under various operating conditions in a low-cost and zero-risk environment. In this paper we address the digital twin of a networked electrical drive system, focusing on the real-time communication network used to connect the drives. In particular, we describe the simulation model of Profinet IO RT Class 1, implemented as an extension of the INET library of OMNeT++. Moreover, we present the outcomes of the tests carried out on a prototype simulated network and compare them with those of the equivalent real one.**

***Index Terms*—Real-Time networks, Profinet IO, Digital Twin, OMNeT++.**

## I. Introduction

The complexity of modern manufacturing plants, their interconnection and modularity have a strong impact on the structure of the communication systems they adopt [1]. Thus, the execution of tests and measurements may result problematic due to the diverse network topologies and devices, often produced by different manufacturers. As a consequence, the exhaustive commissioning of such systems is very difficult to carry out.

A viable solution to these problems is the Digital Twin (DT) [2], a one–to–one representation of the real system, with all its functionalities in the digital world. The DT can be used to study the behavior of the physical system counterparts possibly under diverse set of operating conditions in a low-cost and zero-risk environment.

In the context of Industry 4.0, a DT requires very accurate simulation models of the involved communication systems, that take into account the environment in which they are used, as well as the types of traffic they handle. Indeed, the creation of such models is of paramount importance for real-time networks where multiple types of traffic may coexist contemporaneously, and where the network dynamic is often determined by the interaction among communication protocols, device configurations, and network topology.

In this paper, we address the digital twin of a networked electrical drive system, focusing on one of its most challenging aspects, namely the real–time communication system used to connect drives. Particularly, we present a Profinet IO RT Class 1 simulator based on the popular OMNeT++ network simulator.

Profinet uses protocols typical of the Information Technology, jointly with highly optimized real-time protocols. Over the years, it has been the subject of many research activities, for example about its performance [3], [4], or alternative scheduling algorithms [5], just to mention a few. Nonetheless, its complexity, as well as the lack of legacy simulation tools, can sometimes make it difficult to study, plan and optimize networks based on this communication protocol.

The benefits deriving from the adoption of the proposed DT can be mainly found in three different phases of a plant lifetime:

- Design: allows to check the effects of the application specific traffic on the communication network. For example, the traffic generated by ProfiDrive telegrams.
- Pre-commissioning: planning and test the entire network in different scenarios, with the DT, allows to obtain an estimate of the performance and highlights possible problems.
- Diagnosis and Maintenance: analysing the traffic on the real network, with respect to that of the DT, definitely facilitates the process of fault detection.

In the proposed implementation, we choose to exploit (and to extend) the OMNeT++ INET framework in order to take advantage of all the already available layers, protocols and features. The main objective is to create an OMNeT++ module that simulates both the Profinet Controller and Device and that implements the cyclic message exchange. In this regard, it has been implemented a module for the network layer which performs the encapsulation and extraction of a Profinet PDU within/from an Ethernet frame. Moreover, it has been developed a basic scheduler able to emulate the real one. This modular structure reveals particularly advantageous for the future steps towards the full implementation of the digital twin, since the electrical drive models will be easily integrated with the communication ones.

The remainder of this paper is organized as follows. Section II provides a brief review of the Profinet IO RT standard and discusses the internal hardware architecture of a Profinet Device. Section III presents some aspects of the simulator development process. Section IV evaluates the simulation model comparing the results it provides with some measurements on a real test bench. Section V concludes the paper.

## II. Profinet IO RT Overview

Profinet IO is defined by [6] and [7] as part of the IEC 61158 International Standard. The application area of Profinet is very wide, with diverse and heterogeneous performance requirements. For this reason, the protocol specifications define four different classes of service namely RT Class UDP, and RT Class 1 to RT Class 3. These classes differ in real-time capabilities as well as in the availability of the clock synchronization. RT Class 1, on which we will focus on this paper, is usually adopted in factory automation applications where, typically, cycle times are in the order of some ms, with very limited jitter [8]. This class operates using the IEEE 802.1Q standard, which defines eight frame priorities (from 0 to 7, where a lower value indicates low priority traffic). Typically in a Profinet network RT Class 1 frames are tagged with priority 6. To correctly handle VLAN tag priorities, Profinet uses the so-called managed switches. Such devices are able to manage the packets according to their priorities: frames with higher priority take precedence during the queuing phase in the switches. Thus, the use of VLAN-based priority ensures that real-time packets are served first, with respect to standard TCP packets.

