Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Intrinsic safety in distributed real-time systems: modeling, design and test of applications for the remote handling of mobile industrial equipment.

Ph.D. candidate
Giovanni Peserico

Advisor
prof. Stefano Vitturi

Director & Coordinator
prof. Andrea Neviani

Ph.D. School in
Information Engineering

Department of
Information Engineering
University of Padova

2022

Intrinsic safety in distributed real-time systems: modeling, design and test of applications for the remote handling of mobile industrial equipment.

Ph.D. candidate: Giovanni Peserico

Director: prof. Andrea Neviani
Advisor: prof. Stefano Vitturi

Ph.D. School in Information Engineering
Department of Information Engineering
2022

Abstract

Safety is defined as the absence of unacceptable risk. In this sense, safety systems are those that integrate methods to reduce the probability of risks to an acceptable level. Industrial transmission safety systems were developed starting from their ad-hoc networks, which were conceived to satisfy three essential requirements of this environment: real-time, deterministic performances; high reliability; and the ability to withstand extreme conditions. In this context, starting from a simple redundant algorithm, many safety protocols were created. For example, the most famous ones such as Profisafe, which was created for starting from Profibus and Profinet; FailSafeOverEthercat developed for Ethercat; CANOpen Safety developed for CAN, etc.
With Industry 4.0, with technology development and growth of available data, with the resulting born of Internet Of Thing (IoT) and in particular of Industrial Internet of Thing (IIoT), the industrial environment has seen the rise of new necessity, such as the enlargement of the production site, the decrease of wiring costs or, if possible, its complete elimination, the introduction of remote control and the ability to acquire and save a big amount of data. In this context the development of wireless technologies, in continuous evolution and optimization, allows the industrial networks to satisfy their new requirements. However, concerning safety, no safety protocols were developed and certified for a wireless network, since the necessity of determinism often collides with the lack of a sure, well-defined, transmission medium and with the use of complex algorithms. Despite that, the continuous growth and use of wireless systems, lead to favor for those protocols which were designed with a "black channel" approach. Indeed, these protocols were theoretically designed to be implemented without knowing the underlying layer, but in practice they require some minimal characteristics of the channel, consequently requiring the design of new stack and the identification of general characteristics useful to satisfy both safety that functionality requirements. In this complex coexistence of different technologies and of necessity of safety, with this thesis many technologies were studied, analyzed and implemented, highlighting features, uses and proposing possible improvements. Starting from the most famous Fieldbuses, many wireless technologies

were then analyzed, to finally reach those newest technologies developed for IIoT. Then the issue of safety over wireless was studied, implementing different possible stacks, acquiring and analyzing the obtained results and finally proposing some improvement and considerations.

Sommario

La sicurezza (safety) è definita come assenza di rischi inaccettabili. I sistemi di sicurezza sono quindi quelle strutture che integrano metodi per la diminuzione della probabilità di un rischio per renderlo accettabile. I sistemi di sicurezza riguardanti la trasmissione di dati nell'ambito industriale sono stati sviluppati a partire dalle rete progettate per tale ambiente, le quali sono caratterizzate da 3 caratteristiche fondamentali, ovvero le performance real-time, deterministiche ; l'alta affidabilità; e la capacità di resistere a condizioni estreme di lavoro. Partendo perciò da semplici algoritmi di ridondanza, sono stati creati dei protocolli di sicurezza dedicati. E' questo l'esempio dei più famosi protocolli di sicurezza industriali: Profisafe, sviluppato a partire da Profibus e Profinet; Fail Safe Over EtherCAT sviluppato, come dice il nome stesso, proprio per essere integrato in reti EtherCAT, CANopen Safety sviluppato per reti CAN etc...
Con l'avvento, però, dell' Industria 4.0, lo sviluppo delle tecnologie e il crescere dei dati a disposizione con la conseguente nascita dell' Internet Of Thing (IoT), il mondo industriale ha visto sorgere nuove esigenze come l'ingrandimento dei siti produttivo, la diminuzione dei costi dei cablaggi o la rimozione stessa, se possibile, il controllo da remoto e la necessità di acquisire e salvare un esponenzialmente crescente numero di dati. In questo contesto lo sviluppo di tecnologie wireless, sempre più evolute ed ottimizzate, ha permesso alle reti industriali di soddisfare queste necessità. Per quanto riguarda la sicurezza (safety), però, non sono stati sviluppati protocolli di sicurezza progettati e certificati per reti wireless, in quanto la necessità di determinismo spesso va a configgere con la mancanza di un mezzo fisico ben determinato e l'uso di metodi di ottimizzazione complessi. Nonostante ciò, la continua crescita dei sistemi wireless, ha portato a privilegiare lo sviluppo di quei protocolli di sicurezza progettati con un approccio definito a "canale nero", ovvero designati, in linea teorica, per poter cambiare i livelli sottostanti. In realtà a livello pratico e implementativo tali protocolli necessitano comunque alcuni requisiti fondamentali, andando a dipingere un approccio più a "canale grigio" piuttosto che a "canale nero". Inoltre il lungo iter di certificazione di qualsiasi stack differente dallo standard, va a impedire, o perlomeno rallentare, lo sviluppo di

questi nuovi protocolli. Tuttavia la strada tracciata da questa nuova filosofia ha portato allo sviluppo del un nuovo protocollo di sicurezza OPC UA safety, pensato anche per tecnologie wireless e che ha avviato il procedimento per la certificazione dello standard stesso; Inoltre numerose prove vengono continuamente fatte per la creazione di nuovi stack che comprendano protocolli wireless, che riescano a soddisfare sia requisiti di sicurezza che di funzionalità. In questo complesso connubio di diverse tecnologie e di sicurezza, con questa tesi si sono volute approfondire le diverse tecnologie studiate, analizzate e implementate nel corso di questi anni di dottorato, andando ad evidenziarne le caratteristiche, l'uso e i possibili miglioramenti che sono stati proposti. Partendo dai primi più famosi Fieldbuses, si sono poi analizzate numerose tecnologie wireless, per cercare di raggiungere infine quelle nuove tecnologie adatte all'IIoT. Si è poi quindi cercato di andare ad approfondire il tema della sicurezza su sistemi wireless, andando a fare un'effettiva implementazione di diversi possibili stack, acquisendo ed analizzando i risultati trovati e andando infine a proporre delle migliorie e delle considerazioni.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence.
AIFS	Arbitration Interframe Space.
ALARP	As Low As Reasonably Possible.
AOA	Angle Of Arrival.
AP	Access Point.
BPM	Burst Position Modulation.
BPSK	Binary Phase-Shift Key.
CAGR	Compound Annual Growth Rate.
CC	Conformance Classes.
CDF	Cumulative Density Function.
CPF	Communication Profile Family.
CPU	Central Processing Unit.
CRC	Cyclic Redundancy Check.
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance.
CTS	Clear To Send.
CW	Contention Window.
DCF	Distributed Coordination Function.
DIFS	DCF Interframe Space.

DIFSHCF Hybrid Coordination Function.

DMA Direct Memory Access.

DSSS Direct-Sequence Spread Spectrum.

DT Digital Twin.

E/E/PE Electrical/Electronic/Programmable Electronic.

E/E/PES Electrical/Electronic/Programmable Electronic Safety-related.

EDCA Enhanced Distributed Coordination Access.

EIFS Extended Interframe Space.

EtherCAT Ethernet for Control Automation Technology.

EU-OSHA European Agency for Safety and Health at Work.

FCC Federal Communications Commision.

FCS Frame Check Sequence.

FH Frequency-Hopping.

FSN Functional Safety Network.

FSoE Fail Safe over EtherCAT.

GSD General Station Descriptor.

HSE Health and Safety Executive.

HT High Throughput.

IBSS Indipendent Basic Service Set.

IFS Interframe Space.

IIoT Industrial Internet of Things.

IoT Internet of Things.

IP Internet Protocol.

IRT Isochronous Real Time.

ISO International Organization for Standardization.

LAN Local Area Network.

LLC Logical Link Control.

MAC Medium Access Control.

MCR Multicast COmmunication Relation.

MDP Markov Decision Process.

MIMO Multiple Input Multiple Output.

MPDU MAC Protocol Data Unit.

MSDU MAC Service Data Unit.

MSE Mean Square Error.

NAV Network Allocation Vector.

OFDM Orthogonal Frequency-Division Multiplexing.

OFDMA Orthogonal Frequency-Division Multiple Access.

OMNeT++ Objective Modular Network Testbed in C++.

OPC-UA Open Platform Communication-Unified Architecture.

OSHA Occupational Safety and Health Administration.

OSI Open Systems Interconnection.

PCF Point Coordination Function.

PDoA Phase Difference of Arrival.

PDU Protocol Data Unit.

PER Packet Error Rate.

PLC Programmable Logic Controller.

Profibus DP Profibus Decentralised Protocol.

Profibus PA Profibus Process Automation.

QoS Quality of Service.

REP Residual Error Probability.

RL Reinforcement Learning.

RRAA Robust Rate Adaptation Algorithm.

RT Real-Time.

RTS Request To Send.

SAP Service Access Points.

SDN Send Data No acknowledge.

SFRT Safety Function Response Time.

SIFS Short Interframe Space.

SIL Safety Integrated Level.

SNR Signal-to-Noise Ratio.

SRD Send and Request Data.

STW Short Term Window.

TCP Transmission COntrol Protocol.

TDoA Time Difference of Arrival.

TOF Time Of Flight.

TSN Time-Sensitive Networking.

TWR Two Way Ranging.

UDP User Datagram Protocol.

UDPc UDP with caching layer.

UP	User Priority.
Wi-Fi	Wireless-Fidelity.
WLAN	Wireless Local Area Network.
WSAN	Wireless Sensor/Actuator Network.
WSN	Wireless Sensor Network.

