BE in CPPS

Innovation Action Project

HORIZON 2020 - EU.2.1.5. - Ref. 680633

D2.8 – Smart Systems Platform Federation

Lead Author: TTTech

With contributions from: ITI, NISSA, FORTISS

Reviewer: CEA, HOLONIX

Deliverable characteristics

Deliverable nature	R
Dissemination level	PU
Contractual delivery date	31 October 2017
Actual delivery date	14 November 2017
Version	1.0
Keywords	Smart Systems Platform, Field Level, Embedded Systems, Real-time Communication, Wireless communication

Version history

Nr.	Date	Notes and comments
0.1	03/10/2017	First ToC, initial Draft
0.2	09/10/2017	Description of Smart Systems Platform
0.3	11/10/2017	Description of WSN4CPPS (ITI) and OPC UA over TSN (TTT)
0.4	17/10/2017	Description of Embedded CEP (NISSA)
0.5	18/10/2017	Description of TSN-enabled 4diac RTE (Fortiss)
0.6	2/11/2017	Final version ready for Review
1.0	14/11/2017	Final version ready for quality check

Deliverable Peer review summary

Id.	Addressed	Comments
CEA	X	General comments, overall very positive and no comments to the content

HLX	X	Small writing errors corrected, content wise small comments (regarding open source and proprietary components) addressed

List of Figures

List of Tables

None found

Table of contents

Executive summary

The updated BEinCPPS Reference Architecture, introduced in Deliverable D2.2 (which is an updated version from the one in Deliverable D2.1), is similarly as its previous version divided into three different levels, the Cloud Level, the Factory Level and the Field Level. Each level focuses on different aspects of Cyber-Physical Production Systems (CPPS). This document reports on the individual foreground component on the Field Level in the Reference Architecture. Embedded systems (field computation, logic) and real-time networks (field communication, wired and wireless) are the two main topics covered on this level.

Most of the components identified as foreground on the field level are loosely coupled and can be applied as stand-alone parts by the end users. An exception here is the integration of TSN-enabled 4diac RTE, which is a combination of two components on the field level.

As mentioned before, the focus of this deliverable, in comparison to the previous deliverable D2.7, is on the foreground components. They have been identified already in the previous document, but were then under development. The components described here are available, either as open source or as proprietary components (the ones coming from industrial companies).

The components introduced in this document are the foreground components that are developed inside WP2 of the BEinCPPS project. There are four components developed, namely Embedded CEP for CPPS, TSN-enabled 4DIAC, wireless communication for CPPS and Time-Sensitive Networking (TSN) for CPPS (in combination with Real-time OPC UA).

The different components will be integrated in the final demonstrators, either at the BEinCPPS champions or at some of the winners of the open call. These demonstrators will be described in more detail in their respective deliverable according to the corresponding work packages.

Finally, all the above introduced foreground components are described in more detail in their respective factsheets. The sheets are attached to this deliverable and a reference to them can be found in the according Annex.

1 Introduction

1.1 Introduction

The Smart Systems Platform has been introduced in previous documents as the lowest level, the field level, in the overall BEinCPPS Reference Architecture. This level describes the components that are (further) developed within the BEinCPPS project, that directly interact with the already present components on the field level (e.g. controllers, robots, machines, conveyor belts etc.).

Within the BEinCPPS project, multiple components have been identified and developed by the Smart Systems Platform partners. The components described in this document target the areas of real-time networks (field communication, wired as well as wireless) and embedded control (field computation). The components are all loosely coupled and can be applied as individual, stand-alone solutions, but some of them are targeted to be used as a combination (e.g. TSN-enabled 4DIAC RTE, which combines two technologies).

Additionally, the described components are separated into open source and proprietary solutions. The factsheets attached to this document as annexes, will describe clearly how the different components can be accessed and applied in the manufacturing settings.

1.2 Scope of the deliverable

Deliverable D2.8 updates and finalizes the work introduced in deliverable D2.7 [2] with respect to the Smart Systems federation. Whereas deliverable D2.7 also focused on components that were background components this deliverable will not repeat the information, but put the main emphasis on the foreground information developed since deliverable D2.7.

The document starts in section 2 with a short introduction of the Smart Systems Platform and the updated BEinCPPS reference architecture (which is described in more detail in deliverable D2.2 – BEinCPPS Architecture and Business Processes). Additionally, the Smart Systems Platform is again positioned within the updated architecture together with the individual components.

Section 3 summarizes the final foreground components that are included in the Smart Systems Platform in more detail. In comparison with the previous deliverable (D2.7), here only the focus will be on the foreground components, thus showing the results of the developments of the BEinCPPS project. In this section, the foreground components will be shortly described on a high level and the more technical details (like e.g. functionalities, installation, tutorials, etc.) will be presented in the factsheets that are attached to this deliverable.

Finally, section 4 concludes the deliverable.

1.3 Contributions to other WPs

The content of this deliverable relates to all the different technical parts in the other work packages where the Smart Systems Platform technologies are being deployed (as seen in Figure 1). There is a close cooperation with the work in Work Package 3, where there is a description of how the different technologies can be applied. Work Packages from 4 to 8 involve the champions and their demonstrators. Within these demonstrators, the different technologies from the Smart Systems Platform will be integrated. For a detailed description of how the technologies are being applied in these demonstrators, we refer to the respective deliverables of the individual Work Packages.

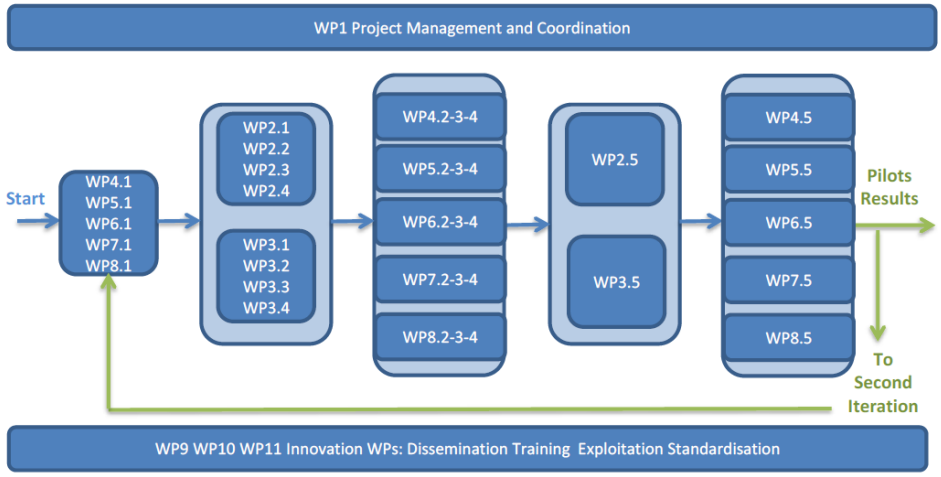


Figure 1 Project PERT

1.4 Contributions to other deliverables

Description of Actions for the BEinCPPS project, providing the basis for the entire project and this deliverable content:

- D2.2: BEinCPPS Architecture and Business Processes, Identification and update of the Smart System Platform
- D2.7: First version of the Smart Systems Platform, which forms the basis for the final version of Smart Systems Platform, being described in this deliverable.

2 Architecture of the Smart Systems Platform Federation

2.1 Introduction and Positioning in the updated BEinCPPS Architecture

As described in the BEinCPPS deliverable D2.2 [1], an updated version of the BEinCPPS architecture has been made. As can be seen in Figure 2, the architecture has been simplified and split up in three different layers, representing the field, factory and cloud level. The Smart Systems Platform Federation remains at the Field Level (or also still mentioned as the shopfloor level), containing both hardware and software assets (as can be seen in Figure 3). The components of the Smart Systems platform mainly cover (wired and wireless) deterministic communication and (embedded) control or logic.

The Figure 2 till Figure 5 show the different perspectives of the BEinCPPS reference architecture and from these perspectives it becomes clear what the components of the Smart Systems Platform will focus on. The final perspective, the Implementation one, shows the different assets that will be available on the field level and identifies which ones are back- and foreground.