### A. Communication cycle

In Profinet, cyclic data, such as process values, are transmitted with a fixed period, referred to as send clock time. Such a value is device–specific and selected during the configuration of the network by the user. It is defined as an integral multiple of the basic time unit of 31.25 µs. Typically, the send clock time is in the range 31.25 µs – 4 ms, expressed as

$$Send\ clock\ time = send\ clock\ factor \cdot 31.25\mu s$$

where the send clock factor is specified by the standard in the range 1 – 128.

To guarantee real time and at the same time support for generic TCP-type communications, a TDMA-type access mode is adopted: the send clock time is divided into several intervals, each containing different types of traffic.

### B. Profinet RT Frame

A Profinet frame is encapsulated in an Ethernet II frame as represented in Fig. 1. It is composed by the following fields:

- The *VLAN TAG* has a total length of 2 byte and defines the priority, the Canonical Format Identifier and the VLAN ID. For Profinet in a standard Ethernet network it is 0xC000.
- The *Type* identifies the network protocol following in the data field. For Profinet it has the value 0x8892.
- *Frame ID* indicates the frame type.
- *Data* contains the actual process data.
- *Cycle counter* is incremented by one unit every 31.25 µs
- *Data Status* indicates whether the transmitted data is valid
- *Transfer Status* indicates whether the transmission was successful.

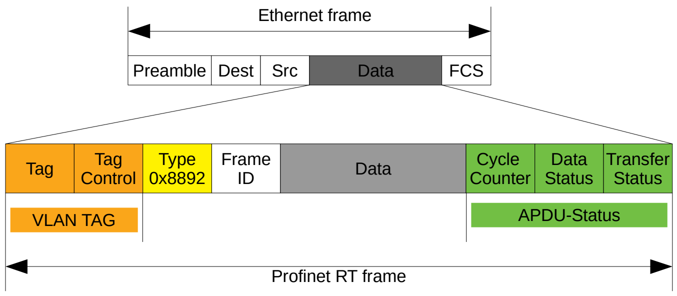


Fig. 1. Structure of Profinet RT frame

## III. Implementation of the simulation model

OMNeT++ is an extensible, modular, component-based C++ discrete event systems simulator. It provides several extended frameworks, such as INET, that contains models of widespread wired and wireless networking protocols, including UDP, TCP, LLDP, Ethernet, DCP and many others. This is a particularly interesting feature since Profinet extensively uses some of these protocols. Another interesting feature is the possibility of natively recording network traffic in *pcap* format, allowing to analyze and compare acquisitions with general purpose network analyzers tools such as Wireshark.

### A. Device model

A Profinet IO device used in networked electrical drive systems is equipped with a built–in switch, so that it can be connected in daisy chain configurations, which are very popular in such application contexts. From this point of view, a Profinet IO device can be seen as a compound module composed of an addressable device with multiple ports and switching capabilities [9]. In other words, it can be modeled as a generic host connected to a 2 ports managed switch as shown in Fig. 2.

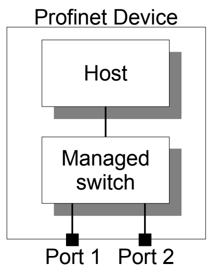


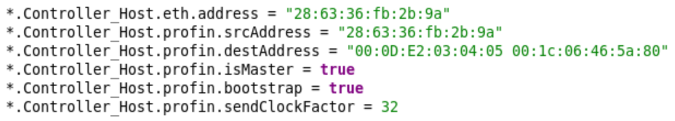
Fig. 2. Model of a Profinet Device

Notably, this model can be used also for different Profinet nodes, such as IO Controllers, simply using only one of the switch ports.