1 Introduction

1.1 Functional safety

First industrial safety concepts were introduced in Europe with the labor movement during the Industrial Revolution. During that period, workers formed unions and began to demand better working conditions leading the government organizations to a regularization of the workplace. Because of the diversity of the industry, the relative organizations developed safety industry-specific regulations independent of each other. Despite the born of these regulations, only more than a century later, in 1970, the U.S Department of Labor founded the Occupational Safety and Health Administration, which enabled the federal government to regulate safety in the workplace through new standards, published one year later, in 1971, which provided the baseline for safety and health protection in American workplaces. The corresponding European organization, the European Agency for Safety and Health at Work (EU-OSHA), was set up in 1994 by Council Regulation. However, each state had already started to regulate safety after the OSHA foundation. Indeed in 1974, the UK published the Health and Safety at Work Act, which led to the creation of what is now the UK's Health and Safety Executive, and then it was followed by other states. It was in 1998 that the first International standard concerning safety was published by the International Electrotechnical Commission consisting of methods on how to apply, design, deploy and maintain automatic protection systems called safety-related systems. This standard is the IEC 61508 and it is titled Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES). It introduced the concept of safety as intended today, indeed it is still the reference standard for functional safety, with a second edition published in 2010. The regulation introduced the concept that risk is a function of the frequency (or likelihood) of the hazardous event and the event consequence severity. The risk is reduced to a tolerable level by applying safety functions which may consist of E/E/PES, associated mechanical devices, or other technologies. Despite that, the zero risk can never be reached, only probabilities can be reduced, in particular the ones associated with non-tolerable risks (ALARP). In this sense, safety is defined as the absence of unacceptable risk, where unacceptable is correlated to the probability and the dangerousness of the risk. With continuous growth of digitalization and networking, one fundamental aspect of safety is to guarantee, within the required probability, the correct transmission of data. For this reason, a dedicated regulation was released: IEC 61784-3, which was published firstly in 2007. Despite that, different protocols designed for the transmission of safety data were released several years before. For example, specifications for safety communication over Profibus and Profinet were published already in 1999, creating the first safety protocol, which is Profisafe. This protocol was then included with the IEC61784-3 standard, becoming part of the regulation itself (IEC 61784-3-3). As it is possible to appreciate from the story of safety regulation, the standards are usually released after a necessity of the industrial environment or of the workers. In this context, safety does not always keep up with time. An essential example is the use of wireless networks. Indeed, with the born of the Internet of Things (IoT) paradigm and in particular of Industrial Internet of Things (IIoT), the industrial environment has seen the rise of new necessities, such as the enlargement of the production site, the decrease of wiring costs or, if possible, its complete elimination, the introduction of remote control and the ability to acquire and save a big amount of data. And on this basis, the development of wireless technologies, in continuous evolution and optimization, allows the industrial networks to satisfy their new requirements. However, for as concerning safety, no safety protocols were developed and certified for wireless networks, since the necessity of determinism often collides with the lack of a sure, well-defined, transmission medium and with the use of complex algorithms. Although, the continuous growth and use of wireless systems, lead to favor for those protocols which were designed with a “black channel” approach, as specified by the IEC61784-3. In these terms, these protocols were theoretically designed to be implemented without knowing the underlying layer, but in practice they require some minimal characteristics of the channel, defining a more “grey channel” rather than a “black” one. So, in this context, the implementation over a wireless channel is theoretically possible. For example, the new safety protocol, OPC UA Safety, was developed for wireless technology and it started the certification iter to become part of IEC61784-3; In addition, many research activities are proposed for the design of new stack and for the identification of general characteristics useful to satisfy both safety that functional requirements, and all these work will lead to the development of new part of the standard or a new one. Despite that, safety is still far from the continuous evolution and interconnection of new systems. According to Wijethilaka and Liyanage (2021), it is expected that the number of interconnected IoT devices will be 27 billion by 2024 so 5G, Real-time wired networks, and Wi-Fi 6 IoT will connect a massive number of smart devices and make a contribution to meet market demand through highly integrated services to stimulate new economics and social development (Vitturi, Zunino, and Sauter (2019); Cavalcanti, Perez-Ramirez, Rashid, Fang, Galeev, and Stanton (2019); T. Adame and Bellalta (November 2020); Dawy, Saad, Ghosh, Andrews, and Yaacoub (2017)). Moreover new devices are able to aggregate and categorize information for better management and use, for example, in Artificial Intelligence systems (Rekkas, Sotiroudis, Sarigiannidis, Wan, Karagiannidis, and Goudos (2021); Nguyen, Cheng, Ding, Lopez-Perez, Pathirana, Li, Seneviratne, Li, and Poor (2020)). While for safety systems is highly recommended to not use machine learning algorithms, as suggested by IEC 61508. In this sense, many steps must be done to bring research and safety closer.

1.2 Contribution

In this manuscript, we address the challenges concerned with the conjunction of functional safety networks and protocols with distributed measurement systems and real-time communications in Industrial Internet of Things ecosystems. In this context, in chapter 2, an overview of the existing industrial communication technologies is introduced. In particular, firstly, the most important fieldbuses are briefly presented. Then, common wireless technologies are investigated highlighting those features which are particular useful for Industrial Internet of Things. In detail, these systems were deeply analyzed and studied during the first year of my Ph.D., allowing me to have all the knowledge to confidently manage these technologies. After that, it was possible to implement them on real systems trying to propose some feasible improvements or additional features. These results are presented in chapter 3, where the obtained results with technologies such as Wi-Fi are proposed, with particular attention to its Rate Adaptation feature and possible application to sensor selection of wireless smart edges. Moreover a realistic

industrial simulator OMNeT++, which is a finite-discrete event simulator used for network reproduction, was analyzed, used and improved for Wi-Fi and also for Profinet. In addition, another wireless technlogy, the UltrawideBand one, was studied, implemented and improved. All the contribution on these arguments are also reported in Morato, Vitturi, Fedullo, Peserico, and Tramarin (2020), Peserico, Fedullo, Morato, Vitturi, and Tramarin (2020), Peserico, Fedullo, Morato, Tramarin, Rovati, and Vitturi (2021a), Peserico, Fedullo, Morato, Tramarin, and Vitturi (2022), Ballotta, Peserico, and Zanini (2022).
Subsequently to the focus over the industrial technologies and their evolution, a deep study of safety was done. After the analysis of functional safety and of safety communication, there was the tentative of unifying the previously obtained industrial communication knowledge, both theoretical that practical ones, with the safety environment. In detail, the safety protocol Fail Safe over EtherCAT (FSoE) was selected for the following works, thanks to: its black channel approach, which, as proposed, is the most adequate for the evolution of technologies, its quite simple machine-state, its robust countermeasurements against all possible errors and since it is part of the standard, and it is in continuous evolution. In this sense, in chapter **??**, a presentation of the safety is done, introducing various standards, with a particular focus on IEC61784-3 and, as anticipated, of FSoE. The implementation of the FSoE protocol on top of IEEE802.11 is considered in Chapter 4. After a detailed overview of the implementation, the extensive tests carried out on a prototype implementation are discussed. In particular there is a focus on the polling time, i.e. the time needed to complete an exchange of safety data, on the packet loss rate which also reflects the maximum achievable safety level and on the Safety Response Time (SFRT). Starting from the contributions proposed by Morato, Vitturi, Cenedese, Fadel, and Tramarin (2019), which revealed some concerns about the packet loss, which makes the implementation unsuitable for the safety system which required a defined Safety Integrated Level (SIL), different transport layer protocols were assessed, in particular UDP and TCP. However, even if TCP seems the best candidate thanks to flow control and quality of service mechanisms, it results a bit slower, due to its bigger header and also to the acknowledgment messages, and it lacks multicast and broadcast services, which could be very useful to manage the network. For this reason, a protocol-agnostic caching layer was proposed, bringing support for frame retransmission and duplicate message management in any layer of the protocol stack. The algorithm and the working principle of this layer have been discussed in detail. In addition, the three variants namely TCP, UDP, and UDPc have been compared with each other both through a simulated environment and on a prototype, analyzing the impact on polling time and packet loss.

The Contributions are also reported in Peserico, Fedullo, Morato, Tramarin, and Vitturi (2021b) and Peserico, Morato, Tramarin, and Vitturi (2021c). Always in Chapter 4, the implementation of an OMNeT++ simulation model for functional safety over wireless is introduced. In particular, the model was tuned on the experimental measurements obtained with the previous implemented set-up. The main purpose of the simulator is to be able to analyze safety-critical distributed systems, with multiple nodes under different scenarios and different environmental conditions. Since the previous analysis showed that not always it is possible to guarantee the required SIL level with the FSoE over Wi-Fi implementation, the main goal of this simulator was to reproduce the real network communication, to understand if the obtained time deadlines are in accordance with a possible application and if the loss and/or timeout allow to achieve the required SIL. The simulator has been tested and validated considering an industrial scenario comprising multiple nodes. These contributions are bases on Peserico et al. (2021b) and Morato, Peserico, Fedullo, Tramarin, and Vitturi (2021a)

2 Analyzed communication Networks for Industrial environment

Communication networks are strongly used, at all hierarchical levels, in modern automation industrial systems. The introduction of industrial networks started in the 1980s, with the so-called “Fieldbuses”, defined by IEC 61158, which were specifically conceived for field-level data exchange between controllers and sensors or actuators. They have been followed by Real-time Ethernet networks, at the beginning of the 2000s and, some years later, by industrial wireless networks. Industrial networks are rather different from traditional communication systems typically used for general purpose applications. Indeed, these types of systems were conceived to satisfy three essential requirements of this environment: real-time, deterministic performances; high reliability; and the ability to withstand extreme conditions which are typical of an industrial environment. For example, a device may be strongly influenced by, humidity, dust, vibrations, magnetic interference, mechanical and thermal stresses. In this scenario, in recent years, industrial networks have become of fundamental importance, not only for the communication between the devices at all automation levels but also for the implementation of intrinsic safety applications that, traditionally, were based on specific and dedicated hardware systems. This opportunity is particularly appealing since also safety applications can be now integrated into the context of factory automation and, more in general, in Industrial

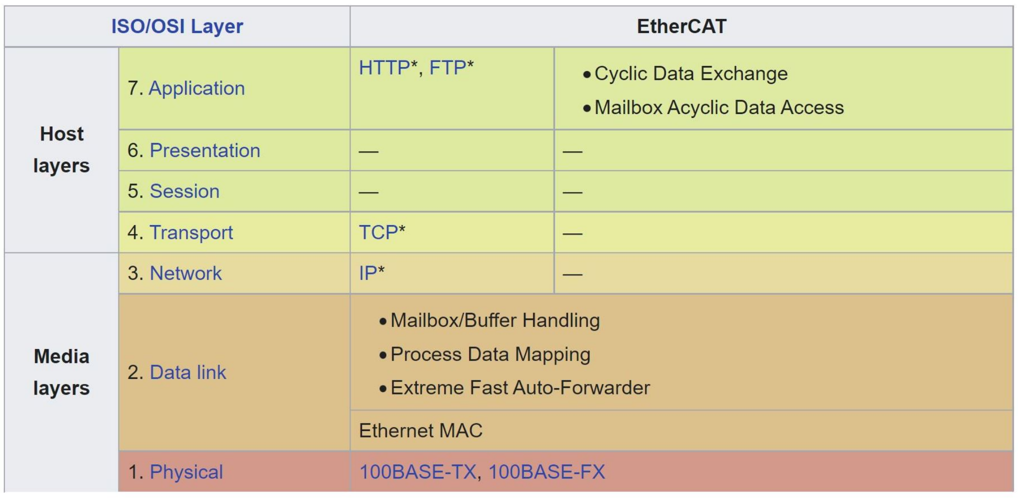
Internet of Things (IIoT) and Industry 4.0 systems. With this context, in this chapter, firstly the common Fieldbuses, EtherCAT, Profibus and Profinet are briefly introduced since they were the starting point of the Ph.D works. Many other Fieldbuses, part of the standard, can be cited, like CIP, Ethernet POWERLINK, CC-Link, but they are not presented in this work. After that, the principal features of IEEE802.11 standard, commonly called Wireless-Fidelity (Wi-Fi), are shown, since this technology was deeply used for all following works. In industrial Environment many other wireless technologies were designed and used. Despite that, Wi-Fi was selected for its large utilization in all fields, not only industrial one thanks to its continuous evolution, which allows to have a constant improvement of performances, its easy availability and possibility of practical implementation and possible modification. Finally regulated ultra wideband wireless protocol IEEE802.15.4z is presented, since this standard is particularly useful for positioning and localization applications, which can be important for the safety environment.