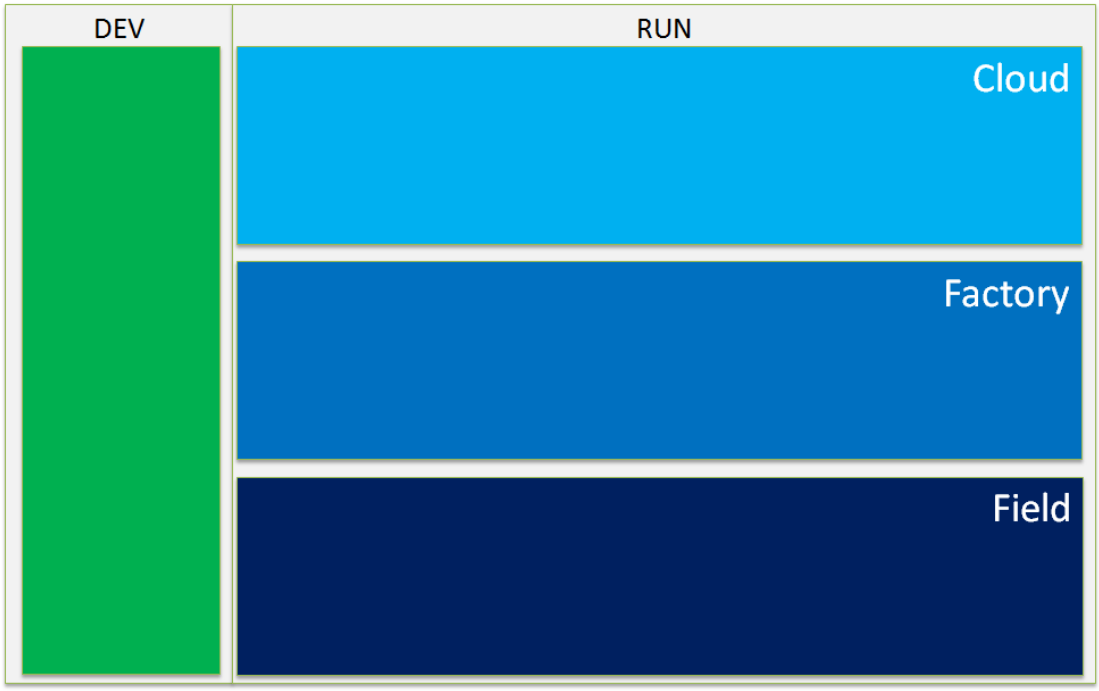


Figure 2: BEinCPPS Architecture – Structural

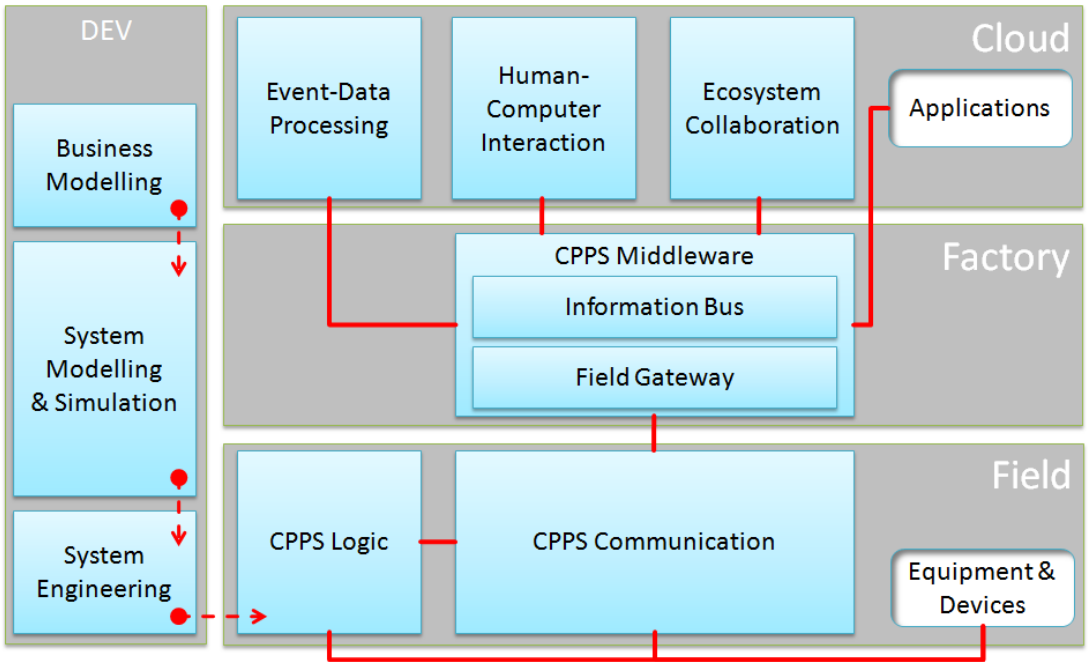


Figure 3: BEinCPPS Archtiecture – Functional Perspective

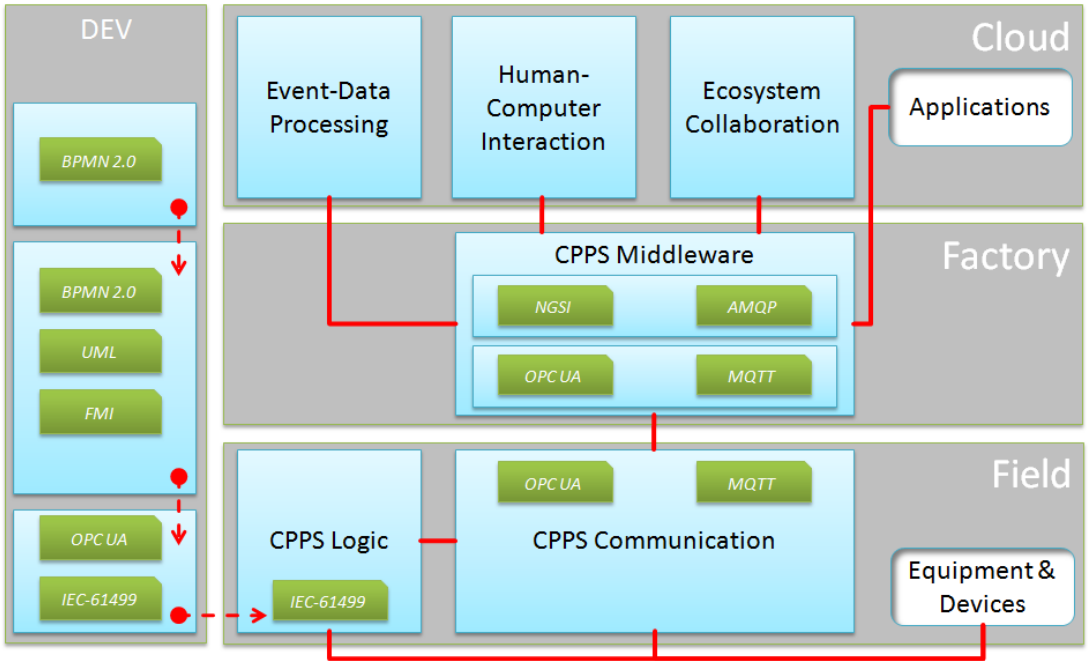


Figure 4: BEinCPPS Architecture – Technical Perspective

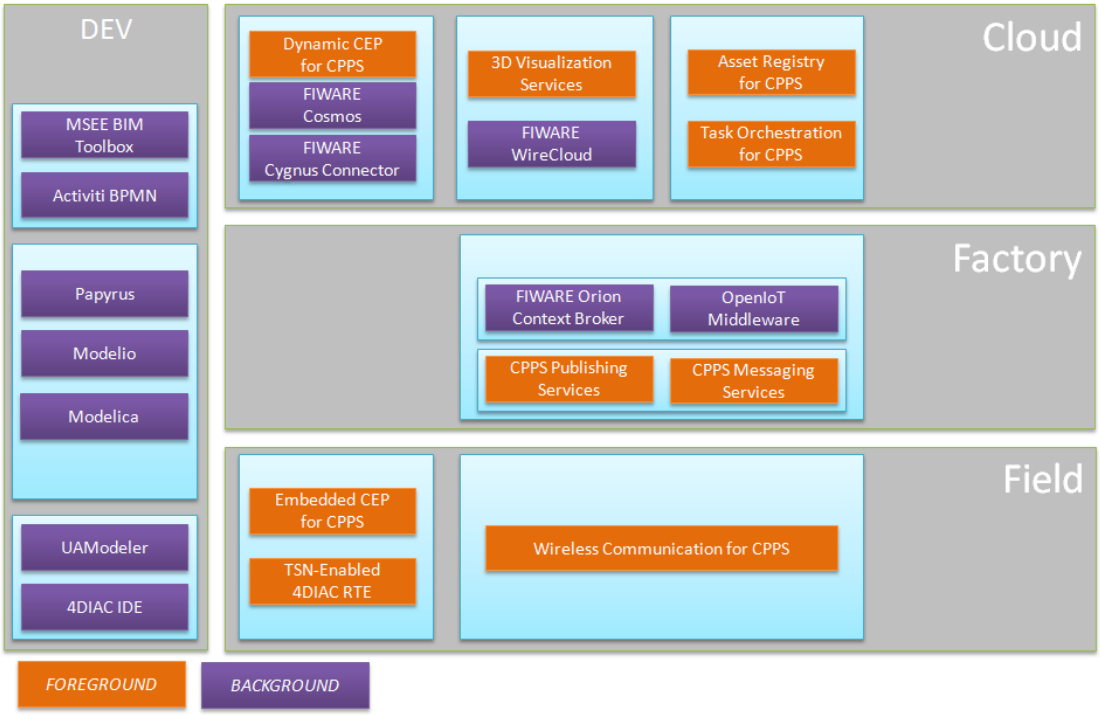


Figure 5: BEinCPPS Architecture – Implementation Perspective

2.2 Updated Smart Systems Federation Architecture

The main responsibility of the Smart System Federation Architecture is enabling the CPS-ization of the factory (as described in Deliverable D2.2), which is identified as the transformation of the field level towards more intelligent, active and connected devices.

The final version of the Smart Systems Federation Architecture is closely related to the version introduced in the previous version of this deliverable.

The Smart Systems federation is still located on the lowest level of the BEinCPPS architecture, namely on the Field Level. The scope of this level is the shopfloor and its physical processes.

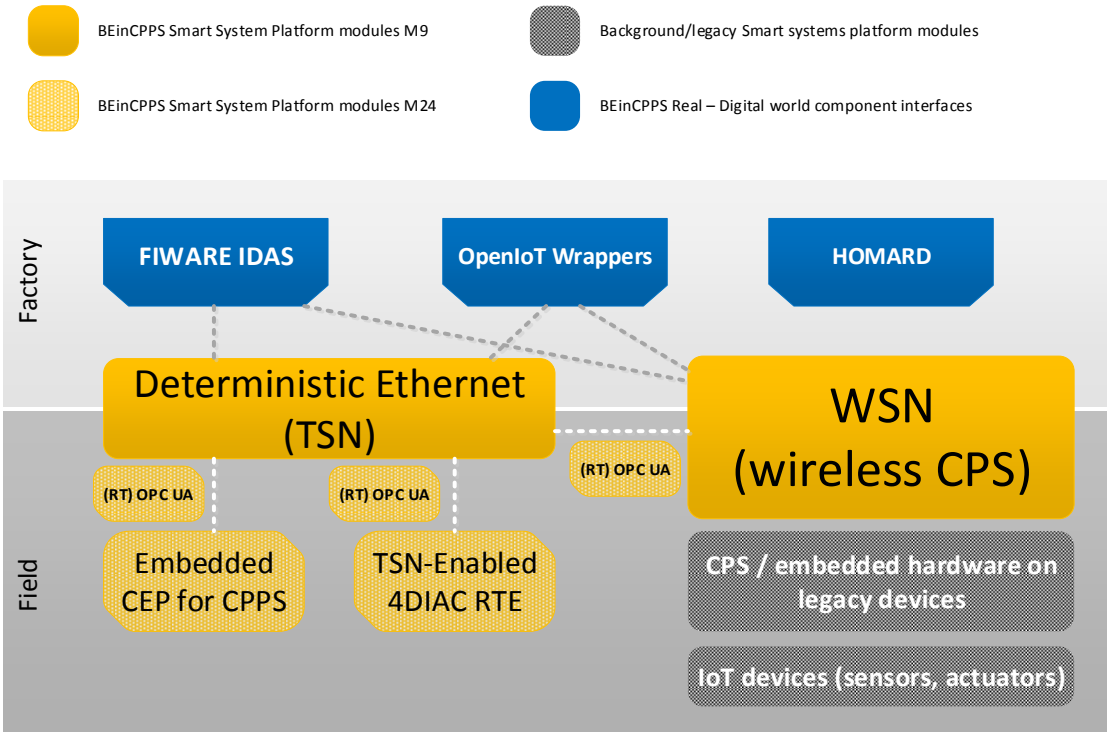


Figure 6: Updated Smart Systems Federation Architecture

The aim of the Smart Systems architecture was to provide loosely coupled components, with no dependencies on each other and that can be applied individually, based on the requirements from the end users (see Figure 6). Some minor modifications with respect to the version in deliverable D2.7 have been made. This goal has to a certain extent been reached, e.g. the TSN-enabled 4DIAC RTE is dependent on the Deterministic Ethernet component.

The components that are in the final version of the smart systems federation architecture are described in the following section.

3 Components

The implementation perspective of the BEinCPPS reference architecture introduced the relevant foreground components that have been developed within the BEinCPPS project for the Smart Systems federation. The following components are identified as foreground in the Field Level:

- Embedded CEP for CPPS (developed by partner NISSA)
- TSN-Enabled 4DIAC RTE (developed by partner fortiss in cooperation with TTT)
- Wireless Communication for CPPS (developed by partner ITI)

These components will be made available as open source solution (latest after the end of the project). Additionally, one value added service have been developed as foreground components within the project, which is however a proprietary solution. This is the following:

- Time-Sensitive Network for CPPS (combined with real-time OPC-UA, developed by partner TTT)

In the following sections the individual assets are shortly introduced and the changes with respect to deliverable D2.7 are highlighted. A more detailed description of the different assets will be given in the fact sheets that are attached to this deliverable.