Since in real devices, typically, the internal connection between the host and the switch is handled by an ASIC (Application Specific Integrated Circuit) and the inter-port delay in a Profinet device is about 16 ns (see for example,

[10]), the connection between the host and the switch has been modelled as a high bandwidth low latency connection.

The Host has been implemented to simulate both Profinet IO Controller and IO Device. The user can characterize a node either as IO Controller or IO Device simply modifying some parameters in a configuration file (*omnetpp.ini*). To parameterize an IO Device, it is sufficient to set its own MAC address. On the other hand, as can be seen in Fig. 3, an IO Controller takes as parameters the list of the MAC addresses of the IO Devices, a flag to indicate that the node is an IO Controller, a bootstrap flag and, finally, the send clock factor. These last tree parameters are necessary to enable scheduling and initialization capabilities of the Profinet IO Controller. Indeed, the *isMaster* flag enables the scheduler which calculates the sending interval by using the *sendClockFactor* parameter.



```
*.Controller_Host.eth.address = "28:63:36:fb:2b:9a"
*.Controller_Host.profin.srcAddress = "28:63:36:fb:2b:9a"
*.Controller_Host.profin.destAddress = "00:0D:E2:03:04:05 00:1c:06:46:5a:80"
*.Controller_Host.profin.isMaster = true
*.Controller_Host.profin.bootstrap = true
*.Controller_Host.profin.sendClockFactor = 32
```

Fig. 3. Configuration of the Controller in OMNeT++

The Profinet Host described in Fig. 2 has been built by extending the OMNeT++ module *EtherHost*, as shown in Fig. 4.

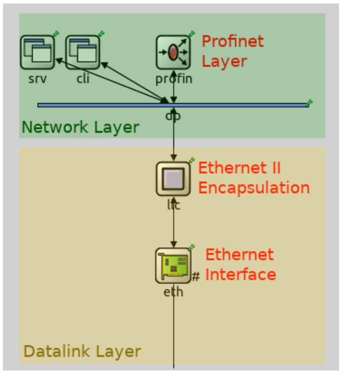
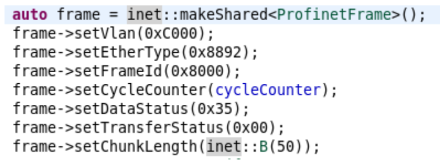


Fig. 4. Layers structure of the Controller/Device in OMNeT++

As can be seen, the EtherHost module already implements the whole Data–Link layer protocol, while the RT class 1 Profinet module has been formally implemented at the Network layer. This module, in particular, implements two essential features of the protocol, namely, the packet encapsulation/extraction and the generation of the basic time unit of 31.25 µs, which is used by both the scheduler and cycle counter. The encapsulation/extraction has been developed according to the packet structure in Fig. 1; its implementation in OMNeT++ is shown in Fig. 5.

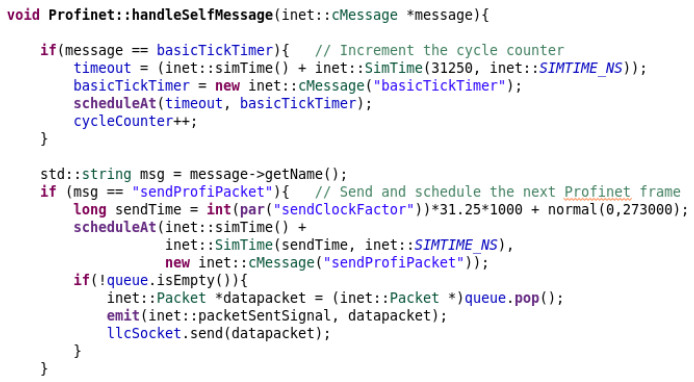


```
auto frame = inet::makeShared<ProfinetFrame>();
frame->setVlan(0xC000);
frame->setEtherType(0x8892);
frame->setFrameId(0x8000);
frame->setCycleCounter(cycleCounter);
frame->setDataStatus(0x35);
frame->setTransferStatus(0x00);
frame->setChunkLength(inet::B(50));
```

Fig. 5. Profinet frame dissection in OMNeT++

Such a feature allows to manipulate individual fields at runtime by means of class methods. As an example, it allows to extract the process data to be passed to the application layer.