2.1 Fieldbuses

In this part of the chapter, the fieldbuses which were analyzed and implemented on real system, during the Ph.D., are briefly introduced to give the basis for the following works. In particular their communication features are highlighted, but complete descriptions can be found in the relative standard and in work such as the one proposed by Zurawski. The fieldbuses proposed are Ethercat, Profibus and Profinet.

2.1.1 EtherCAT

EtherCAT (Ethernet for Control Automation Technology) is an Ethernet-based Fieldbus system, designed by Beckhoff Automation and based on a master/slave architecture. This protocol lays its foundation on Industrial Ethernet since it uses standard frames and the physical layer as defined in the Ethernet Standard IEEE 802.3. However, it also addresses the specific demands faced in the automation industry. Indeed, to satisfy industrial requirements, EtherCAT uses a high-performing mode of operation, in which a single frame is usually sufficient to send and receive control data to and from all nodes. The EtherCAT master sends a telegram that is received and possibly modified by each node. In particular, each EtherCAT slave device reads data addressed to it and inserts its data in the PDU while it is moving downstream. In this way, the frame is delayed only by hardware propagation delay times. The last node in a segment or branch detects an open port and sends the message back to the master using Ethernet technology's full duplex feature. The EtherCAT master is the only node able to actively send an EtherCAT frame; all other nodes merely forward frames downstream. In such way it is possible to prevent unpredictable delays and to guarantee real-time performances. The master uses a standard Ethernet Media Access Controller (MAC) without an additional communication processor. As said, EtherCAT uses standard Ethernet first layer, for this reason it embeds its frame in Ethernet one, identifying its protocol through the EtherType field which is set to Ox88A4. Since the EtherCAT protocol is optimized for short cyclic process data, the use of bulky protocol stacks, such as TCP/IP or UDP/IP, can be eliminated or it can optionally be tunneled through a mailbox channel without impacting real-time data transfer. During startup, the master device configures and maps the process data on the slave devices. In this sense, an entire EtherCAT network can be identified in a ISO/OSI (Open Systems Interconnection Reference Model) protocol as proposed in Figure 2.1. This means that EtherCAT provides principally the application layer with some improvement of media-link of standard Ethernet, which is the basis of the first and second one. Instead, as said, the network and transport layer are optionally and can include a TCP/Ip or UDP/IP layers.



ISO/OSI Layer		EtherCAT	
Host layers	7. Application	HTTP*, FTP*	• Cyclic Data Exchange • Mailbox Acyclic Data Access
	6. Presentation	—	—
	5. Session	—	—
	4. Transport	TCP*	—
Media layers	3. Network	IP*	—
	2. Data link	• Mailbox/Buffer Handling • Process Data Mapping • Extreme Fast Auto-Forwarder	
		Ethernet MAC	
	1. Physical	100BASE-TX, 100BASE-FX	

Figure 2.1: EtherCAT in ISO/OSI model

The EtherCAT frame (Figure 2.2) contains the frame header and one or more datagrams and different amounts of data which can be exchanged with each slave, from one bit up to kilobytes of data. The datagram header indicates the type of access to the master device (read, write, or read-write) and the way of addressing it. In particular, the master could access a specific slave device through direct addressing, or access multiple slave devices through logical addressing, which is used for the cyclical exchange of process

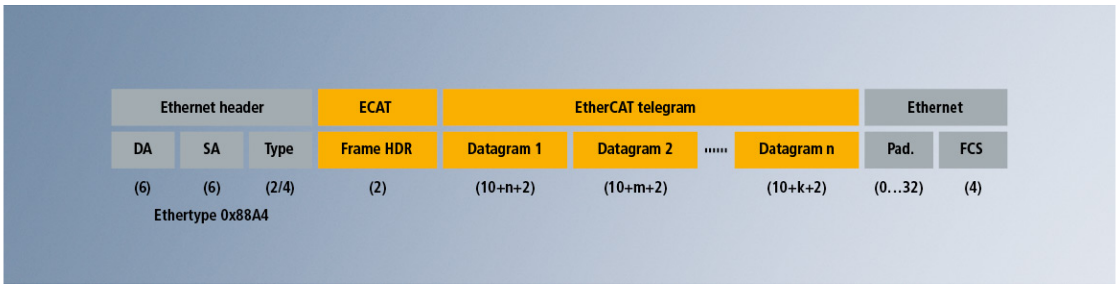


Figure 2.2: EtherCAT frame

data. Each datagram addresses a specific part of the process image in the EtherCAT segment, for which 4 GBytes of address space is available. At startup, one or more addresses in the global address space are assigned to each slave. If slaves share addresses in the same area, they can be addressed with a single datagram. Datagrams completely contain all the data access-related information, allowing the master to decide when and which data to access without using a fixed data structure. The master device can use simultaneously both short cycle times and longer cycle times, accordingly to the application. For example, it can use the smallest one to refresh data on the drives, while it could use a longer one to sample the I/O.
This also relieves the master device in comparison to conventional Fieldbus systems, in which the data from each node had to be read, sorted with the help of the process controller, and only finally copied into memory. Instead with EtherCAT, the master device only needs to fill a single EtherCAT frame with new output data, and send the frame via automatic Direct Memory Access (DMA) to the MAC controller. When a frame with new input data is received via the MAC controller, it can be copied via DMA into the computer's memory – all without the CPU having to actively copy any data. EtherCAT provides further datagrams, which can be used for asynchronous or event-driven communication, and, moreover, it provides additional logical addressing via nodes' positions in the network. After checking the network configuration, the master can assign each node a configured supplementary address and communicate with the node via this fixed address. This enables targeted access to devices, allowing possible network topology reconfiguration. For as concerning slave-to-slave communication, EtherCAT gives two possible methods: downstream communication or communication through the master. In the first case, data can be sent directly to another slave device that is connected further downstream in the network. In particular, since EtherCAT frames can only be processed going forward, this type of direct communication depends on the network's topology, and it is particularly suitable for a constant topology (e.g. in printing or packaging machines). In contrast, the second slave-to-slave communication requires

two bus cycles. EtherCAT offers a lot of flexibility regarding both the topology and the cable type, indeed almost all the topologies (line, tree, star, or daisy-chain) are supported and each segment can use the exact type of cable that best meets its needs. A final essential feature of EtherCAT, which must be cited, is the mechanism of alignment of the distributed clocks, which is a hardware-based calibration of all clocks' nodes. In detail, the time from the first DC slave device is cyclically distributed to all other devices in the system, allowing a precise adjustment of each slave clock to this reference clock with a resulting jitter smaller than 1 µs. Clearly, since transmission of the time information results delayed due to physical constraints, the propagation delay is measured and compensated for each slave device to ensure synchronicity and simultaneousness. In particular, this delay is possibly measured during network startup or, even continuously during operation. Through the alignment of the distributed clocks, all nodes have the same time information, so, they can set their output signals simultaneously and affix their input signals with a highly precise timestamp. Finally, the use of Distributed Clocks also unburdens the master device; indeed, since actions such as position measurement are triggered by the local clock, the master device doesn't have such strict requirements for sending frames, allowing a software implementation of its stack on standard Ethernet hardware. In addition, the system results very robust to important jitter, also in the range of several microseconds. In fact, it does not modify the accuracy of the Distributed Clock, since this accuracy does not depend on time when it is set and so the frame's absolute transmission time becomes irrelevant. In such a way, the EtherCAT master needs only to ensure that the telegram is sent early enough before the DC signal in the slave devices triggers the output.

2.1.2 Profibus

The design of the technology modules with PROFIBUS is oriented toward the OSI layer model (Figure 2.3). Where the communication process between two nodes is distributed over the seven layers, from the physical one to application one.
PROFIBUS defines:

- at physical layer (layer 1), PROFIBUS provides the use of copper-wire versions (RS485 and MBP) and optical and wireless transmission.
- at data-link layer (layer 2), which defines the description of the bus access method, including data security, Profibus provides a master-slave protocol in conjunction with the token method.
- at application layer (layer 7), Profibus DP (or AP) is used.

Despite that, the actual application process lies above layer 7 and is not part of the OSI model.

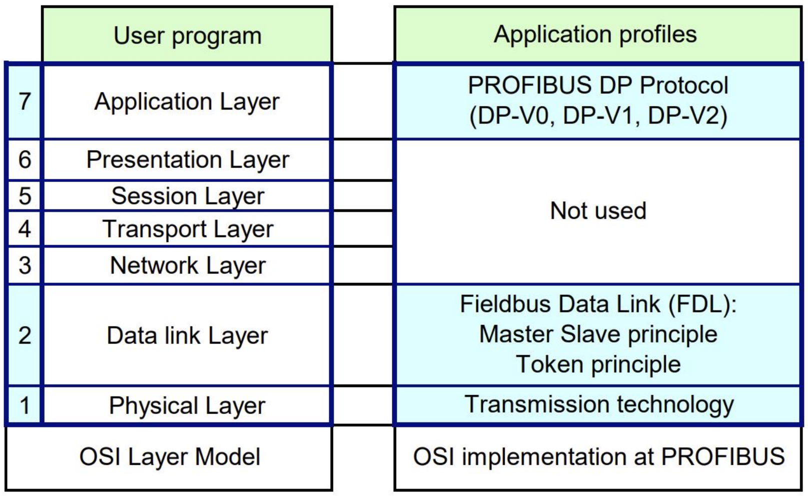


Figure 2.3: Profibus in ISO/OSI model

As proposed, at application layer there are two available variants of Profibus today:

- Profibus DP (Decentralised Peripherals), which is the most common, used to operate sensors and actuators via a centralized controller in production automation applications.
- Profibus PA (Process Automation), which is used to monitor measuring equipment via a process control system in process automation applications with particular attention to those applications in explosive/hazardous areas. Indeed the Physical Layer was designed to be in conformance to IEC 61158-2. This means that the power is delivered over the bus to field instruments, limiting current flows to avoid explosive conditions also in malfunction cases.

As said, Profibus provides a master/slave token bus protocol for deterministic communication between Profibus masters and their remote I/O slaves, which are peripheral passive devices. Indeed they do not have bus access rights, and can only acknowledge, or send response messages to the master upon request. All Profibus slaves have the same priority, and all network communications are originated by the master. Instead Profibus master forms an 'active station' on the network which is subdivided into two different in Class 1 and Class 2. Devices owning to first one, handle the normal communication

and exchange data with the slaves assigned to it. Masters of the second group, instead, are special devices primarily used for commissioning slaves and for diagnostic purposes. The token bus protocol is used to regulate the access to the network by the different masters. This means that all masters form a logical ring in which each device occupies a position that has a fixed address. In such a way every node knows the addresses of the previous and the next device. A special PDU, called token, allows the device holding it to manage its tasks and the network. In particular, it can transmit in the network, it can interview its slaves, or can manage the network, while the other nodes must only listen or respond. This PDU is held by the device only for a fixed time, which corresponds to the token holding time, after this period the device holding it must send it to the next node. This working configuration provides some basic functions useful to the maintenance of the network such as the addition of a new station in the logical ring, erasure of station from the network, initialization of the logical ring, and the management of the loss of the token. The master-to-master communication takes place only with the token exchange. Instead, a master operates using a cyclic transfer to communicate with slaves, with the possibility of addressing an individual slave, a defined group of slaves (multicast), or all connected slaves (broadcast). The length (and timing) of the I/O data to be transferred from a single slave to a master is predefined in the slave's device database or general station descriptor (GSD) file, which contains parametrization and configuration data, an address allocation list, and the bus parameters for all connected stations. A master uses this information to set up communication with each slave during startup. With all these specific, periodic polling methods with predetermined length and timing messages, the master-slave communication results completely deterministic.
PROFIBUS provides four possible services to manage different types of necessities. In particular, it includes the possibility of sending data with or without acknowledgment and a request/response mechanism that can be cyclical or not. Despite that, PROFIBUS DP supports only the services: send data with no acknowledgment (SDN) and send and request data with reply (SRD). In particular, the first one is used for broadcast from a master station to all other stations on the bus. Conversely, the second one is based on a real dual relationship between the initiator and the responder. Most data transfers are defined as SRD, where a response can occur as a one-byte acknowledgment or a data packet and where a message cycle of an addressed station has to respond within a bounded time interval. Clearly, in accordance with the different types of messages, the structure of the frame depends on its function (Figure 2.4). In particular, the telegram can be: with no data, used to coordinate the network and send function information; with variable length data, to send messages of variable length; with fixed length data, to

send messages of fixed length; the Token, to communicate between masters; the short acknowledgment for the service providing it.