3.1 Foreground Components

This section describes the foreground components developed in the duration of the BEinCPPS project. The foreground components are distributed in open source and proprietary components or also called Value Added Services (VAS).

3.1.1 Open Source Components

Embedded CEP for CPPS

Embedded CEP for CPPS is the framework for deploying and running CEP application in the resource-constrained devices (like Raspberry Pi).

It is a very thin application that runs on an embedded system which has installed an operating system with a java virtual machine. The system should have a network connection for deploying applications and sending alarms.

Several functions are supported:

- They can aggregate different types of sensor data and normalize them into standard IP traffic that is well-understood by IT.
- An intelligent edge gateway has the processing capacity to perform additional analytics in real or near-real time to make data-driven decisions as close to the data generation as possible.

- Performing analytics on the gateways helps reducing network bandwidth cost because only meaningful information needs to be sent to the next tier, whether it is another gateway, the datacenter, or cloud.
- Moreover, Embedded CEP can communicate with the analytics modules placed on the server side and bring the global context close to the edge, by opening new possibilities for dynamic monitoring/processing on the edge.

There are there main value propositions:

- Enabling creation of the real-time situational awareness on the edge.
- Increasing reactivity close to the shop-floor (e.g. reacting on anomalies).
- Supporting decision making on the edge (even in the case that connectivity is weak).

Unique Selling Points are:

- Enabling very complex edge processing
- Supporting dynamicity on the edge
- Easy deployment of new patterns (situation of interest).

Detailed information can be found in the factsheet #1.

TSN-Enabled 4DIAC RTE

Eclipse 4diac™ (eclipse.org/4diac) is an open source project developed at fortiss. It is composed of the 4diac™ integrated development environment (4diac-IDE) and the 4diac runtime environment (4diac-RTE, forte). forte supports many different industrial communication protocols (like e.g. OPC UA, openPOWERLINK, MQTT, …), which are implemented by different communication layers. A new Time-Sensitive Networking (TSN) enabled communication layer for forte has been implemented with the BEinCPPS project that supports real-time and deterministic traffic via standard Ethernet. TSN is currently standardized by the Time-Sensitive Networking Task Group which is part of the IEEE 802.1 Working Group.

The advantages of TSN are:

- **Convergent network**: real-time critical and uncritical traffic are transmitted on a single network interface, hence better bandwidth utilisation and reduction of cabling effort.
- **Interoperability**: standard Ethernet removes the need for dedicated fieldbuses
- **Determinism and real-time:** TSN guarantees that packets are transmitted correctly and fulfil real-time requirements.

The TSN communication layer implements publish-subscribe connections via Service Interface Function Blocks (SIFB) which are enhanced by a VLAN ID and the message priority, represented in the ID data input parameters of the publish FB. These SIFB can now send prioritized UDP multicast packets.

Extended information can be seen in the corresponding factsheet #2.

Wireless Communication for CPPS

Wireless sensors networks (WSN) offer great potential for the future, since they allow a large number of sensor points to be covered on a surface at a low cost and without the use of wiring. However, this technology still has certain drawbacks that are delaying its market penetration: the heterogeneity of existing communications protocols, and the complexity of its deployment and maintenance. In order to provide a solution to the second problem, ITI has adapted and developed a set of protocols and tools that facilitate the deployment of wireless sensors and simplify their management by users. On the one hand, the system uses dynamic communication protocols that adapt to the conditions of the environment, and adapt to node dropping, concealment and interference, allowing an unattended operation. On the other hand, each of the nodes offers a simple user interface to assist in the deployment phase, which calculates the communication quality of the node at the current point. This algorithm takes into account all the jumps that node needs to reach the network gateway, and considers aspects such as packet loss and signal strength. The user does not need to be aware of the technology he uses below, aspects of topology or communications, or whether the Gateway is at one hop or more, all this is transparent to him. The technology developed also includes the procedures to be followed for the deployment phase, and various additional technologies such as dynamic routing based on the link quality information, automatic discovery of Gateway and simplified interface for connection of sensors.

The design of an efficient media access control (MAC) protocol is also of vital importance for WSN, especially in the for its adoption in industrial use cases. The MAC layer is responsible for channel access, scheduling, buffer management, and error control policies. In a WSN deployed in an industrial environment, it is necessary to include a MAC protocol that provides reliability, energy efficiency, low bandwidth delay and utilization as top priorities. The solution provided by the developed WSN4CPPS nodes feature state of the art mechanisms with bi-dimensional time/frequency scheduling and autonomous dynamic scheduling algorithms. Extended information can be seen in the corresponding factsheet #3.

Also, the WSN4CPPS Gateway enables the interaction of the wireless nodes with the wired network and other devices, features application protocols adopted as standards in Industrie 4.0 as OPC UA, or the also extended MQTT, enabling data flow from field devices toward the BEinCPPS Information Bus or other platforms (see Figure 7).

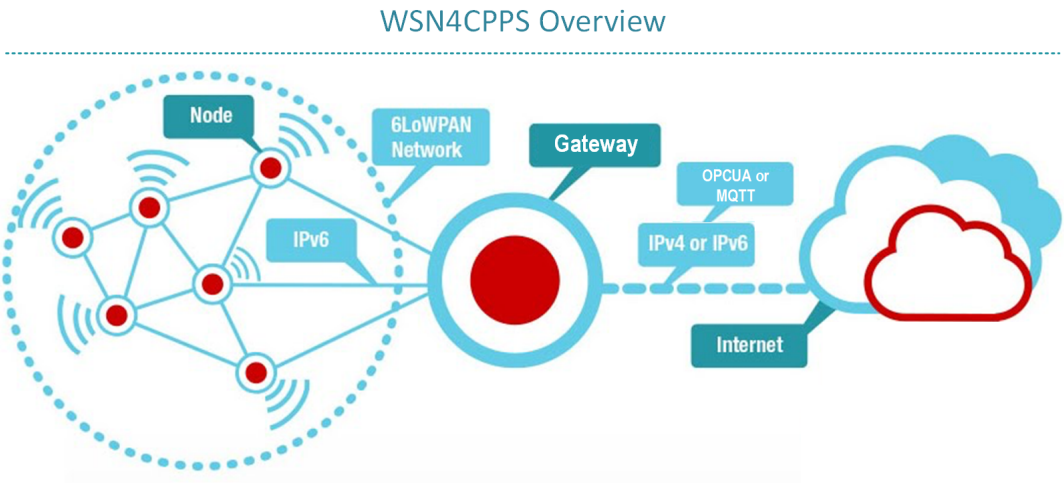


Figure 7: WSN4CPPS Overview

3.1.2 Proprietary Components/Value Added Systems

OPC UA over TSN for CPPS

OPC Unified Architecture (OPA UC) is a machine to machine communication protocol for industrial automation developed by the OPC Foundation. It provides a framework that defines rules how to transform data into information and how to manage that information in a distributed environment where data needs to be exchanged between e.g. machines (Machine-to-Machine, M2M) to from machines to the Cloud.

However, until now OPC UA used to rely on a Client/Server model to perform communication between different machines, which had its limitations with respect to complex processes with real-time requirements. The new model is based on a Publish/Subscribe approach, where a device (the publisher) provides standard format data to a set of other devices (the subscribers). The publisher can provide this information at set time intervals so subscribers know when to expect new machine data.

The core technology of TTT is IEEE Time-Sensitive Networking (TSN), which provides an open, standard method for communicating data in real-time over Ethernet by extending existing Quality of Service mechanisms in Ethernet with additional capabilities for latency control, determinism, and high availability. TSN achieves deterministic real-time communication over Ethernet by using global time and a schedule which is created for message paths across multiple network components. By defining queues which transmit their messages based on a time schedule, TSN ensures a bounded maximum latency for scheduled traffic through switched networks.

TTT focused its work in the BEinCPPS on the integration of OPC UA Pub/Sub into IEEE TSN. OPC UA Pub/Sub is not enough to provide real-time communication

with OPC UA. The integration with TSN enables the change from the classical automation pyramid (Industry 3.0), where the different layers in the factory are not able to directly communicate with each other. The introduction of Industry 4.0 and the Industrial Internet of Things (IIoT), where everything is connected to each other requires a new machine to machine communication approach, including also higher level (ERP) to lower level (field devices) communication. TSN, in combination with OPC UA, enables real-time communication between the different layers, thereby opening for companies the possibility to integrate Industry 4.0 ready real-time communication concepts in their manufacturing lines.

A more detailed information about the OPC UA Pub/Sub, TSN and OPC UA over TSN can be found in factsheet #4.

4 Conclusion

This deliverable describes the updated Smart Systems Platform federation developed inside the BEinCPPS project. This deliverable is a follow-up from the deliverable D2.7 and uses the updated BEinCPPS reference architecture (described in deliverable D2.2) as a basis for the new Smart Systems component. The components identified in this deliverable are integrated in the various BEinCPPS champions and the external partners that joined the consortium. The components described within the document are the following:

- Embedded CEP for CPPS
- TSN-enabled 4diac RTE
- Wireless communication for CPPS
- OPC UA over TSN for CPPS

In comparison to the previous deliverable, this document only focuses on the foreground components developed inside BEinCPPS. The different foreground components are separated in open source and proprietary components and are elaborated in more detail in the attached factsheets (appended to this document in the Annexes).

5 Bibliography

[01] BEinCPPS Deliverable D2.2 – BEinCPPS Architecture & Business Processes, Available online, http://www.beincpps.eu/

[02] BEinCPPS Deliverable D2.7 – Ssystems Platform Federation

6 Appendix – Factsheet 1: Embedded CEP

Innovation Action Project

HORIZON 2020 - EU.2.1.5. - Ref. 680633

D2.8 - Smart Systems Platforms Federation

Factsheet #1

Relevant for µCEP Component

Lead Author: Nenad Stojanovic [Nissatech]

With contributions from: -

1. Short Description of the Component

The task of Embedded CEP is to pre-process events in close proximity to their source, avoiding network latency and thus enabling a first level of true real-time control. Additionally, this approach protects the privacy of data since the manufacturing data will be processed locally

Embedded CEP will enable complex processing on the edge on the network, opening the possibilities for realizing Edge Computing infrastructure.

It represents a new generation of components, like intelligent edge gateways, which are providing enterprises with the option to address computing challenges by performing critical data analytics close to endpoints at the edge of the network.