Since OMNeT++ is an event based simulator, the most simple and effective way to implement the generation of the basic time unit is by using self messages. As can be seen in Fig. 6, a self message is scheduled every 31.25 µs and the cycle counter is incremented.



```
void Profinet::handleSelfMessage(inet::cMessage *message){

    if(message == basicTickTimer){   // Increment the cycle counter
        timeout = (inet::simTime() + inet::SimTime(31250, inet::SIMTIME_NS));
        basicTickTimer = new inet::cMessage("basicTickTimer");
        scheduleAt(timeout, basicTickTimer);
        cycleCounter++;
    }

    std::string msg = message->getName();
    if (msg == "sendProfiPacket"){   // Send and schedule the next Profinet frame
        long sendTime = int(par("sendClockFactor"))*31.25*1000 + normal(0,273000);
        scheduleAt(inet::simTime() +
                   inet::SimTime(sendTime, inet::SIMTIME_NS),
                   new inet::cMessage("sendProfiPacket"));
        if(!queue.isEmpty()){
            inet::Packet *datapacket = (inet::Packet *)queue.pop();
            emit(inet::packetSentSignal, datapacket);
            llcSocket.send(datapacket);
        }
    }
```

Fig. 6. Implementation of the scheduler and cycle counter

With the same approach, the simulator schedules and sends the cyclic process data frames. The possible presence of jitter, in this case, has been simulated by adding a random normal term to the schedule time.

## IV. Evaluation of the simulation model

The actual validation of a simulation model is a task that needs particular care and requires the execution of several tests. In this Section we reports the outcomes of some experimental sessions we carried out towards the final validation of the proposed simulator. We refer to the prototype network described in Fig. 7 we implemented. This network comprises a PLC as Profinet IO Controller (Siemens Simatic 1500 with CPU 1511T–1PN) and two drives, configured as IO Devices, namely a Siemens Sinamics V90 servo drive and a Brushless motor with integrated servo drive (IBD60-PNT, produced by an Italian firm [11]). The network also includes a ProfiTAP device used for traffic acquisition. All components are connected via a daisy–chain with full-duplex 100 Mbps Ethernet. The cable length between each node is about 3 m. The send clock factor has been set to 32, corresponding to a period of the IO Controller of 1000 µs. The same network was implemented with OMNeT++.

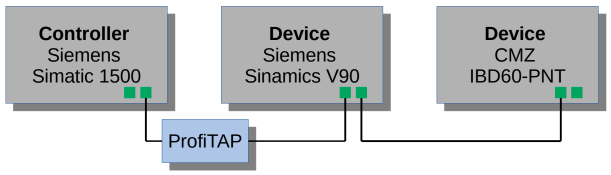
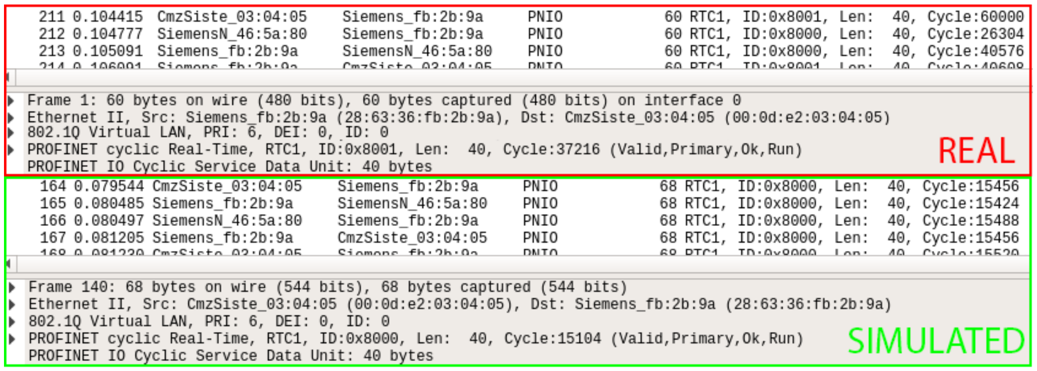