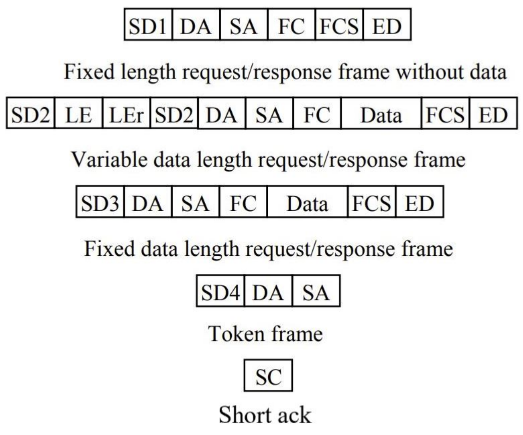

SD1 DA SA FC FCS ED
Fixed length request/response frame without data
SD2 LE LEr SD2 DA SA FC Data FCS ED
Variable data length request/response frame
SD3 DA SA FC Data FCS ED
Fixed data length request/response frame
SD4 DA SA
Token frame
SC
Short ack

Figure 2.4: Profibus frames

Where the acronym are described in Table 2.1, Table 2.2, in particular in Table 2.2 are presented the Service Access Points (SAP)), which are internal addresses used at any level, allowing to access to different services.

SD	Start delimiter
LE	Length of protocol data unit
LEr	Repetition of length of protocol data unit (with an Hamming distance equal to 4)
FC	Function Code
DA	Destination Address
SA	Source Address
DSAP	Destination Service Access Point
SSAP	Source Service Access Point
PDU	Protocol Data Unit
FCS	Frame Checking Sequence, calculated by adding up the bytes within the specified length
ED	End Delimiter

Table 2.1: acronym definition

Finally, concerning the physical level, as initially proposed, Profibus provides two different carrier band transmission, a broadband transmissions, the possibility of using the optical fiber, and, also a wireless medium. The carrier band transmissions are called phase-continuous and phase-coherent. The first one provides a velocity of 1 Mbit/s and

Default 0	Cyclical Data Exchange(WriteReaData)
54	Master-to-Master SAP(M-M Communication)
55	Change Station Address(SetSlaveAdd)
56	Read Inputs(RdInp)
57	Read Outputs(RdOutp)
58	Control Commands to a DP Slave(GlobalControl)
59	Read Configuration Data(GetCfg)
60	Read Diagnostic Data(SlveDiagnosis)
61	Send Parametrization Data(SetPrm)
62	CheckConfiguration Data (ChkPrm)

Table 2.2: SAP definition

a transmission in which the logical level 0 and 1 are identified by different frequencies, 3,75 Mhz and 6,25 Mhz respectively. The transition between the 2 frequencies is gradual and for this reason the name continuous. The Phase-coherent modality uses the same methods but links the transmission velocity to the frequencies. For example for a velocity of 5 Mbit/s the frequencies used are 5 Mhz and 10 Mhz. Both carrier band transmissions use a coaxial cable. The broadband method, instead, provides fixed velocity (1, 5, and 10 Mbit/s) and relatively fixed frequencies (1.5, 6, and 12 Mhz respectively). The Optical fiber, instead, is the fastest method, but it needs the use of electro-optical converts. For wireless implementation, PROFIBUS provides guidelines that specify the integration of WirelessHART (used in process automation) and Wireless Sensor/Actuator Network (WSAN, used in factory automation).

2.1.3 Profinet

PROFINET is the open Industrial standard developed and maintained by PROFIBUS & PROFINET International (PI). It is standardized in IEC 61158 and IEC 61784 and, as a universal communication technology, covers all requirements of automation technology. Profinet was born as an evolution of Profibus to allow communication between several Fieldbus using industrial Ethernet. In particular, it is 100% Switched Ethernet according to IEEE 802.3 and is thus also open for application of all Ethernet technologies and parallel operation of multiple Ethernet protocols. The functional scope of PROFINET can be scaled, according to multiple layered Conformance Classes (CC) (Figure 2.5). These combine minimum application-oriented properties: CC-A includes the basic functions and is used, for example, in building automation; CC-B expands the functional scope to include network topology information and diagnostic function; CC-B(PA) adds functions relevant for process automation such as redundancy and optional ”dynamic reconfiguration” both for communication parameters and for topology; CC-C further expands the functions for implementation of Isochronous Real Time (IRT) communication and is thus the basis for clock-synchronized applications. PROFINET recognizes the following device families: PROFINET controller (corresponds to PROFIBUS master class 1), PROFINET device (corresponds to PROFIBUS slave), and PROFINET supervisor (corresponds to PROFIBUS master class 2).

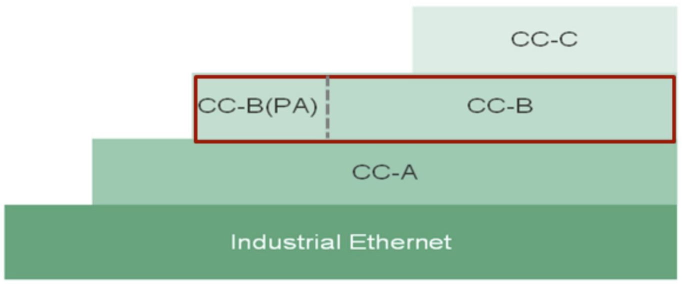


Figure 2.5: CC classes for Profinet

As said for EtherCAT, industrial automation requirements regarding time behavior and isochronous operation cannot be fully satisfied using the TCP/IP channel, which is typical of standard Ethernet communication via TCP(UDP)/IP. For this reason, Profinet introduced a scalable real-time approach, which can be realized with standard network components, such as switches and standard Ethernet controllers. The real-time (RT) communication takes place without TCP/IP information, while it is based on cyclical data exchange using a provider/consumer model. In this type of model, there is not a device that controls the access to the bus and so every node is able to start the transmission if the communication medium is idle(Producer interface). Instead, every node is able to read the bus if someone is transmitting (Consumer interface). Profinet defines real-time classes for data exchange to enable enhanced scaling of communication options and, thus, also of determinism. These classes involve unsynchronized and synchronized communication. In addition, RT frames are automatically prioritized in Profinet compared to UDP/IP one, and in such a way, they can not be delayed by this last type of frame. Profinet differentiates 4 classes for RT transmission, which differ in determinism rather than in performance.

- RT CLASS 1: Unsynchronized RT communication within a subnet, which does not require special addressing information, indeed the destination node is identified using the standard Destination Address. This type of communication is used for CC-A and CC-B classes and relative applications.

- RT CLASS 2: Communication characterized by both synchronized and unsynchronized frames. This type of communication is deprecated and not used today
- RT CLASS 3: Synchronized communication within a subnet. This type of communication is used for motion or high-speed applications and so for CC-C classes.
- RT CLASS UDP: The unsynchronized cross-subnet communication between different subnets, which requires addressing information via the destination network (IP address).

Cyclic I/O data are transmitted unacknowledged as real-time data between provider and consumer in a parametrizable resolution. They are organized into individual I/O elements (sub-slots). The connection is monitored using a watchdog (time monitoring mechanism). During data transmission in the frame, the data of a sub-slot are followed by a provider status. This status information is evaluated by the respective consumer of the I/O data. In particular, the cycle can be subdivided into 4 different phases (Figure 2.6):

- RED phase: In this phase only RT Class 3 can be sent at scheduled instants and using a rigid fixed path.
- ORANGE phase: In this phase only RT Class 2 can be sent in scheduled instants but without a fixed network topology.
- GREEN phase: In this phase only messages that use Ethernet priority defined by IEEE 802.1Q standard can be sent. In particular, RT Class1, RT Class2, and TCP and UDP protocols messages are sent.
- YELLOW phase: Characterized by the same traffic of the GREEN phase but with the additional constraint that only the frame that can be sent completely are effectively transmitted.

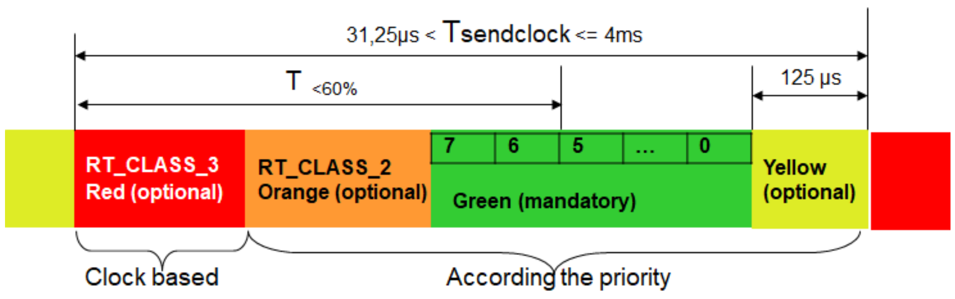


Figure 2.6: Profinet cycle phases

Instead acyclic data exchange can be used to parametrize and configure IO-Devices or to read out status information. This is accomplished with read/write frames via standard IT services using UDP/IP. PROFINET provides also an Isochronous data exchange defined as Isochronous Real-Time (IRT). These data exchange cycles are usually in the range of a few hundred microseconds up to a few milliseconds. The difference with real-time communication is essentially the high degree of determinism so that the start of a network cycle is maintained with high precision. The start of a network cycle can deviate up to 1 µs (jitter). IRT is required, for example, CC-C application in combination with RT class 3 messages. In addition, Profinet defines the Multicast Communication Relation (MCR), which is designed for data exchange with multiple parameters: in detail, it allows direct data traffic from a provider to multiple nodes (up to all nodes) as direct data exchange. MCRs within a segment are exchanged as RT frames. Cross-segment MCR data follow the data exchange of the RT class. All the PROFINET messages are structured as an ethernet frame and in particular, the structure is presented in the following figure (Figure 2.7). Besides at physical level, common ethernet specific are used.