Several functionalities are supported:

- They can aggregate different types of sensor data and normalize them into standard IP traffic that is well-understood by IT.
- an intelligent edge gateway has the processing capacity to perform additional analytics in real or near-real time to make data-driven decisions as close to the data generation as possible
- Performing analytics on the gateways helps reduce network bandwidth cost because only meaningful information needs to be sent to the next tier, whether it is another gateway, the datacenter, or cloud.
- Moreover, Embedded CEP can communicate with the analytics modules placed on the server side and bring the global context close to the edge, by opening new possibilities for dynamic monitoring/processing on the edge

2. Summary of Main Functionalities

The main functionalities are:

- Manage CEP patterns (create, edit, delete)
- Manage input streams (select inputs for patterns, connect them)
- Enable connection to input streams (Bluetooth, …)
- Detect a relevant situation (CEP pattern)
- Define output events and actions

Extended functionalities:

- Store events locally
- Connect to broker
- Upload events to server
- Publish patterns to server
- Import patterns/events from server

QoS:

- Ensure privacy/security of data services

3. BEinCPPS Specific Developments

The entire component is a specific development for the project

µCEP engine is based on alarm model from the mobile pattern editor.

Patterns are made of Complex and Basic nodes. Main difference between them is that Complex alarm node can have as descendant other alarm nodes and have compare operator for the descendants and basic alarm node has event with properties. µCEP engine is intended to work only with this model of patterns, it will not have pattern description language.

µCEP has been implemented using finite state machine (FSM) model. A complex event patterns is represented as a binary tree of finite state machines. There can be 4 different types of finite state machines based on type of nodes, Complex and Basic, and based on operators that Complex node supports, *AND*, *OR* and *FOLLOWED BY*.

It is implemented in Java and can be deployed in different platforms.

4. HW/SW Prerequisite

HW prerequisite: Raspberry Pi 3

SW prerequisite: Java

Specification: A 1.2 GHz 64-bit quad-core ARMv8 CPU 802.11n Wireless LAN Bluetooth 4.1 Bluetooth Low Energy (BLE). Like the Pi 2, it has also: 1GB RAM 4 USB (101KB).

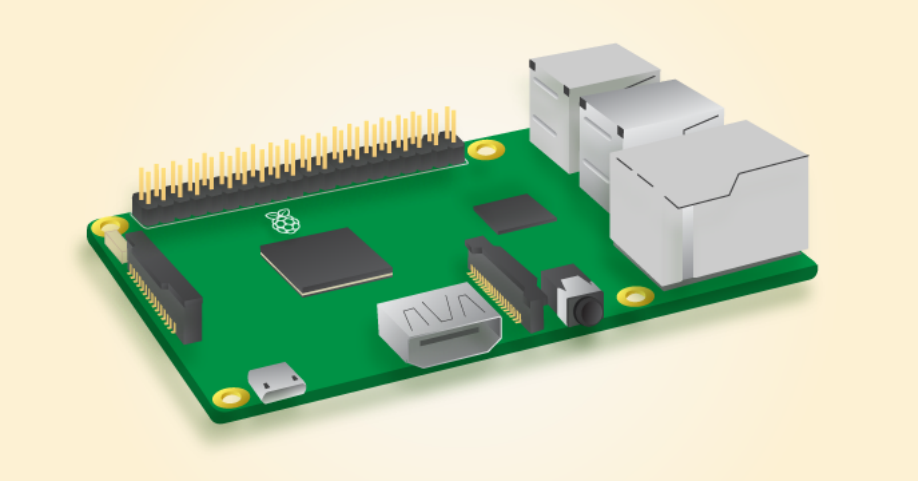


Figure 8: BEinCPPS Embedded CEP

5. Installation Instructions

This section describes usage of raspberry pi machine for deploying and running CEP application and generating alarm.

Pre-requirements

Since µCEP application is Java based application, only requirement for Raspberry is that it has installed operating system with java virtual machine. The suggestion is Linux based OS. Also Raspberry should have network connection for deploying application and sending alarms.

Pattern deployment

Patterns are part of application and they are inseparable. Pattern (application) deployment can be done through remote control protocols, for example SSH. After accessing Raspberry Pi through SSH, jar file is copied on it and started using java. If it is Linux machine application is run using nohup in order to run in background.

Sending alarms

Raspberry Pi must be connected to internet in order to be able to send alarms. Alarm sending is usually achieved through message broker. Application has client for broker which connects and sends new alarms when they are detected.

Event Processing

When event is received it is passed to root FSM of the pattern which forwards it to child machines. Every machine can have update listener which is called when final state is reached.

6. User Manual

No user manual publicly available

7. Developers' Guide

No developers' guide publicly available

8. Examples

The following are two examples of usage:

Events: Event1, Event2
Pattern: (E1 & E2) -> E1 (& = AND, -> = Followed By)
Received: Event2, Event1, Event1, Event2, Event1, Event1, Event2

Events: Event1, Event2

Pattern: (E1 & E2) -> (E1 & E2)

Received: Event1, Event2

9. Licensing

Open Source Apache License, Version 2.0

Appendix – Factsheet 2: TSN-enabled 4diac rte

Innovation Action Project

HORIZON 2020 - EU.2.1.5. - Ref. 680633

D2.8 - Smart Systems Platforms Federation

Factsheet #2

Relevant for TSN-enabled 4diac rte Component

Lead Author: Ben Schneider [fortiss]

With contributions from: Lăcrămioara Aştefănoaei, Alois Zoitl [fortiss]

1. Short Description of the Component

Eclipse 4diac™ (eclipse.org/4diac) is an open source project developed at fortiss. It is composed of the 4diac™ integrated development environment (4diac-IDE) and the 4diac runtime environment (4diac-RTE, forte):

- 4diac-ide is an extensible, IEC 61499 compliant engineering environment for distributed control applications. The modelled applications can be downloaded to distributed field devices
- forte is a small portable implementation of an IEC 61499 runtime environment targeting small embedded control devices (16/32 Bit), implemented in C++. forte provides a flexible communication infrastructure via so called communication layers.

IEC 61499 based systems follow an application centric design, which means that the application of the overall system is created at first. Each 4diac™ application is created by interconnecting the desired function blocks (FB) in terms of a function block network (FBN). As soon as the hardware structure is known it can be added to a project's system configuration and the already existing application can be distributed onto the available devices.

4diac's communication capabilities have been improved with a new extended IP communication layer with TSN support. The advantages of TSN are:

- **Convergent network:** real-time critical and uncritical traffic are transmitted on a single network interface, hence better bandwidth utilization and reduction of cabling effort.
- **Interoperability:** standard Ethernet removes the need for dedicated fieldbuses
- **Determinism and real-time:** TSN guarantees that packets are transmitted correctly and fulfil real-time requirements.

The communication layer implements publish-subscribe connections via Service Interface Function Blocks (SIFB) which are extended by a VLAN ID and the message priority, represented in the ID data input parameters of the publish FB.

2. Summary of Main Functionalities

Deterministic and real-time communication via publish/subscribe Service Interface Function Blocks implemented by a new communication layer in 4diac that supports TSN specific configuration parameters.

3. BEinCPPS Specific Developments

The new 4diac TSN enabled IP communication layer can send UDP multicast packets on a specific VLAN with a specified priority. The correct tagging with VLAN ID and priority is used by the TSN enabled network to transmit packets according to their real-time requirements. The different VLANs are related to different forwarding and scheduling strategies of the TSN network.

An example of a TSN enabled PUBLISH FB is shown in Figure 9.

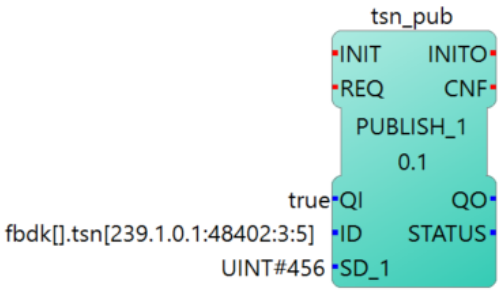


Figure 9: TSN enabled publish SIFB in 4diac

In 4diac, the SIFB's interface remains the same in order to keep the usability of the user interface as easy as possible. The ID data input parameter of the SIFB is used to select a specific network stack and provide configuration parameters for the communication layers. With the new TSN communication layer the configuration string is as follows:

fbdk[].tsn [<ip>:<port>:<vlan_id>:<prio>]

fbdk is responsible for marshaling the input data into a message following an ASN.1 encoding defined in IEC 61499-1 annex F. TSN specifies that the new TSN communication layer of 4diac should be used for this message stream with properties described as follows:

- **ip:** The IP address for UDP unicast or multicast
- **port:** The UDP port used for the data transmission
- **vlan_id:** The VLAN ID that the stream is assigned to
- **prio:** The priority of the VLAN

There is no need to change the subscribed SIFB interface for TSN related traffic, as the subscriber as no influence on the VLAN or priority of a specific traffic stream. It is only interested in IP address and port of the stream.

4. HW/SW Prerequisite

This paragraph summarizes the hardware and software prerequisites needed for successfully deploying 4diac's TSN communication layer.

Software:

- Implementation of IEEE 1588 or IEEE 802.1AS for time synchronization (e.g., the Linux PTP Project)
- Driver for IEEE 802.1Q for VLAN tagging

Hardware:

- A TSN capable switch (e.g. TTTech Akro 6/0 TSN, TTTech Hermes 0/4 TSN, NXP LS1021ATSN, Belden RSPE35 with Alpha Firmware, …)
- A general purpose computer, embedded PC, or PLC that is supported by forte (e.g., Wago PFC 200, Revolution Pi, …)
- *Optional:* You could use hardware supported timestamping and VLAN tagging that are supported by some NICs like the Intel I210 or CPUs like the AM3358 used for example, by the BeagleBone Black

5. Installation Instructions

This section describes how to compile forte (Linux / Cygwin toolchain) with the new TSN communication layer.

```
$ git clone git://git.eclipse.org/gitroot/4diac/org.eclipse.4diac.forte.git
$ cd org.eclipse.4diac.forte
$ git checkout develop
```

Create a bin folder, invoke CMake and configure the following CMake variables:

```
$ mkdir –p bin/posix
$ cd bin/posix
$ cmake ../.. -D FORTE_ARCHITECTURE=Posix -DFORTE_COM_TSN=ON
```

Compile forte using:

```
$ make
```

6. User Manual

A user manual will be available on the 4diac homepage https://www.eclipse.org/4diac/en_help.php under "Using Communication Protocols/TSN"

7. Developers' Guide

A developers's manual is available on the 4diac homepage https://www.eclipse.org/4diac/en_help.php under "Communication Architecture".

8. Examples

Figure 10 shows a picture of a best effort and a TSN enabled traffic stream. The ID parameter fbdk[].tsn[239.0.0.1:61499:3:5] shows an example configuration for a UDP stream with multicast group IP address 239.0.0.1 on port 61499 which is mapped to VLAN 3 with priority 5.

The TSN enabled network handles packets received on VLAN with ID 3 according to its configuration and guarantees the transmission of the UDP packets and the timing requirements. If a disturbance is added to the network which floods the net with generated bulk traffic, one would encounter packet losses on the best effort stream and no losses on the TSN enabled UDP stream.