Fig. 7. Network topology

A first outcome is shown in Fig. 8, which reports the Wireshark acquisition plots for both the real and simulated networks. As can be seen, from a qualitative point of view the two acquisitions are practically indistinguishable. Notably, Wireshark revealed able to interpret the frame transmitted on the simulator, meaning that the fields are adequately encapsulated and carry correct data. Analyzing both timestamp and cycle counter, it can be seen that the IO Controller sends the



```
 211 0.104415  CmzSiste_03:04:05    Siemens_fb:2b:9a     PNIO            60 RTC1, ID:0x8001, Len:  40, Cycle:60000
 212 0.104777  SiemensN_46:5a:80    Siemens_fb:2b:9a     PNIO            60 RTC1, ID:0x8000, Len:  40, Cycle:26304
 213 0.105091  Siemens_fb:2b:9a     SiemensN_46:5a:80    PNIO            60 RTC1, ID:0x8000, Len:  40, Cycle:40576

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▶ Ethernet II, Src: Siemens_fb:2b:9a (28:63:36:fb:2b:9a), Dst: CmzSiste_03:04:05 (00:0d:e2:03:04:05)
▶ 802.1Q Virtual LAN, PRI: 6, DEI: 0, ID: 0
▶ PROFINET cyclic Real-Time, RTC1, ID:0x8001, Len:  40, Cycle:37216 (Valid,Primary,Ok,Run)
  PROFINET IO Cyclic Service Data Unit: 40 bytes
```

```
 164 0.079544 CmzSiste_03:04:05    Siemens_fb:2b:9a     PNIO            68 RTC1, ID:0x8000, Len:  40, Cycle:15456
 165 0.080485 Siemens_fb:2b:9a     SiemensN_46:5a:80    PNIO            68 RTC1, ID:0x8000, Len:  40, Cycle:15424
 166 0.080497 SiemensN_46:5a:80    Siemens_fb:2b:9a     PNIO            68 RTC1, ID:0x8000, Len:  40, Cycle:15488
 167 0.081205 Siemens_fb:2b:9a     CmzSiste_03:04:05    PNIO            68 RTC1, ID:0x8000, Len:  40, Cycle:15456

▶ Frame 140: 68 bytes on wire (544 bits), 68 bytes captured (544 bits)
▶ Ethernet II, Src: CmzSiste_03:04:05 (00:0d:e2:03:04:05), Dst: Siemens_fb:2b:9a (28:63:36:fb:2b:9a)
▶ 802.1Q Virtual LAN, PRI: 6, DEI: 0, ID: 0
▶ PROFINET cyclic Real-Time, RTC1, ID:0x8000, Len:  40, Cycle:15104 (Valid,Primary,Ok,Run)
  PROFINET IO Cyclic Service Data Unit: 40 bytes
```

Fig. 8. Comparison of real and simulated network in Wireshark

process data every 1000 µs i.e. every 32 basic clock cycle. This is also confirmed by the statistics reported in Table I, which shows similar results for the sending interval on both real and simulated networks.

TABLE I
STATISTICS OF THE CONTROLLER SENDING INTERVAL

| | Mean [µs] | Jitter [µs] |
|---|---|---|
| Real | 1003 | 304 |
| Simulated | 1001 | 273 |

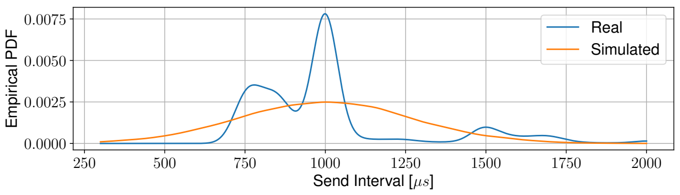


Fig. 9. EPDF comparison of the send interval in the real and simulated network

Finally, the probability distribution of the sending interval is reported in Fig. 9 for both the real and the simulated cases. As can be seen, in the real case, the jitter has a bi–modal behavior not reflected by the simulated one (which was assumed to be Gaussian). Consequently, in order to better tune the simulator, the type of jitter in the real network needs to be further investigated.