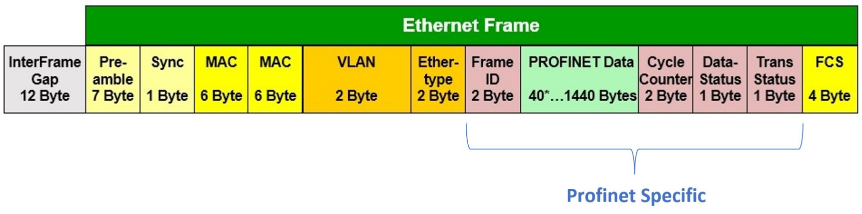


Figure 2.7: Profinet frame

2.2 Wireless protocols in the industrial environment

Wireless communication systems have diffused into an ever-increasing number of application areas and have achieved wide popularity thanks to two essential features: user mobility and cable reduction. Wireless telephony and wireless mobile Internet access are now an important part of our daily lives, and wireless local area network (WLAN) technologies become more and more the primary way to access business and personal data. In factory plants, wireless technology can be used in several interesting ways (Willig (2008); Willig, Matheus, and Wolisz (2005); Gungor and Hancke (2009)). Indeed, WLAN technology can provide communications services for distributed control applications (Hespanha, Naghshtabrizi, and Xu (2007)) involving mobile subsystems like autonomous transport vehicles, and robots. The dynamicity of wireless systems

allows plant reconfiguration to get easier. In addition, the use of wireless technology can reduce risks for those applications of distributed control systems in explosible areas or in the presence of aggressive chemicals, which, otherwise, could destroy cables and stop the communication. However, when adopting WLAN technologies for the factory floor, some problems occur. The first problem is the tension between the hard reliability and timing requirements (hard real-time) pertaining to industrial applications on the one hand and the problem of wireless channels having time-variable and sometimes quite high error rates on the other. A second problem is the desire to integrate wireless and wired stations into one single network (henceforth called a hybrid system or hybrid network). This integration calls for the design of interoperable protocols for the wired and wireless domains. Furthermore, the usage of wireless technology imposes problems not anticipated in the original design of the (wired) Fieldbus protocols: security problems, interference, mobility management, and so on.
Despite that, the literature proposes many possible solutions for these problems, for example, for a mixed network the Internet Protocol (IP) layer is transparent to the underlying ones and it is possible to communicate to a another network device, without even knowing if it is an ethernet device or a Wi-Fi one. Also for the hard reliability and timing requirements, many solutions are proposed, but in particular, the TSN standard which was designed principally for ethernet systems is in development also for the Wi-Fi solution. From the proposed advantages and the increasing number of proposals for deterministic wireless networks, it is clear that these types of solutions are replacing the cable ones. In particular, the IEEE802.11 standard (Wi-Fi) will be the most used wireless technology thank to its continuous evolution, its important high performances, adaptability, and easy modifiability features.
In this context, given the importance of this technology in the industrial environment, it was specifically studied, analyzed, implemented, and modified during the Ph.D.. For this reason, a brief introduction of the main features of the standard is proposed. Despite that, a complete description can be found in the relative standard IEEE802.11 and in work such as the ones proposed by Zurawski and by Gast.

2.2.1 Ieee802.11 (Wi-Fi)

IEEE 802.11 is part of the IEEE 802 set of local area network (LAN) technical standards, and it specifies the set of medium access control (MAC) and physical layer (PHY) protocols for implementing wireless local area network (WLAN) computer communication. The standard and amendments provide the basis for wireless network products using the Wireless-Fidelity(Wi-Fi) brand.

IEEE 802.11 uses various frequencies including, but not limited to, 2.4 GHz, 5 GHz, 6 GHz, and 60 GHz frequency bands. Although IEEE 802.11 specifications list channels that might be used, the radio frequency spectrum availability allowed varies significantly by regulatory domain and so according to the different national laws.
The protocols are typically used in conjunction with IEEE 802.2, which defines logical link control (LLC) as the upper portion of the data link layer of the OSI Model. Moreover it is designed to interwork seamlessly with Ethernet, and are very often used to carry Internet Protocol traffic.

2.2.1.1 Story of IEEE802.11, standards and features

The base version of the standard was released in 1997 and has had subsequent amendments. While each amendment is officially revoked when it is incorporated in the latest version of the standard, the corporate world tends to market the different revisions because they concisely denote the capabilities of their products. As a result, in the marketplace, each revision tends to become its own standard. The original version of the standard is basically obsolete today. It specified very low bit rates of 1 or 2 megabits per second (Mbit/s) and three alternative physical layer technologies which could implement diffuse infrared operating at 1 Mbit/s, frequency-hopping (FH) spread spectrum operating at 1 Mbit/s or 2 Mbit/s, or direct-sequence spread spectrum (DSSS) operating at 1 Mbit/s or 2 Mbit/s.
The DSSS version of legacy 802.11 was rapidly supplemented by the 802.11b amendment in 1999, which increased the bit rate to 11 Mbit/s.
Widespread adoption of 802.11 networks only occurred after the release of 802.11b, in early 2000. One disadvantage of 802.11b devices is the fact they may have interference issues with other products operating in the 2.4 GHz band such as microwave ovens, cordless phones, Bluetooth devices, baby monitors, and some amateur radio equipment. Interference issues and user density problems within the 2.4 GHz band have become major issues as the popularity of Wi-Fi has grown. For this reason, the 802.11a standard was added to the original standard. This standard uses the same core protocol as the original standard and was the first of the 802.11 family to operate in the 5 GHz band. It uses a 52-subcarrier orthogonal frequency-division multiplexing (OFDM) with a maximum raw data rate of 54 Mbit/s, which typically yields a throughput in the mid-20 Mbit/s. Today, many countries around the world are allowing operation in the 5.47 to 5.725 GHz Band. This will add more channels to the overall 5 GHz band, increasing significantly the overall wireless network capacity. 802.11a is not interoperable with 802.11b since they operate on different frequency bands. However, most enterprise-class Access Points

have multi-band capability today.
If the reduction of inference devices was an important advantage, the higher 5 GHz frequency also brings a slight disadvantage as the effective range. 802.11a signals cannot penetrate as far as those for 802.11b because they are absorbed more readily by walls and other solid objects in their path and because the path loss in signal strength is proportional to the square of the signal frequency. On the other hand, OFDM has fundamental propagation advantages in a high multipath environment, such as an indoor office, and the higher frequencies enable the use of smaller antennas with higher RF system gain, which counteracts the disadvantage of a higher band of operation.
By January 2003 the 802.11g standard was released. This standard works in the 2.4 GHz band (like 802.11b), but uses the same OFDM based transmission scheme as 802.11a. It operates at a maximum physical layer bit rate of 54 Mbit/s. 802.11g hardware is fully backwards compatible with 802.11b hardware. In an 802.11g network, however, the presence of a 802.11b device will significantly reduce the speed of the overall 802.11g network. Despite its major acceptance, 802.11g suffers from the same interference as 802.11b in the already crowded 2.4 GHz range. Additionally, the success of the standard has caused usage/density problems related to crowding in urban areas. To prevent interference, there are only three non-overlapping usable channels in the U.S. and other countries with similar regulations (channels 1, 6, 11, with 25 MHz separation), and four in Europe (channels 1, 5, 9, 13, with only 20 MHz separation). Even with such separation, some interference due to side lobes exists, though it is considerably weaker.
The 802.11n standard includes many enhancements to improve WLAN range, reliability, and throughput. It was firstly published in 2006 in a draft version and only in 2009 as an amendment. At the physical (PHY) layer, advanced signal processing and modulation techniques have been added to exploit multiple antennas and wider channels. At the Media Access Control (MAC) layer, protocol extensions make more efficient use of available bandwidth. All together, these High Throughput (HT) enhancements can boost data rates up to 600 Mbps. IEEE 802.11n builds on previous 802.11 standards by adding multiple-input multiple output (MIMO) and 40 MHz channels to the PHY layer, and frame aggregation to the MAC layer. Behind most 802.11n enhancements lies the ability to receive and/or transmit simultaneously through multiple antennas. 802.11n defines many "M x N" antenna configurations, ranging from "1 x 1" to "4 x 4". MIMO uses multiple antennas to coherently resolve more information than possible using a single antenna. One way it provides this is through Spatial Division Multiplexing, which spatially multiplexes multiple independent data streams, transferred simultaneously within one spectral channel of bandwidth. MIMO can significantly increase data throughput as

the number of resolved spatial data streams is increased. Each spatial stream requires a discrete antenna at both the transmitter and the receiver. Another optional 802.11n feature is 40 MHz channels. Prior 802.11 products use channels that are approximately 20 MHz wide. 802.11n products have the option to use 20 or 40 MHz wide channels. Channels operating with a bandwidth of 40 MHz provide twice the PHY data rate available over a single 20 MHz channel. The wider bandwidth can be enabled in either the 2.4 GHz or the 5 GHz mode, but must not interfere with any other 802.11 or non-802.11 (such as Bluetooth) system, using the same frequencies. The standard 802.11ac was developed from 2011 through 2013 and approved in January 2014.This standard provided some important improvement such as Wider RF bandwidth (up to 160 MHz), more MIMO spatial streams (up to 8), Multi-user MIMO, High-density modulation (up to 256-QAM), which allows to achieve performances of multi-station WLAN throughput of at least 1 Gbps and a single link throughput of at least 500 Mbps. The standard 802.11ax was approved on February 2021 and published on May. It is the successor to 802.11ac, marketed as Wi-Fi 6 (2.4 GHz and 5 GHz) and Wi-Fi 6E (6 GHz) by the Wi-Fi Alliance. It is also known as High Efficiency Wi-Fi, for the overall improvements to Wi-Fi 6 clients under dense environments. For an individual client, the maximum improvement in data rate (PHY speed) against the predecessor (802.11ac) is only 39%. Yet, even with this comparatively minor 39% figure, the goal was to provide 4 times the throughput-per-area of 802.11ac (hence High Efficiency). The motivation behind this goal was the deployment of WLAN in dense environments such as corporate offices, shopping malls and dense residential apartments. This is achieved by means of a technique called OFDMA, which is basically multiplexing in the frequency domain (as opposed to spatial multiplexing, as in 802.11ac).

Finally 802.11be Extremely High Throughput (EHT) is the next amendment of the 802.11 IEEE standard,marketed as Wi-Fi 7. Build upon 802.11ax, this amendment defines standardized modifications to both the IEEE Std 802.11 physical layers (PHY) and the Medium Access Control Layer (MAC) that enable at least one mode of operation capable of supporting a maximum throughput of at least 30 Gbps, as measured at the MAC data service access point (SAP), with carrier frequency operation between 1 and 7.250 GHz while ensuring backward compatibility and coexistence with legacy IEEE Std 802.11 compliant devices operating in the 2.4 GHz, 5 GHz, and 6 GHz bands. Development of the 802.11be amendment is ongoing, the initial draft was published on March 2021, and a final version expected by early 2024.

The most important amendments were presented, but many other ones, which had no much success, can be cited (802.11aq,ai,ah,af,ad,y,p). As proposed, the IEEE802.11 is still an continuous evolution, which started in 1997 up today and the future and many improvements were proposed during the years, from modulation, to multiple antennas, increasing bandwidth, decreasing waiting interval, etc...