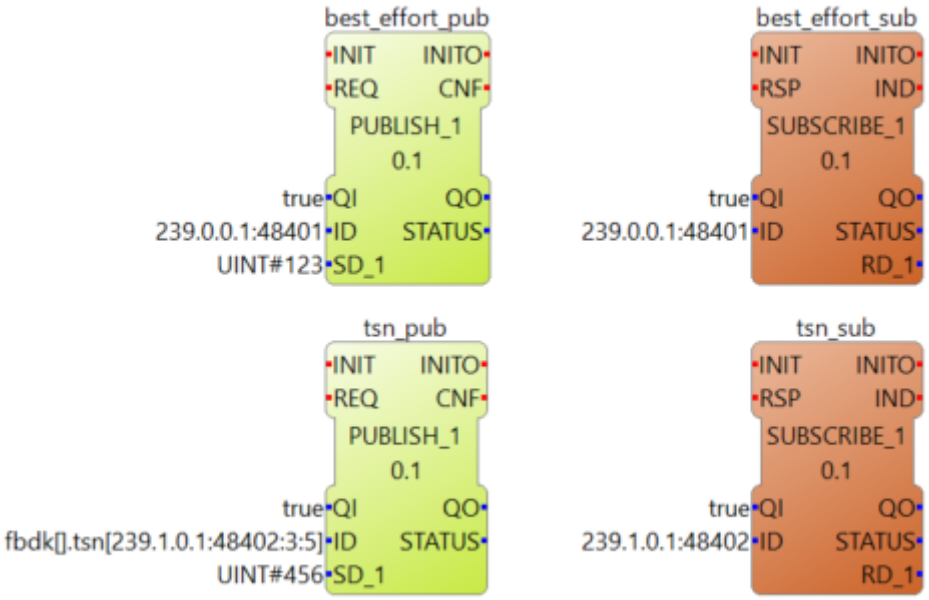


Figure 10: Example application with a best effort and a TSN enabled data stream

9. Licensing

Forte and its new TSN communication layer is open source licensed under EPLv1.0, which is accessible here: http://git.eclipse.org/c/4diac/org.eclipse.4diac.forte.git/tree/epl-v10.html

Appendix – Factsheet 3: WSN for CPPS

Innovation Action Project

HORIZON 2020 - EU.2.1.5. - Ref. 680633

D2.8 - Smart Systems Platforms Federation

Factsheet #3

Relevant for WSN for CPPS Component

Lead Author: David Todoli [ITI]

With contributions from: -

1. Short Description of the Component

The Wireless Sensor Network for CPPS, in the form of firmware and software modules for compatible hardware, is designed to provide wireless communication capabilities to sensors for non-critical actuators and enable them to interact with other systems. To achieve this functionality, multi-hop ad hoc networks are not isolated, self-configured network, but rather become an extension of wired infrastructure networks. A common requirements across such applications for the Industrial Internet of Things is for WSNs to deliver both low power and wire-like reliability and to do so across a broad spectrum of network shapes, sizes and data rates. A first version of this component and corresponding factsheet can be found in the deliverable D2.7. In this second iteration, a series of enhancements and changes has been introduced, mainly focused on improving the range of compatible devices and hardware, as well as new sensors drivers, and more importantly, user oriented interfaces and deployment options to enhance the user experience, minimizing network deployment time.

2. Summary of Main Functionalities

Following is a reminder of the main functionalities of the component:

1. Wireless IIoT: wireless communications allow working without infrastructures, with a lower cost of installation and maintenance. The provided WSN for CPPS includes:
    - Full Protocol stack design offering a robust and resilient wireless communication for harsh industrial applications, including mechanisms to avoid interferences and noise.
    - Deterministic MAC, scheduling algorithms for self-healing multi-hop networks.

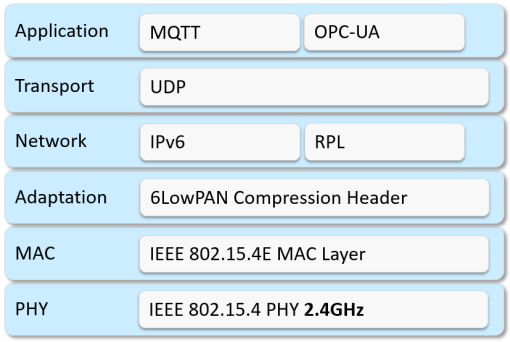


Figure 11: WSN4CPPS protocol stack

2. Wireless CPS with Deploy&Forget: these Deploy & Forget systems absorb all the complexity of deploying wireless networks, like preliminary analysis, RF planning, etc.:
    - Quality link estimators and network analysis tools facilitate deployment even for non-technical personnel.

- Smart, dynamic and autonomous routing resilient to network changes.
- Sensor and application configuration via different HMIs.

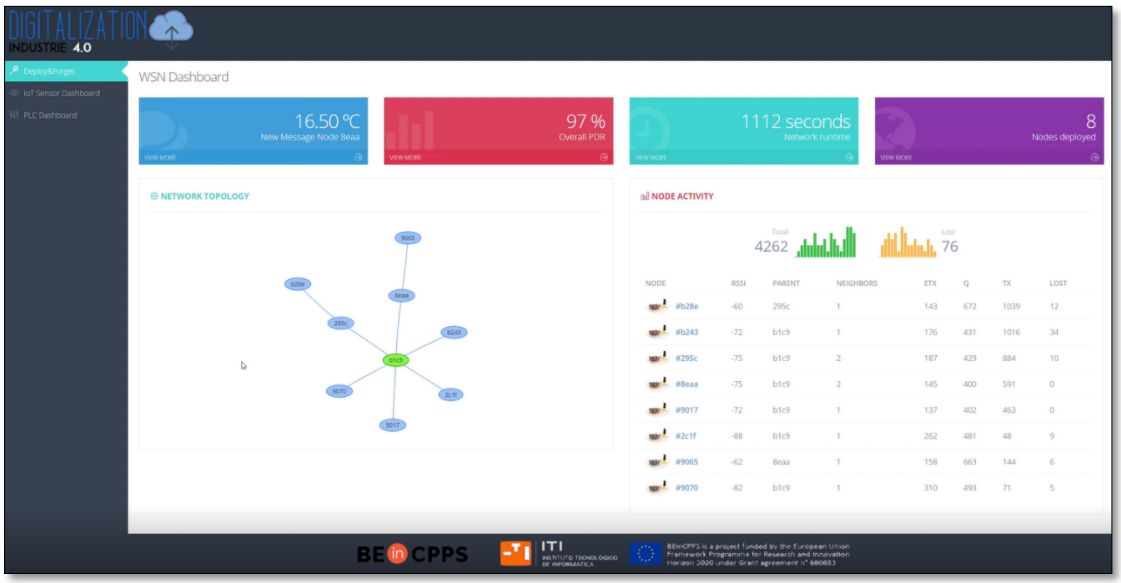


Figure 12: Screenshot of WSN4CPPS deployment HMI tool

3. I4.0 Architectures: Ethernet in plant level enables a new communication channel for all systems. New architectures use semantic languages that integrate seamlessly between machines and technologies as Cloud & Big Data.
    - Integration with I4.0 oriented architectures (FiWare, OpenIoT, …), through the WSN Gateway, enabling Plug&Play and interoperability features for WSN CPPS applications.
    - Integration with deterministic wired network (TSN) in floor plant, for soft real-time and QoS requirements depending on the use case.

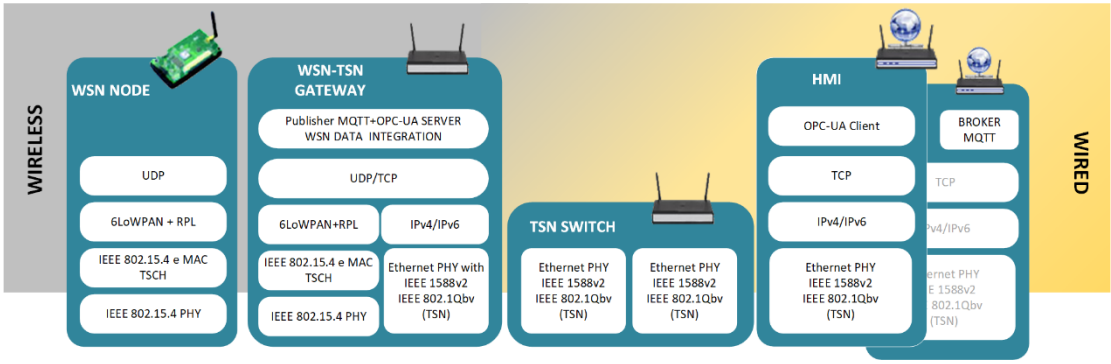
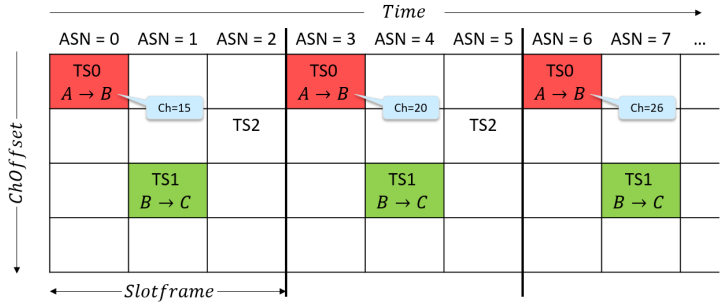


Figure 13: Floor plant communication architecture from WSN nodes to other devices or users, including to higher layer agents and wrappers

3. BEinCPPS Specific Developments

As introduced in the previous version of the component, the TSCH mechanism provided by the IEEE 802.15.4e standard for WSN provides better reliability and

bandwidth for wireless communication by using a combination of a channel hopping mechanism (mitigates channel fading effects and noise) and time slotted medium access, allowing different pair of nodes (a link in a multi-hop network) to have guaranteed bandwidth without interfering other links operating at the same time slot. The issue with this TSCH mode is that the scheduling of this time slots/channels is left for the user to implement. More information on these mechanisms can be found in the previous deliverable D2.7.



$$CH = HoppingSeqList[(ASN + ChOffset) mod (HoppingSeqLength)]$$

Figure 14: TSCH bi-dimensional MAC scheduling

At the start of the BEinCPPS project, the available implementation of the TSCH and upper layers like 6LoWPAN was rather limited. The first version of the component included basic configurations and tools to operate a WSN TSCH network making use of the above mechanisms (TASA algorithm or minimal TSCH configuration). At that time, the list of compatible hardware and OS was also restricted in order to use this protocols.

The new release of firmware and tools provided by the WSN4CPPS module of the Smart Systems platform enables a wider set of hardware by using Contiki O.S, which provides support for many more devices. All firmware files have been migrated and adapted.

Also, the scheduling tools have been replaced by the ORCHESTRA autonomous algorithm, which enables each node to set its own MAC schedule by using the smart routing protocol RPL information (a node knows with whom it has to be synchronized and calculates time slots and channel accordingly), and generates 3 different scheduled slotframes according to priority of the message (synchronizing beacons, routing and control messages, and application data). This is done without any interaction with the user, avoiding the need for planning tools and node's configurations.