## V. CONCLUSION AND FUTURE WORK

In this paper we addressed the digital twin of networked electrical drive systems. Particularly, we focused on the simulation of Profinet IO RT Class 1 networks with OMNeT++, which represents the first step towards the implementation of the digital twin. To this purpose, we developed a model for Profinet IO Controller and Device, exploiting the INET library, which allows to adopt several available protocols used by Profinet. This choice ensured a high degree of modularity, so that the the final completion of the simulator and, more importantly, of the digital twin will be facilitated.

Future activities are envisaged in different directions. First, the simulator needs to be completed. Indeed, there is the need to implement further modules such as that which handles the acyclic traffic. Moreover, the handling of frame priorities has to be improved, possibly using the CORE4INET framework made available by OMNeT++. Secondly, as we have shown, the model adopted for the jitter on the sending period does not reflect adequately the real case behavior. This aspect suggests further investigations in order to better tune the simulator. More in general, extensive session tests on both real networks and their simulated counterparts are necessary to come to the effective validation of the simulator. Finally, adequate models for the electrical drives need to be developed. This activity requires to deeply analyze the adopted components, in order to infer their behaviors and, particularly the latency they introduce in the whole system [12].

## REFERENCES

[1] S. Vitturi, C. Zunino, and T. Sauter, "Industrial Communication Systems and Their Future Challenges: Next-Generation Ethernet, IIoT, and 5G," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 944–961, Jun. 2019.

[2] M. Schluse, M. Priggemeyer, L. Atorf, and J. Rossmann, "Experimentable Digital Twins—Streamlining Simulation-Based Systems Engineering for Industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1722–1731, Apr. 2018.

[3] S. Vitturi, L. Peretti, L. Seno, M. Zigliotto, and C. Zunino, "Real-time Ethernet networks for motion control," *Computer Standards & Interfaces*, vol. 33, no. 5, pp. 465–476, 2011.

[4] H. Kleines, S. Detert, M. Drochner, and F. Suxdorf, "Performance Aspects of PROFINET IO," *IEEE Transactions on Nuclear Science*, vol. 55, no. 1, pp. 290–294, 2008.

[5] Z. Hanzálek, P. Burget, and P. Šůcha, "Profinet IO IRT Message Scheduling With Temporal Constraints," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 369–380, Aug. 2010.

[6] International Electrotechnical Commission, *IEC 61158: Industrial Communication Networks – Fieldbus Specifications. Part 5-10: Application Layer Service Definition - Type 10 Elements*, 2019.

[7] International Electrotechnical Commission , *IEC 61158: Industrial Communication Networks – Fieldbus Specifications. Part 6-10: Application Layer Protocol Specifications - Type 10 Elements*, 2019.

[8] G. Prytz, "A performance analysis of EtherCAT and PROFINET IRT," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, Sep. 2008, pp. 408–415.

[9] M. May, D. Ganz, and H. Dermot Doran, "HSR and PROFINET IRT bandwidth management in generic embedded systems," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*. Krakow, Poland: IEEE, Sep. 2012, pp. 1–4.

[10] Siemens. (2010) ERTEC 200 – Enhanced Real–Time Ethernet Controller. [Online]. Available: https://cache.industry.siemens.com/dl/files/782/23234782/att_98701/v1/ERTEC200_Manual_V112.pdf

[11] "CMZ Sistemi Elettronici s.r.l." https://www.cmz.it/, via dell'Artigianato, 21 – Vascon di Carbonera (TV), Italy.

[12] L. Seno, F. Tramarin, and S. Vitturi, "Performance of Industrial Communication Systems: Real Application Contexts," *IEEE Industrial Electronics Magazine*, vol. 6, no. 2, pp. 27–37, 2012.