2.2.1.2 MAC common features

Despite this important progress, the basis of the IEEE802.11 remains the same since 1997. Indeed each of the 802.11 standards should therefore only differ in physical layer (PHY) characteristics in the ISO/OSI model. In particular for all the amendments of this standard, which deal with the two lowest layers of the OSI reference model, the goal is to be backward compatible at the Medium Access Control (MAC) or Data Link layer. The MAC layer remains the same, providing the access to contention-based and contention-free traffic on different kinds of physical layer, defining the type of communication, the topology, error detection and re-transmission mechanisms. The main features of the MAC layer are introduced following.
To connect to a WLAN network, a device has to be equipped with a wireless network interface controller. The combination of microcontroller and network wireless interface is called a station (STA). A group of wireless stations (STA) that form an association is called a Basic Service Set or BSS. There are two types of BSS:

- Ad hoc, which offers direct communication between STAs, but does not include central control.
- Infrastructure, in which STAs are associated with an Access Point (AP), which may also be connected to a network.

All Wi-Fi radios sense the medium for an idle state beforehand to avoid collisions as defined in the so-called Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) protocol. In particular IEEE802.11 standard defines two basic functions for CSMA/CA: the Distributed Coordination Function(DCF) and the Point Coordination Function (PCF), but the second one is completely unused, and so it is out of the goal of this thesis. Even if DCF is basically the only one used, CSMA/CA function can be implemented through two possible different techniques: the basic and the enhanced one. The basic access technique is a two-way handshake characterized by the transmission of a packet immediately followed by an acknowledgment sent by the receiver. In addition, a four-way handshake, often called request-to-send/clear-to-send (RTS/CTS), is specified by the standard. In detail (Figure 2.8), when a frame arrives at the head of the First-In-First-Out (FIFO)-transmission-queue, a station monitors the channel for an idle time of a DCF Interframe Space (DIFS). To detect an idle channel, 802.11 uses a carrier sense function on the physical layer and a virtual carrier sense function on the MAC layer. The physical carrier sense function monitors the channel for any ongoing transmission. The virtual carrier sense function is based on a timer called Network Allocation Vector (NAV). Every station manages its own NAV which represents the time the medium is estimated busy. The NAV is propagated in a header field of every 802.11 frames called the Duration/ID field. All stations can receive this information and update their NAV accordingly. Only when the NAV reaches zero and the physical carrier sense function detects no transmission for the time of a DIFS, the medium is sensed idle. Clearly, to prevent all stations from transmitting at the same time when the medium is sensed idle, a dedicated mechanism is required. Ieee802.11 provides the so-called back-off process (Figure 2.9). In particular, at the beginning of this collision avoidance action, a random back-off interval time is sampled from a uniform distribution between zero and the current Contention Window (CW) size.

$$CW = rand[0 : CW_i], \qquad CW_{min} \leq CW_i \leq CW_{max} \tag{2.1}$$

Each wireless device has to wait its own random interval time before its transmission. Moreover, the back-off procedure employs a slotted and discrete time scale and a station is allowed to transmit only at the beginning of each slot. The slot time duration depends on the underlying physical layer and should be equal to the time needed for every station to detect a transmission on the medium. If a transmission starts and the channel becomes busy during a back-off decrease, the value is frozen and the process pauses. When the channel becomes idle again for the time of a DIFS, the process is resumed from the frozen value. When the counter reaches zero, the station is allowed to start the transmission in the current slot. To increase the performance in the case of an idle medium, there is the possibility to overcome the back-off procedure: when a frame arrives at an empty queue and the medium has been idle longer than a DIFS, the transmission starts immediately. The receiving station waits for the time of a Short Interframe Space (SIFS) to send an acknowledgment back to the transmitter to confirm the successful transmission. This time is needed to ensure the medium is idle again. Since the time of a SIFS is shorter than the time of a DIFS, the transmission of the acknowledgment is protected from other stations' contention.

If the acknowledgment is not received within the so-called ACK timeout, the frame is considered as lost and needs to be re-transmitted with the same procedure, but increasing the Contention Window:

$$CW_{i+1} = 2 * (CW_i + 1) - 1$$

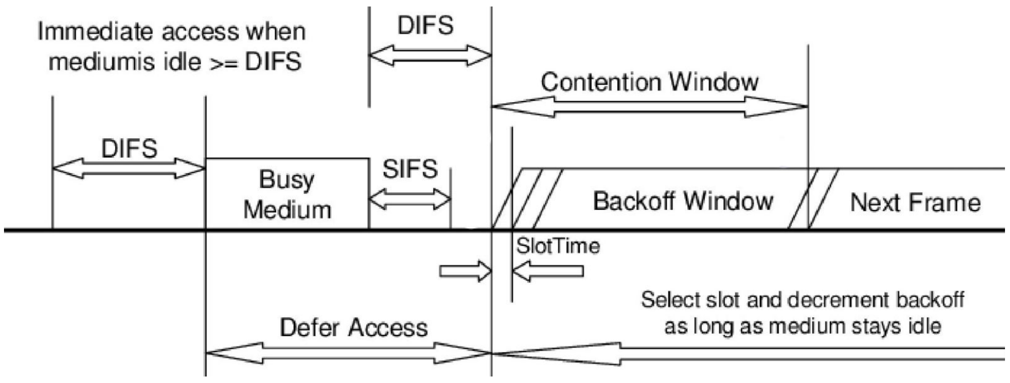


Figure 2.8: IEEE802.11 Distributed Coordination Function basic peration

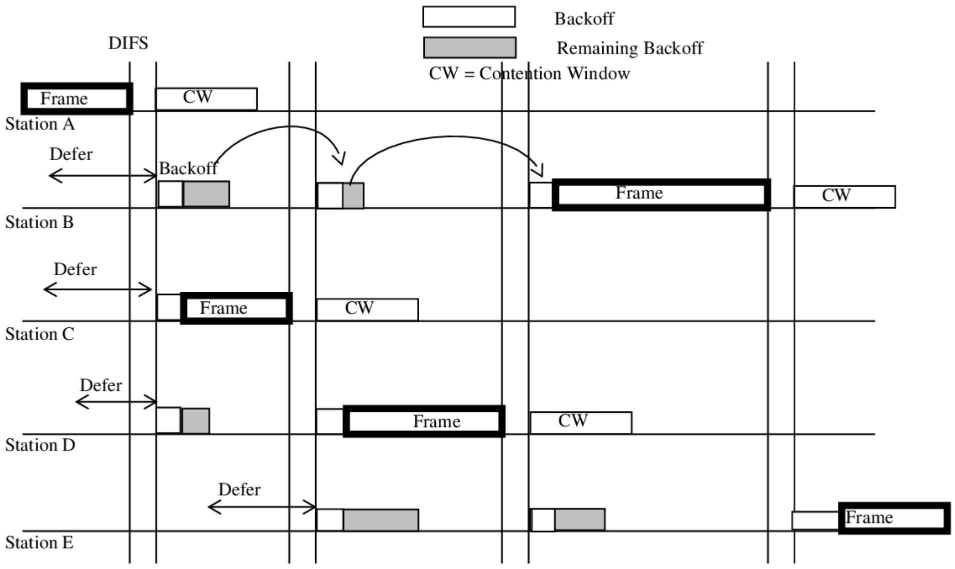


Figure 2.9: IEEE802.11 Distributed Coordination Function contention window operation

This decreases the probability of two stations picking the same back-off slot, but it increases the back-off time.
After a successful transmission, the station sets CW back to CW_{min}, to re-improve the channel utilization and performs a DIFS deference as well as a random back-off, even if there is no packet in the queue. This mechanism ensures there is at least one back-off interval between two transmissions. Instead, if the packet is lost again, the procedure is repeated until the maximum number of re-transmission, and the relative CW_{max} is achieved. Usually, IEEE802.11 provides a maximum of 7 re-transmission, but many devices allow to change this default to improve the real-time performances, accepting the possible loss. Another important InterFrame Space(IFS) besides SIFS and DIFS is called Extended Interframe Space (EIFS). This IFS shall be used when a received frame contains a detected error. Two different error cases can be reported:

1. The PHY detects and reports an error, e.g. carrier lost.
2. The MAC detects and reports an incorrect Frame Check Sequence (FCS).

For stations receiving an erroneous packet, it is in most cases possible to detect this error, however, it is not possible to determine the identity of the receiver. To provide a station (the proper receiver) with the chance to send an acknowledge, these nodes wait for the time of an EIFS. All the presented MAC parameters depend on the underlying PHY. Table 2.3 sums up the parameters for common 802.11 PHYs.

Standard / Parameter	Slot (μs)	SIFS (μs)	DIFS (μs)	CW_{min} (μs)	CW_{max} (μs)
802.11-1997 (FHSS)	50	28	128	15	1023
802.11-1997 (DSSS)	20	10	50	31	1023
802.11b	20	10	50	31	1023
802.11a	9	16	34	15	1023
802.11g	9/20	10	28/50	31	1023
802.11n (2.4 GHz)	9/20	10	28/50	15	1023
802.11n (5 GHz)	9	16	34	15	1023
802.11ac	9	16	34	15	1023

Table 2.3: MAC parameters with different IEEE802.11 standards.

However, all IFS depends on the slot time leading to the possibility of generic calculations which are given below.

$$DIFS = SIFS + 2 * SlotTime$$

$$EIFS = SIFS + DIFS + ACKTxTime$$

In the year 2005, the IEEE published the 802.11e standard a MAC extension for QoS provisioning. This supplement defines a new coordination function called Hybrid Coordination Function (DIFSHCF), which, for the case of Independent Basic Service Set (IBSS) networks, employs the contention-based channel access referred to as Enhanced Distributed Coordination Access (EDCA). This method distinguishes between 8 different User Priorities (UPs) from 0 to 7 to provide differentiated access on a per frame basis. For every MAC Service Data Unit (MSDU) arriving at the MAC layer, higher layers need to provide a UP to take advantage of this differentiation.
Before transmitting QoS-data frames, each station maps the UP to one of the four so-called Access Category (AC):

- Background (AC_{BK})
- Best Effort (AC_{BE})
- Video (AC_{VI})
- Voice (AC_{VO})

Each AC distinguishes itself by individual MAC timings, still utilizing the basic DCF scheme to guarantee interoperability. For every AC, values for AIFS[AC], $CW_{min}[AC]$, and $CW_{max}[AC]$ are assigned which correspond to DIFS, CW_{min}, and CW_{max} in the case of DCF. In general Arbitration Interframe Space (AIFS) is calculated as:

$$AIFS(AC) = SIFS - AIFSN(AC) * SlotTime$$

where AIFSN(AC) is represented as an integer greater one. In the infrastructure mode, an AP distributes the contemplated EDCA parameters to the stations. Instead, in ad-hoc mode, predefined values are used (Table 2.4).

AC	CW_{min}	CW_{max}	AIFS	$TXOP_{max}$
Background (AC_{BK})	15	1023	7	0
Best Effort (AC_{BE})	15	1023	3	0
Video (AC_{VI})	7	15	2	3.008 (ms)
Voice (AC_{VO})	3	7	2	1.504 (ms)

Table 2.4: MAC parameters with different IEEE802.11 standards.