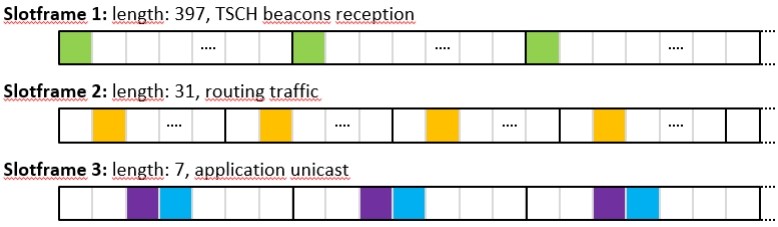


Figure 15: Orchestra autonomous scheduling algorithm 3 levels/slotframes

The updated component also includes a revised version of the smart routing metrics, and a tool to extract communication quality information to aid the user during deployment and maintenance of the WSN. The Quality Estimator, through statistics and fuzzy logic methods, combines 3 different metrics that address different issues in a wireless network, and resolves the best routes and paths for each node. Each metric (μ_x) can be weighted (α_x) to adapt better to the application, with predetermined weights set uniformly to balance all 3 metrics:

$$Quality\ Estimator = \alpha_{RSSI} \cdot \mu_{RSSI} + \alpha_{ETX} \cdot \mu_{ETX} + \alpha_{HOPS} \cdot \mu_{HOPS}$$

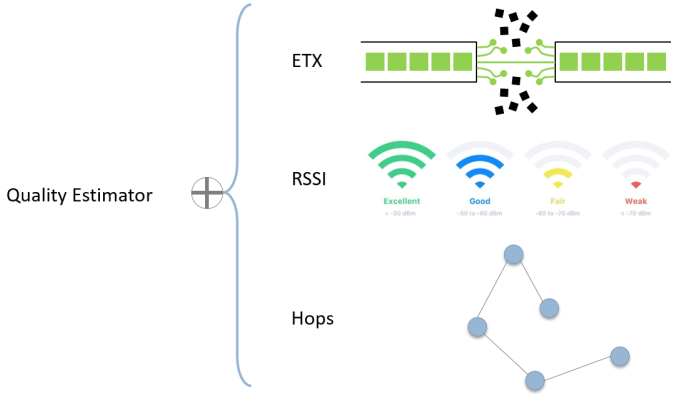


Figure 16: Metrics combined for a smarter routing algorithm, selecting best quality paths

The calculated Quality estimator output feeds the node's configurable LEDs in order to show, via a color code, the health or quality of the connection in the actual node's position, so during deployment, the operator can easily, without specialized training and tools, locate the nodes in optimum conditions.

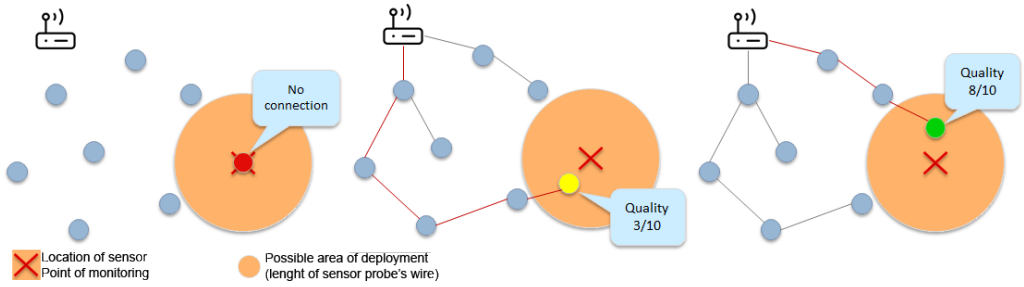


Figure 17: LED interface in nodes shows connection quality, allowing faster and optimal deployment

Regarding the interaction with architecture's higher layers, the WSN4CPPS provides a Gateway that collects WSN node's sensor data and enables different channels or APIs compatible with the BEinCPPS information bus, in this case through an OPC-UA endpoint server, or via an MQTT publisher.

Lastly, the gateway also enables a web application or interface supporting the deployment, where the operator can check in real time the network information and topology and check the communication to external servers through the APIs. This interface currently supports the WSN network discovery and MQTT and OPC-UA data flow check. In following weeks, the modules to configure/update configurations will be available as well.

An enhanced version of this interface has been developed for WP3 real world deployments, enabling, at floor plant level, M2M communication and integration of legacy devices such as PLCs, but this requires personalized configurations and includes many devices out of the scope of the component, so it is not provided as part of the WSN4CPPS asset.

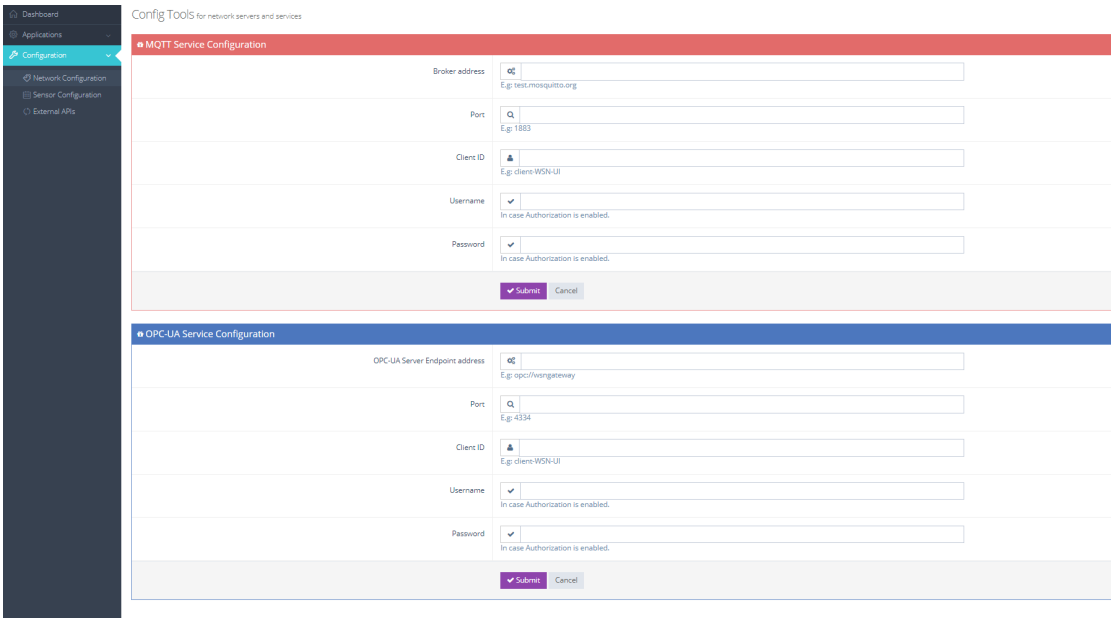


Figure 18: Deployment HMI tool screenshot

4. HW/SW Prerequisite

There are certain prerequisites needed to deploy the WSN component in a selected use case:

- Compatible hardware with radio support for standard IEEE 802.15.4e and Contiki SO. (see a recommended short list below)
  - Zolertia RE-mote (http://zolertia.io/product/hardware/re-mote)
  - OpenMote CC2538 (http://www.openmote.com/shop/openmotecc2538.html)
- Gateway device that enables WSN nodes to connect with external LANs/Internet. Typically, an embedded PC with an IEEE802.15.4e interface (may be another mote), which acts as WSN root node, and another network interface (wired or wireless). Linux based OS.
- Batteries and recharging infrastructure (USB cables).
- Computer with USB slots available and Linux OS for programming and updating node's software. It can be the same device used as gateway.
- C programming language compiler.
- Node.js installed.
- Sensor interfaces: temperature probe, accelerometer, current clamp.

5. Installation Instructions

Every module of software can be installed in the same machine, which will act as gateway (may be a BeagleBone, Raspberry PI or similar).

For installing Node.js, please visit:

- https://nodejs.org/en/download/package-manager/

With Node.js installed, the gateway also requires the related libraries to enable MQTT and OPC-UA in the gateway software. To install this, please follow the commands:

```
$ mkdir gw
$ cd gw
$ npm init                          # create a package.json
$ npm install node-opcua -save
$ cd node_modules/node-opcua
$ npm install mqtt -save
```

Then, download the file wsn-server.js by visiting

- https://dbox.iti.upv.es/oc/index.php/s/1gj6aQy3RaUgsMx

And copy it to:

```
$ mv wsn-server.js gw/node_modules/node-opcua
```

Also, place in home folder the following scripts:

- https://dbox.iti.upv.es/oc/index.php/s/kWqcCZjg56Xx9Tx
- https://dbox.iti.upv.es/oc/index.php/s/OPeRc83xIcnO4VU

For installing the Contiki OS files needed to update the devices, the user needs to download or clone the zip file provided, which contains a folder with a personalized version of Contiki OS (beware that this is not the official release, but the modified one to include the developed enhancements and drivers). It can be found at:

- https://dbox.iti.upv.es/oc/index.php/s/UMZAaSKxIhmlr80

Finally, if the user wants to run tests to check the WSN connectivity and data flow, a web interface can be installed in the gateway by installing a web server in the device:

```
$ sudo apt-get install nginx
```

And copy the following file in the corresponding folder:

- https://dbox.iti.upv.es/oc/index.php/s/vRL5ZUpUd4iV3NM

```
$ sudo unzip wsn4cpps-hmi.zip
$ sudo mv -r wsn4cpps /var/www/html
```

6. User Manual

Following the next simple steps will start the WSN in its default configuration

WSN nodes firmware and environment

- First step: install the compiler if not present

```
wsn4cpps@vm:~$ sudo apt-get update
wsn4cpps@vm:~$ sudo apt-get install gcc-arm-none-eabi gdb-arm-none-eabi
```

- Second step: download and unzip the modified Contiki folder

```
wsn4cpps@vm:~$ wget https://dbox.iti.upv.es/oc/index.php/s/UMZAaSKxIhmlr80
wsn4cpps@vm:~$ unzip contiki.zip
```

- Third step: Program end devices with provided application, changing TARGET=XXXX
  - **RE-MOTE: TARGET=zoul**
  - **OpenMote-CC2538 : TARGET= openmote-cc2538**

NOTE: this commands may require root privileges. In this case, run commands with 'sudo'.

```
wsn4cpps@vm:~$ cd contiki/examples/ipv6/rpl-udp/
wsn4cpps@vm:~/contiki/examples/ipv6/rpl-udp$ make TARGET=zoul savetarget
saving Makefile.target
wsn4cpps@vm:~/contiki/examples/ipv6/rpl-udp$ make udp-client.upload
```

Output, if successful, will end with a log similar to:

```
arm-none-eabi-objcopy -O binary --gap-fill 0xff udp-client.elf udp-
client.bin
Flashing /dev/ttyUSB0
Opening port /dev/ttyUSB0, baud 500000
Reading data from udp-client.bin
Cannot auto-detect firmware filetype: Assuming .bin
Connecting to target...
CC2538 PG2.0: 512KB Flash, 32KB SRAM, CCFG at 0x0027FFD4
Primary IEEE Address: 00:12:4B:00:06:0D:B3:1C
Erasing 524288 bytes starting at address 0x00200000
    Erase done
Writing 516096 bytes starting at address 0x00202000
Write 8 bytes at 0x0027FFF8F00
    Write done
Verifying by comparing CRC32 calculations.
    Verified (match: 0x2714e1d4)
rm udp-client.co obj_zoul/startup-gcc.o
```