Every traffic class employs a complete DCF entity so all ACs apply for the medium in parallel. After the back-off counter of an AC reaches zero, it applies for transmission on the medium. All other ACs freeze their back-off counters until the medium becomes idle again. Despite that, an additional mechanism, called virtual collision handler, is required for the case when multiple ACs reach zero at the same time. In this case, the station with the higher priority wins, and all other stations increase their Contention Window (CW) size. Is important to highlight an essential difference between EDCA and DCF, which is the decreasing way of the back-off counter. In particular, for the EDCA, the first countdown occurs at the end of the AIFS [AC] interval, while in the DCF case the first decrement occurs at the end of the first slot after the DIFS interval. Moreover, for EDCA either a decrement or a transmission occurs in one slot, whereas the DCF transmits a frame when the counter reaches zero in the same slot.

In Table 2.4 there is a parameter called Transmit opportunity (TXOP). This function was introduced by the 802.11e standard as an additional feature to increase the efficiency of the channel by omitting the back-off and therefore reducing the gap between physical and MAC-layer throughput. This function was still improved by IEEE802.11n. Indeed, besides the several changes on the physical layer, this standard introduces additional and mandatory concepts on the MAC layer, which are a logical extension of the Transmit opportunity (TXOP) to close the gap between the physical and MAC layer by surrendering IFS between data frames. This technique, called "frame aggregation", represents the main 802.11n MAC enhancement. The 802.11n standard distinguishes between two different types of aggregation:

- The aggregate MAC Service Data Unit (MSDU)
- The aggregate MAC Protocol Data Unit (MPDU).

The difference between both methods is the type of aggregated payload. While the A-MSDU logically resides above, the A-MPDU technique operates below the MAC layer.

2.2.1.3 WLAN effective operational process

Finally, the procedure to achieve effective communication in a Wireless Local Area Network (WLAN) with an Access Point (AP) is presented. This mechanism is composed of an initial phase of identification of possible AP, authentication and association. For the first phase, it is important to highlight that when a device is first powered up, the software above the MAC layer stimulates the device to establish contact, but this contact is done through the MAC capabilities. Either active or passive scanning can be used. In particular, the IEEE specification allows different implementations, so characteristics may differ between devices. Passive Scanning uses Beacons and Probe Requests. After selecting a channel, the scanning device listens for Beacons or Probe Requests from other devices. In the case of passive scanning, the client just waits to receive a Beacon Frame from the AP. A Beacon is transmitted from an AP and contains information about the AP along with a timing reference. Devices search for a network by just listening for beacons until it finds a suitable network to join. Instead, with Active Scanning, the device tries to locate an AP by transmitting Probe Request Frames and waits for a Probe Response from the AP. The probe request frame can be a directed or a broadcast probe request. The probe response frame from the AP is similar to the beacon frame. Based on the response from the AP, the client decides to connect to the AP. While active scanning is a faster way to establish contact, it consumes more battery power. Independently by the scanning of each device, the AP periodically broadcasts a Beacon frame (packet) at regular intervals, typically every 100 ms. This is necessary to keep all the clients synchronized with the AP in order for the clients to perform functions as power save. Once identified an AP, the station needs to be authenticated by the AP in order to join the Access point's network. With an open network, a device sends an authentication request and the AP sends the result back. For a secure network, there is a more formal authentication process. 802.1X authentication involves three parties: the access point, device, and the authentication server which is typically a host running software supporting the required protocols. The access point provides the security to protect the network. The device is not allowed access through the AP to the protected side of the network until the device's identity has been validated and authorized. If the authentication server determines the credentials are valid, the supplicant (client device) is allowed to access resources located on the protected side of the network. After authentication, it is necessary the effective association. This procedure enables data transfer between the device and the AP. The device sends an association request frame to the AP which replies to the client with an association response frame either allowing or disallowing the association. Once the association is successful, the AP issues an Association ID to the client and adds the client to its database of connected clients. Finally, the associated device can transfer data with the AP and all other associated devices in the network.

2.2.2 Ieee802.15.4z (Ultra Wideband)

Another wireless network that was studied and implemented during these years, was Ultra Wideband, which, thanks to its features, is particularly suitable for localization and positioning problems. This technology was standardized as IEEE802.15.4z. In particular, according to the Federal Communications Commission (FCC), the UWB radio frequency ranges from 3.1 GHz to 10.6 GHz, with a minimum signal bandwidth of 500 MHz. UWB uses short sequences of very narrow pulses using binary phase-shift keying (BPSK) and/or burst position modulation (BPM) to encode data. The use of narrow pulses results in a transmission exhibiting wide bandwidth, improved range, reduced sensitivity to narrowband interference, and the ability to operate in the presence of multi-path reflections. In particular, the recently released standard, IEEE 802.15.4z, specifically stresses robustness and immunity and ensures a high level of reliability. Moreover, the standard was designed to limit power consumption and to support large numbers of connected devices. UWB can provide performance that may specifically address the needs of real-time capabilities in ranging, positioning, and localization applications as proposed by Liu, Yan, Wang, Ning, and Min (2020). Indeed, this new standard guarantees good accuracy, reliability, timeliness, and low power consumption. Actually, UWB can pinpoint people and things within a few centimeters and presents high immunity to both multipath and interference. Moreover, it is 50 times faster than GPS, with updates up to 1,000 times per second and it is low-powered compared to other mainstream electronic technologies (Yang, Li, and Zhang (2021), Ferrigno, Miele, Milano, Pingerna, Cerro, and Laracca (2021)). UWB localization is based on the Time of Flight (TOF) technique (Zhang, Zhu, Zhao, Liu, and Yang (2020)), which is a method for measuring the distance between two radio transceivers by multiplying the ToF of the signal by the speed of light. From this basic principle, based on the target application's needs, UWB-based localization can be implemented in different ways. In detail, 3 methods are standardized: Two-way ranging (TWR) (Gu and Yang (2018)), Time difference of arrival (TDoA) (Dang, Yang, and Teng (2018)), and Phase difference of arrival (PDoA) (Zhang and Duan (2020)). The first method, TWR, is the simplest one, since it calculates the distance between one device and another one by determining the ToF through the exchange of timestamped messages, and then multiplying the time by the speed of light. As can be seen in Figure 2.10, one device initiates communication with another by sending a poll request. The second device responds to this message including the time to elaborate the response (T_{reply}). When this message is received, the first device calculates the time elapsed between the initial message's transmission and the reception of the subsequent response (T_{round}). The Time of Flight (ToF) is easily calculated as:

$$ToF = \frac{T_{round} - T_{reply}}{2}$$

and finally the distance could be easily calculated multiplying for the speed of light.
The second method requires a more complex network configuration with a shared

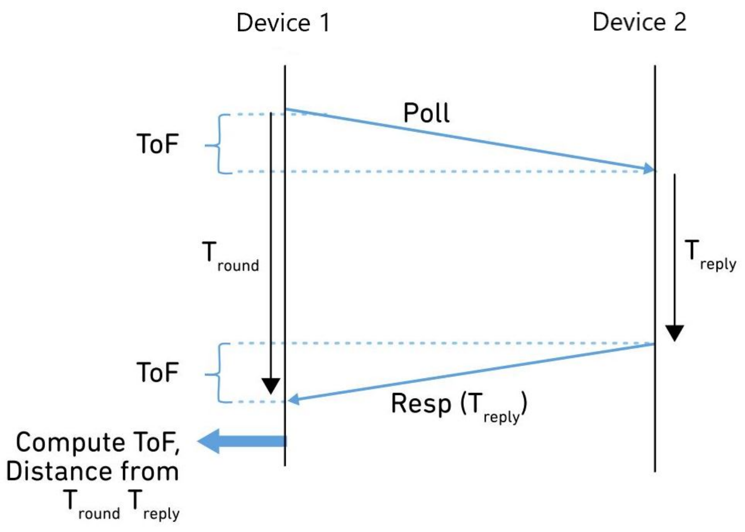


Figure 2.10: Two Way Ranging Scheme

synchronized time information and a relative synchronization algorithm and with fixed, well-known, locations for some specific devices called anchors (Martalo, Perri, Verdano, De Mola, Monica, and Ferrari (2021)). The mobile devices periodically send messages called beacons. When an anchor receives the beacon, it timestamps it, based on the common time. The timestamps from multiple anchors are then forwarded to a central location engine, which will run multilateration algorithms based on TDoA of the beacon signal at each anchor. The result will be either a 2D or 3D location for mobile devices. Finally, the last method, PDoA, combines the distance between two devices with a measure of the bearing (the horizontal angle between the direction of an object and another object) between them. To accomplish this, one of the devices must have at least two antennas and be capable of measuring the phase difference of the arriving signal's carrier at each antenna. This technique is similar to the AOA used by Bluetooth, even though UWB employs it in conjunction with the ToF technique.

3
Improvement of existing communication network

After the analysis of the technologies proposed in the previous chapter, the implementation of real systems allowed to achieve a better consciousness of them, highlighting their features and the possibility of improvement. Moreover, the several difficulties found during the design of embedded devices and the differences between the effective applications and their corresponding model underlined the necessity of having a realistic simulator. In addition, the several difficulties found during the design of embedded devices and the differences between the effective applications and their corresponding model underlined the necessity of having a realistic simulator. In this sense, Objective Modular Network Testbed in C++ (OMNeT++) is a discrete event simulator tool, widely adopted to simulate communication networks and to model the surrounding electromagnetic environment, allowing to properly build protocols exploiting existing modules. However, the models already available in OMNeT++ often implement generic calibrations which can simulate a wide range of scenarios but are unlikely to reproduce specific use cases. Therefore, to be able to carefully simulate a realistic network it is necessary to set up a precise calibration of the most important parameters.
OMNeT++ was used to implement a realistic Profinet digital twin, able not only to correctly represent the real behavior of a real-time Profinet device but also to communicate with a real Profinet network. Moreover, the simulator was used to have a realistic model of IEEE802.11 devices, with the goal of proposing a rate adaptation algorithm outperforming

existing ones. In addition, it was also used to simulate a Wi-Fi sensor network, which was trained with Reinforcement Learning Technique to solve a sensor selection and a pre-processing node selection problem.
At the end, the analysis and studies of UWB were applied to real systems, proposing a simple deterministic algorithm to solve a localization problem for a possible safety-critical targeted scenario.

3.1 Profinet digital twin

Digital Twin (DT), as proposed by Schluse, Priggemeyer, Atorf, and Rossmann (2018), is 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 sets 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 these terms, a Profinet IO RT Class 1 simulator was implemented, based on the OMNeT++. The benefits deriving from the adoption of the proposed DT can be mainly found in three different phases of a plant's lifetime:

- Design. Indeed through the DT is possible to check in advance the effects of the application-specific traffic on the communication network. Moreover with a DT is possible to plan and test the entire network in different scenarios, to design the best one.

- Diagnosis and Maintenance. Indeed the simulations provide future behavior of the system, allowing to predict faults and errors, with the possibility of programming, in advance, maintenance on the corresponding real twin.