- Fourth step: follow previous step but selecting the firmware code and device to play the WSN root role (this means the node connected to the WSN gateway):

```
wsn4cpps@vm:~$ cd contiki/examples/ipv6/rpl-border-router/
wsn4cpps@vm:~/contiki/examples/ipv6/rpl-udp$ make TARGET=zoul savetarget
saving Makefile.target
wsn4cpps@vm:~/contiki/examples/ipv6/rpl-udp$ make border-router.upload
```

Output, if successful, will end with a log similar to:

```
arm-none-eabi-objcopy   -O   binary   --gap-fill   0xff   udp-client.elf   udp-
client.bin
Flashing /dev/ttyUSB0
Opening port /dev/ttyUSB0, baud 500000
Reading data from border-router.bin
Cannot auto-detect firmware filetype: Assuming .bin
Connecting to target...
CC2538 PG2.0: 512KB Flash, 32KB SRAM, CCFG at 0x0027FFD4
Primary IEEE Address: 00:12:4B:00:06:0D:B3:1C
Erasing 524288 bytes starting at address 0x00200000
    Erase done
Writing 516096 bytes starting at address 0x00202000
Write 8 bytes at 0x0027FFF8F00
    Write done
Verifying by comparing CRC32 calculations.
    Verified (match: 0x2714e1d4)
rm udp-client.co obj_zoul/startup-gcc.o
```

With this, and the root node connected via USB, user can start the network with the command:

```
wsn4cpps@vm:~/contiki/examples/ipv6/rpl-border-router$ sudo connect-router
using saved target 'zoul'
fatal: Not a git repository: '../../../.git'
sudo ../../../tools/tunslip6 fd00::1/64
********SLIP started on ``/dev/ttyUSB0''
opened tun device ``/dev/tun0''
ifconfig tun0 inet `hostname` mtu 1500 up
ifconfig tun0 add fd00::1/64
ifconfig tun0 add fe80::0:0:0:1/64
ifconfig tun0

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-
00-00-00-00
          inet addr:127.0.1.1  P-t-P:127.0.1.1  Mask:255.255.255.255
          inet6 addr: fd00::1/64 Scope:Global
          inet6 addr: fe80::1/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

This starts a tunnel interface so the wired network can communicate with the wireless network. Nevertheless, the Gateway device needs to run additional software modules to enable the MQTT and OPC-UA flow, and the deployment HMI tools. For this, next stage is installing and running this software.

GATEWAY APIs and interfaces

The software developed to enable interaction with other devices and with the BEinCPPS information via MQTT or OPC-UA protocols, consists in node.js libraries and executables, which installation is explained in the following section. After that step, please run the script in the gateway home:

```
$ cd
$ sudo ./DEMO.sh
```

Finally, check if the web interface is up and running:

```
$ sudo service nginx start
```

If there were no problems during installations, accessing the gateway via its IP in the given url should show the introduced User Interface for deployment support.

http://GATEWAY_IP/wsn4cpps/

IMPORTANT NOTES

Some of the user libraries and modules are updated regularly and some dependencies may break, due to different software and libraries versions incompatibility. Also, debugging and optimizing code, as well as enabling new features required by use cases may introduce changes in the provided software at the given locations. In case the user finds any issue or problem, or for further questions, please contact ITI's staff to solve them.

7. Developer's Guide

For developers wishing to learn more on Contiki part, a starting point can be found on:

- https://github.com/contiki-os/contiki/wiki#Using

For developers wishing to learn more about OPC-UA and its node.js implementation, in order to develop different servers or clients that adjust to their use case:

- https://github.com/node-opcua/node-opcua/tree/master/documentation

8. Examples

Example of usages of the released component can be found in deliverable D3.2.

9. Licensing

The Contiki source code is released under a 3-clause BSD-style license. Under this license, Contiki may be used freely in both commercial and non-commercial systems as long as the copyright header in the source code files is retained. The full license reads as follows:

Copyright (c) 20**, X.

All rights reserved.

Redistribution and use in source and binary forms, with or without

modification, are permitted provided that the following conditions

are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Code developed in Contiki for the WSN4CPPS component still belongs to ITI, which is published under the Affero GPL (https://www.gnu.org/licences/agpl-3.0.en.html).

Appendix – Factsheet 4: OPC UA over TSN for CPPS

Innovation Action Project

HORIZON 2020 - EU.2.1.5. - Ref. 680633

D2.8 - Smart Systems Platforms Federation

Factsheet #4

Relevant for OPC UA over TSN for CPPS Component

Lead Author: Martijn Rooker [TTT]

With contributions from: -

1. Short Description of the Component

The purpose of the component is to provide seamless data communication between various industrial machines. OPC UA TSN (also named OPC UA over TSN) designates the combination of multiple technologies with which it is possible to transfer data vendor independent in industrial manufacturing with a uniform and open standard. OPC UA is a vendor-independent communication protocol designed for industrial use. Time-Sensitive Networking (TSN) is a further development of the IEEE Ethernet standards. Together, they aim to offer the first truly vendor-independent real-time Deterministic Ethernet communication standard.

2. Summary of Main Functionalities

The OP UA over TSN development consists out of two main parts, which provide the main functionalities of the technology. The main functionality of OPC UA over TSN is providing seamlessly shared information between all kinds of machines, devices and sensors in real-time. The individual technologies (Time-Sensitive Networking and OPC UA Publish/Subscribe) required for this functionality and their characteristics are described in more detail below.

Time-Sensitive Networking (TSN)

Time-Sensitive Networking is a set of IEEE 802 Ethernet sub-standards that describe several mechanisms for improved or even guaranteed real-time delivery of Ethernet traffic. TSN defines the first IEEE standard for time-triggered message forwarding in a switched Ethernet network, and therefore fully deterministic real-time communication within the 802 suite of standards. TSN achieves deterministic real-time communication over Ethernet by using global time and a schedule which is created for message paths across multiple network components. By defining queues which transmit their messages based on a time schedule, TSN ensures a bounded maximum latency for scheduled traffic through switched network. In control applications with strict deterministic requirements, such as those found in autonomous and industrial domains, TSN offers a way to send time-critical traffic over a standard Ethernet infrastructure. This enables the convergence of all traffic classes and multiple applications in one network.

Currently, the following TSN standards are supported in the available components:

- *IEEE 802.1Qbv – Scheduled Traffic*: the core of TSN is a time-triggered communication principle, known in TSN as the “Time-Aware Shaper” (TAS), which deterministically schedules traffic in queues through switched networks (see Figure 19). This is standardized in IEEE 802.1Qbv. The TAS enables to control the flow of queued traffic from a TSN-enabled switch. Ethernet frames are identified and assigned to queues based on the priority of the frames. Each queue is defined within a schedule, and the transmission of messages in these queues is then executed at the output ports during the scheduled time

windows. Other queues and ports are then blocked, thereby removing the chance of scheduled traffic being interrupted by non-scheduled traffic.

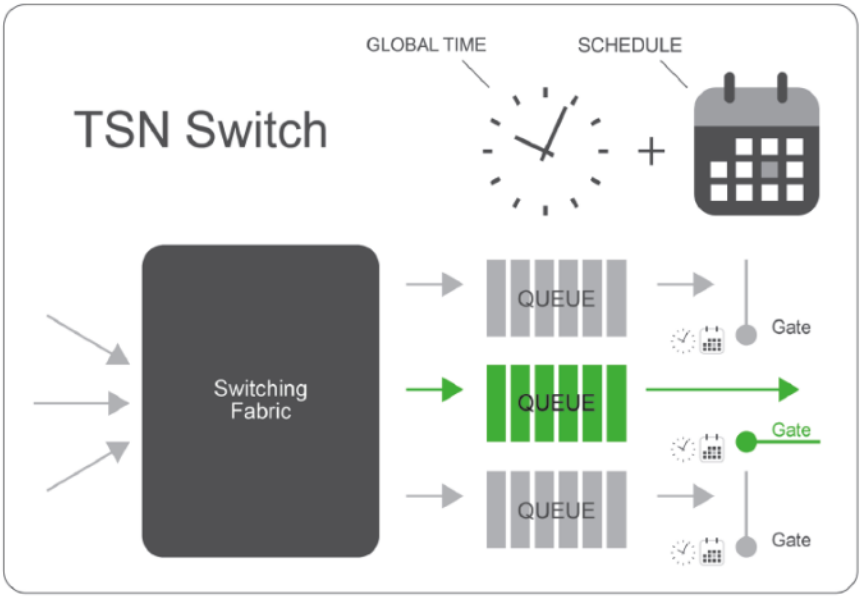


Figure 19: TSN queues and transmission gates

- *IEEE 802.1Qbu – Frame Pre-emption*: This standard describes the concept that the TAS (IEEE 802.1QbV) avoids transmission jitter by blocking lower priority queues in advance of the transmission point of the critical frame. In cases, where minimal latency for scheduled messages is desired, the TAS mechanism may not be an optimal solution. Therefore, on links where pre-emption as defined by this standard is supported, the transmission of standard Ethernet frames can be interrupted in order to allow high-priority frames to be transmitted, and afterwards resume the transmission of the interrupted frame (without discarding the message).
- *IEEE 802.1AS – Timing and synchronization*: Clock synchronization is a vital mechanism for establishing deterministic communication with bounded message latency in TSN. The IEEE 802.1AS standard creates a profile of the IEEE 1588 PTP synchronization protocol for TSN. This profile will enable clock synchronization compatibility between different TSN devices. Additionally, IEEE 802.1AS standardizes the use of multiple grandmaster clocks as well as the possibility to make multiple connections to these grandmaster clocks. Replication of grandmaster clocks results in shorter fail-over times in cases when a grandmaster clock becomes faulty. Furthermore, IEEE 802.1AS support multiple synchronized clocks, enabling timestamping of events such as production data or measurements, and the synchronization of applications such as sensors, actuators and control units.

OPC UA Publish/Subscribe (Pub/Sub)

OPC UA is a cross-platform service-oriented architecture and data exchange technology that enables safe and reliable vendor- and platform communication and is widely used in the industry.

OPC UA has been supporting the Client/Server approach for communication. In this situation a client makes a request and receives an answer from a server (response). Unfortunately, this system will reach quickly its boundaries when the network becomes too large, meaning having to many participants that are sending data over the network. For each communication, a client-server link has to be established. In the case of Industry 4.0 and IIoT, where everything is connected with everything, this will lead to an explosion of the amount of links and the Client/Server approach will not suffice. The Publish/Subscribe (Pub/Sub) model, on the other hand, enables one-to-many and many-to-many communication. A server sends the data in the network (Publish) and each Client can receive this data (Subscribe). The difference is depicted in Figure 20.

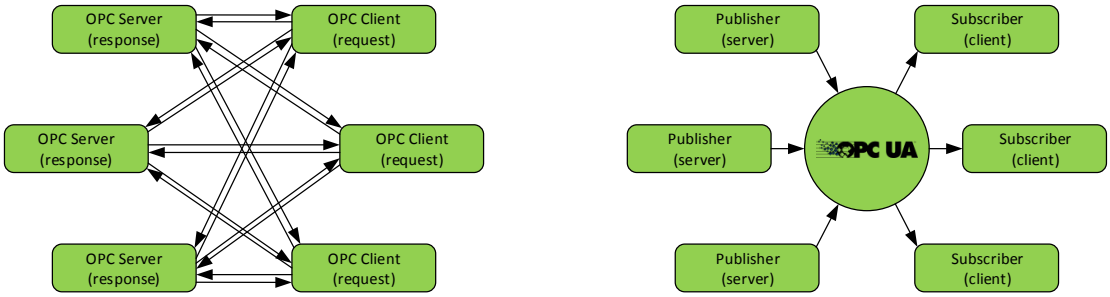


Figure 20: Client/Server vs Publish/Subscribe

3. BEinCPPS Specific Developments

The merging of OPC UA Pub/Sub with the Time-Sensitive Networking (TSN) is more than just adding the two parts together. Currently, the available OPC UA Pub/Sub stacks are not TSN aware and therefore cannot make use of the substantial advantages of TSN (e.g. synchronization, convergence, robust delivery, etc.). To solve this, modifications of the applied OPC UA stacks are mandatory, which is one of the biggest challenges identified within the BEinCPPS project with respect to OPC UA over TSN.

Additionally, TSN is also constantly under development. As mentioned before, TSN is a 802 suite of standards. These standard are unfortunately still under development and have not been finalized yet (see Figure 21). The further development of the TSN solutions is one of the priorities of TTTech within the BEinCPPS project and will at the end flow into their product portfolio.

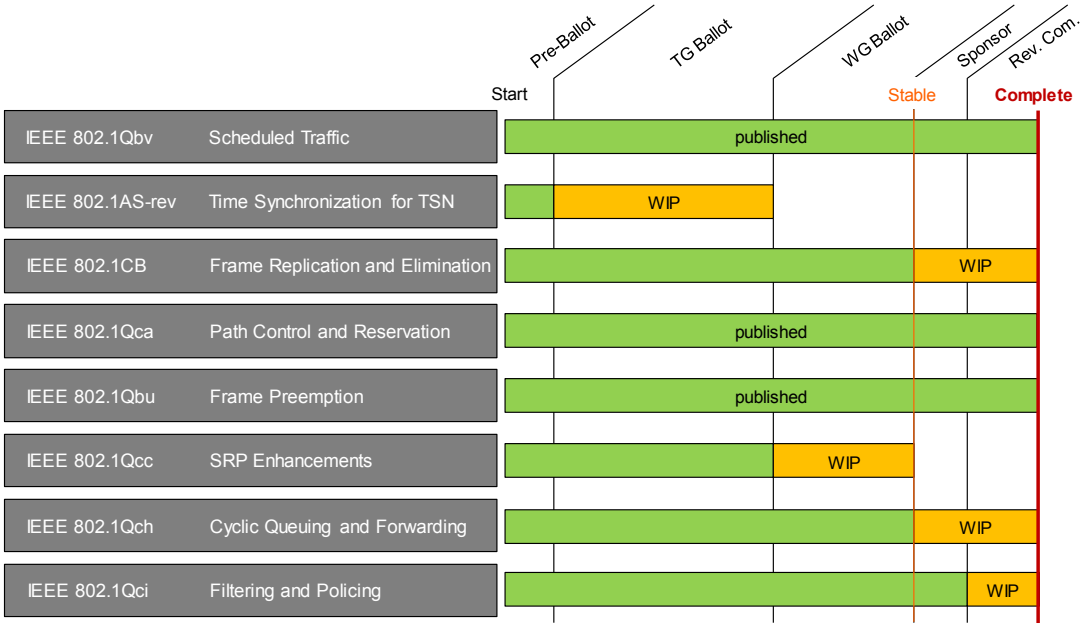


Figure 21: Current TSN Standardization Status (January 2017) (Source: http://www.ieee802.org/1/pages/tsn.html)

To integrate OPC UA with TSN, the OPC UA Pub/Sub stack is integrated with TSN using the User Datagram Protocol (UDP), as depicted in Figure 22). The advantage of connecting over UDP is that communication is not only able for SCADA systems, but also down to the device or the controller and a fully distributed control system can be developed based on OPC UA. This enables the transition from the old automation pyramid into the new architecture (Industrie 4.0), where communication is not only established within the different layers of the pyramid, but also between the different layers. For traditional OPC UA, there is a client-server architecture, where one part of the server provides the data and another part is doing the acquisition, which is performed always in a fixed point-to-point relation (see also section 2.2). Pub/Sub gives you the functionality to define a fixed time-window where data can be exchanged, using multi-point connections via UDP. This gives one a communication frame, which is addressed to many other PLCs instead of a single one. This current version of OPC UA over TSN support real-time communication, enabling transfer of messages within certain limited timeframes. Real-time communication here doesn't refer to being the first of fast in data transfer, but referring to determinism, which can be interpreted as having a time-window in which the communication has to be delivered.

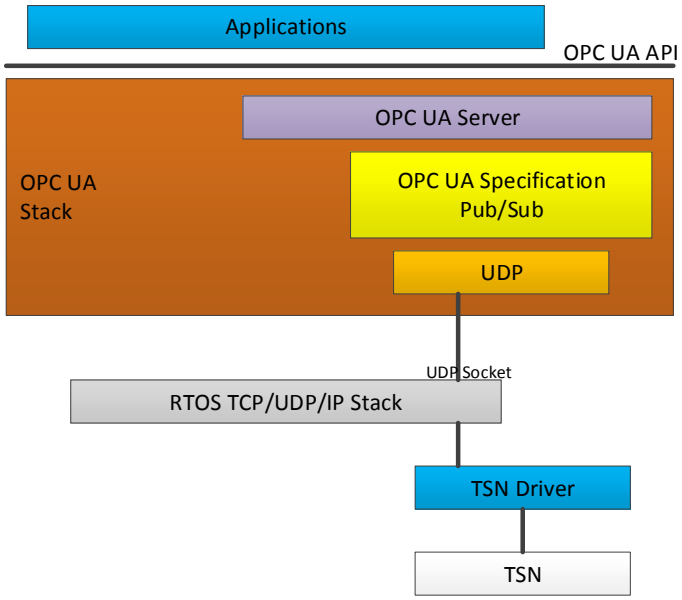


Figure 22: Real-time OPC UA Publish/Subscribe via UDP over TSN

Additionally, the combination of OPC UA Pub/Sub data exchange and TSN deterministic communication requires a mechanism that assigns the TSN schedules to published OPC UA data. This mechanism is known as the Pub/Sub TSN configuration broker. The broker can be interpreted as a network management function as well as an API for the OPC UA layer. Its task is to interface with OPC UA publisher and subscriber devices and the available TSN scheduler software. This interface ensures that three critical elements are available:

- The TSN scheduler knows the publishing interval of OPC UA data (and in some cases and offset within the publishing interval)
- The OPC UA publishing device knows when the network is ready (configured according to the schedule) so that it may start publishing
- The OPC UA subscribers are primed to receive data to which they are subscribed within the requisite timeframe.

Once the TSN scheduler has confirmed that is can assign a path through the network to each OPC UA Pub/Sub data stream requested via the broker, the broker then provides configuration relevant feedback to the OPC UA publishers and subscribers.

The main goal of the (hard) real-time communication development (by using OPC UA over TSN) is to improve the performance of the communication network. In the first version of this development, soft real-time was provided thereby providing round trip delay of around 15msec, without a fixed guarantee of delivery, resulting in a potential loss of messages.

In the final version of the OPC UA over TSN, a lower round trip delay has been achieved with a time of less than ½ msec, without jitter and with guaranteed delivery of the data. This enables precise machine-to-machine communication that can be used

for e.g. coordination between a robot and a tool mounted on it, or coordination of multiple conveyor belts.

4. HW/SW Prerequisites

The following hard- and software prerequisites are needed to deploy the OPC UA over TSN technology in selected use cases:

- Time-Sensitive Networking (TSN) Switch (e.g. https://www.tttech.com/products/industrial/industrial-iot/fog-computing-nerve/mfn-100/)
- TSN end nodes, including with a network interface card (NIC). This enables deterministic communication over dual redundant synchronous network channels for seamless redundancy management in high-availability real-time network. It supports three different traffic classes on the network (see also Figure 23):
    - Best-effort Ethernet traffic
    - Prioritized Ethernet traffic

Scheduled (time-triggered) traffic with hard real-time guarantee and transport-delay jitter in sub-millisecond range for mission-critical and safety-critical real-time applications.

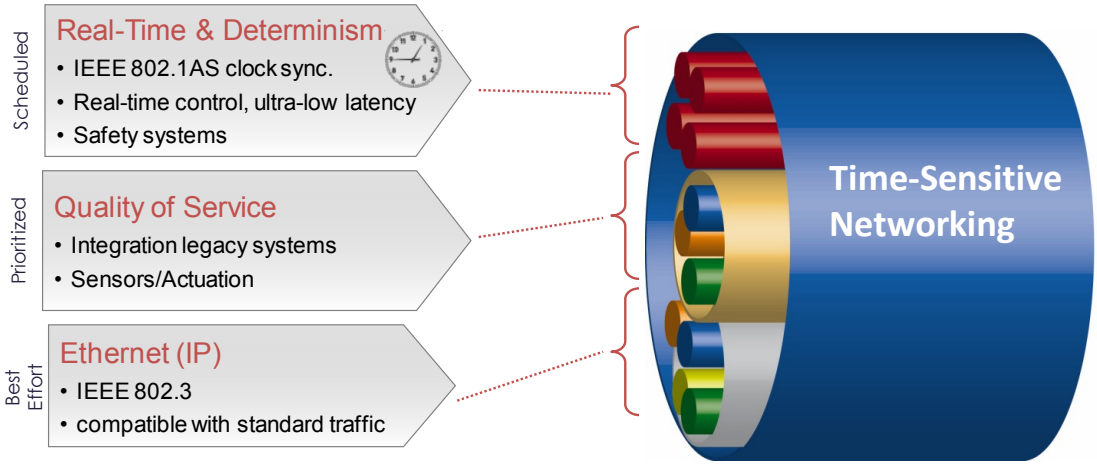


Figure 23: Traffic classes in Time-Sensitive Network

Software level

- RTOS (e.g. Linux Ubuntu 14.04) with RT patch
- RT OPC UA Stack (Binary version is provided by a leading supplier of OPC UA engaged in a development partnership with TTTech)

5. Installation Instructions

Installations instructions will be provided by TTTech.

6. User Manual

The user manual for the TSN-enabled switches together with the (real-time) OPC UA Pub/Sub implementation will be provided by TTTech with the hardware.

7. Developer's Guide

Developers's guide will be provided by TTTech.

8. Examples

Examples of usages of the released component can be found in deliverable D3.2. Additionally, an example of a TSN testbed where TTTech is heavily involved can be found under: https://www.tttech.com/news-events/newsroom/details/tttech-joins-tsn-testbed-with-the-industrial-internet-consortium/.

9. Licensing

Propriety Licensing by TTTech.