The Profinet Host described has been built by extending the OMNeT++ module EtherHost, as shown in Figure 3.1.
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, even if Profinet would be at application one. This module, in particular, implements two essential features of the protocol, namely, the packet encapsulation/extraction and

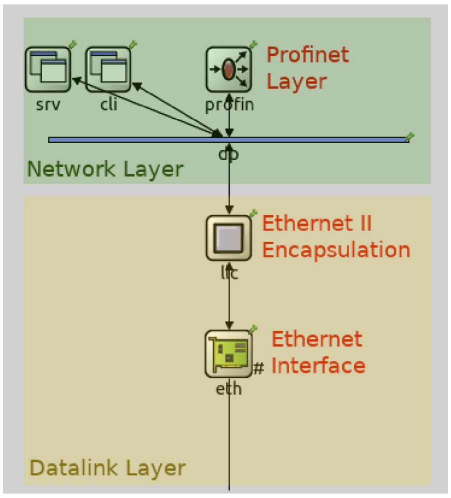


Figure 3.1: Implemented layers structure of the Controller/Device in OMNeT++

the generation of the basic time unit of 31.25 μs, which is used by both the scheduler and cycle counter. In particular the first operation was implemented in accordance with the frame format defined by the Profinet standard, while to the second one was added a random normal term to simulate the possible presence of jitter.
For the validation of the simulation model the prototype network described in Figure 3.2 was taken as reference and implemented in Omnet++. 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 CMZ). 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. Moreover the send clock factor has been set to 32, corresponding to a period of the IO Controller of 1000 μs.

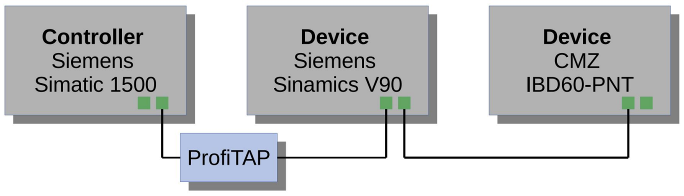
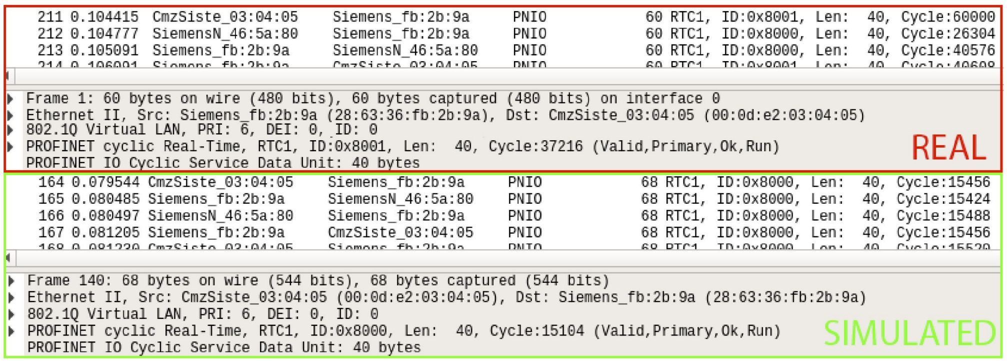


Figure 3.2: Network topology

The Wireshark acquisition plots for both the real and simulated networks (presented in Figure 3.3) show that, from a qualitative point of view, the two acquisitions are practically indistinguishable. Indeed, Wireshark was 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 process data every 1000 µs i.e. every 32 basic clock cycle. This is also confirmed by the statistics reported in Table 3.1, which shows similar results for the sending interval on both real and simulated networks.



```
  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
```

Figure 3.3: Comparison of real and simulated network in Wireshark

	Mean (μs)	Jitter (μs)
Real	1003	304
Simulated	1001	273

Table 3.1: Statistic of the controller sending parameter

Finally, the probability distribution of the sending interval is reported in Figure 3.4 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).

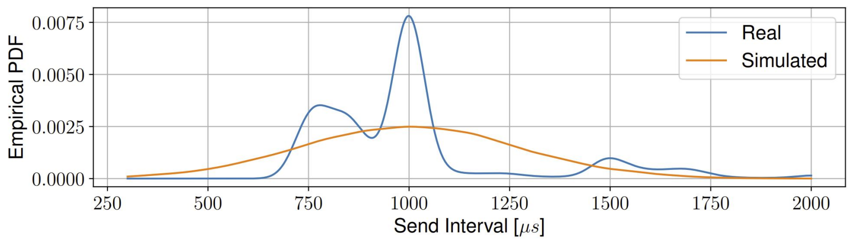


Figure 3.4: EPDF comparison of the send interval in the real and simulated network

3.2 Improvement of rate adaptation for IEEE802.11

During the analysis and study of IEEE802.11, it was clear that the evolution of the standard provides a continuous increasing of the achievable throughput. Despite that, the improvement of the available datarate is consequence of the techniques used for transmitting, which could not fit every situation. For example it is clear that transmitting using a 1024-QAM modulation allows to transmit more information with only one symbol. On the other hand, a simple BPSK allows to have a more robust transmission with the possibility of achieving further distances. In this context it is essential to choose the correct transmission rate according to all the factor that could impact the transmission such as noise, distances, other devices, electromagnetic interference etc. This fact is even more essential for the industrial environment since the necessity could change according to the goal of the transmission, for example for safety messages it is vital to guarantee the integrity of the message, while the amount of data is limited. In this scenario, the IEEE802.11 does not provide a standard rate adaptation algorithm to define the best rate for every situation. Even if the state of art proposes many techniques, there are not many algorithms for the industrial environment. In particular, RSIN is probably the most performing one for the requirements of this domain, but it could be improved.

3.2.1 Reinforcement Learning for Rate Adaptation

In the aforementioned context, the first idea was to propose a preliminary design of a new RA algorithm for Wi-Fi industrial networks, exploiting the features of Reinforcement Learning (RL), whose application in communications and networking are discussed by Luong, Hoang, Gong, Niyato, Wang, Liang, and Kim (2019). Indeed, following the RL concept, the goal is to make an agent learn how to effectively perform RA under some industrial typical constraints. Moreover, the usage of RL within the RA context has been poorly addressed in the literature. This initial work considers the Robust Rate Adaptation Algorithm (RRAA), proposed by Wong, Yang, Lu, and Bharghavan (2006), as a starting point for the introduction of RL techniques. Then, the proposed RA algorithm is preliminarily assessed by means of a simulation campaign that addresses some meaningful performance indicators.

The behavior of RRAA is based on the concepts of Short Term Window (STW) and of loss ratio R_{loss}. Particularly, a specific rate r_i is initially chosen and kept constant for a fixed number of frames, to obtain a STW^i, whose length depends on the rate r_i. At the end of a STW, two predefined thresholds for rate r_i, namely Maximum Tolerable Loss threshold (P^i_{mtl}) and Opportunistic Rate Increase threshold (P^i_{ori}), is compared with the loss ratio, to define the subsequent rate. The loss rate for a specific STW is simply calculated by Equation 3.1.

$$R_{loss} = \frac{\#LostFrames}{\#TransmittedFrames} \tag{3.1}$$

The RRAA algorithm initially starts selecting the highest rate and defining the corresponding STW size. The packet loss is then compared with both P^i_{ori} and P^i_{mtl}. The rate is decreased to the immediately lower rate r_{i-1} if $R_{loss} > P^i_{mtl}$ to decrease the loss probability. Instead, it is increased to the following higher rate r_{i+1} if $R_{loss} < P^i_{ori}$ since it is supposed that in good conditions the throughput can be increased. Finally, if $P^i_{mtl} \leq R_{loss} \leq P^i_{ori}$, the current rate r_i is maintained. Reinforcement learning, that is well addressed by Nian, Liu, and Huang (2020) and by Sutton and Barto (2018), particularly suits RA and moreover RRAA. Indeed RL is based on the definition of *states*, which should be able to effectively describe each possible system condition, *actions*, representing all the possible transitions among states, and *rewards*, that need to be designed accurately to properly evaluate the goodness of an action. Figure 3.5 well describes the RL behavior for a simple RL model: specifically, an *agent* is trained observing the reaction of the *environment*. Within RL, a probability P is associated

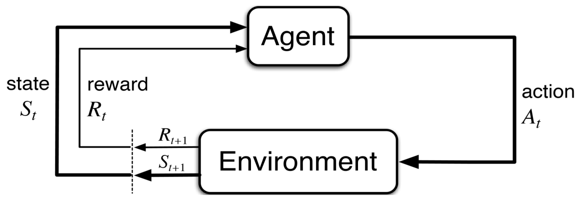


Figure 3.5: RL basic elements.

with the execution of a specific action starting from a specific state. In this way, we can define the tuples $\{S, A, P, R\}$, whose elements are states, actions, probabilities, and rewards, which finally allows to define a Markov Decision Process (MDP). The best policy π, consisting of an ordered sequence of pairs $(state_i,\ bestActionForState_i)$, can be found exploiting several different algorithms. The evaluation of a specific policy π is carried out either with a value function $V^\pi(s) = E_\pi\{R_t|S_t = s\}$, or with an action–value function $Q^\pi(s,a) = E_\pi\{R_t|S_t = s, A_t = a\}$, representing the expected cumulative reward associated with policy π.

Either model-based or model-free techniques can be used to this extent. In the latter case, the evaluation of the value function is made by means of experimental *events*, observed as the result of an agent's action. For example, Monte Carlo or Time Difference methods can be used if, respectively, the evaluation is made once the reward has been produced or through a prediction. The latter ones often are called *bootstrapping algorithms*. Clearly, this breadth of possibilities represents, on the one side, a significant asset of RL while, on the other one, it increases the difficulty to choose and define the best model.

In this context, an MDP was designed drawing inspiration from RRAA, and defined an *action* in such a way that it becomes better as the related packet loss (Equation 3.1) decreases. In addition, the transmission speed was taken into accounts, such as the reward derived from a trade–off between the minimization of the packet loss and the maximization of the rate. Differently from RRAA, this proposal aimed at identifying the "best" rate under any circumstance, removing the constraint of rate changes of only one step, i.e. from r_i to r_{i-1} or from r_i to r_{i+1}, thus allowing to move through all the different rates. The algorithm started from the highest rate and with initial loss rate R_{loss} equal to zero. For simplicity and without loss of generality, in the following the basic IEEE 802.11g amendment was considered. Indeed this choice did not affect the consistency of the design, since the proposed algorithm is general and not specifically tied to the Wi-Fi version at hand. IEEE 802.11g includes 8 different rates, from 6 Mbps up to 54 Mbps. Table 3.2 presents the association between each rate and the specific action, defined in order to allow moving from each state to that specific rate.

Table 3.2: Actions Table

Rate	6Mbps	9Mbps	12Mbps	18Mbps
Action	$A_t = 0$	$A_t = 1$	$A_t = 2$	$A_t = 3$
Rate	24Mbps	36Mbps	48Mbps	54Mbps
Action	$A_t = 4$	$A_t = 5$	$A_t = 6$	$A_t = 7$

For each rate r_i, a set of ten R_{loss} intervals was defined. The states were hence designed to fully characterize the system status: 80 states were identified, in Table 3.3, defined by the transmitting rate r_i and R_{loss}. Furthermore, given the action A_t that allows to select the specific r_i, each state S_t is uniquely identified from R_{loss} and r_i by the following Equation 3.2.

$$S_t = \lfloor (R_{loss} \cdot 100)/10 \rfloor + 10 * A_t \tag{3.2}$$

The assignment of rewards is a critical point of RL. In the proposed algorithm, rewards were calculated by Equation 3.3, where the more R_{loss} is reduced and *Rate* is increased, the higher the reward. The factor β was introduced as a weighting benchmark. In this proposal $\beta = 0.45$ was chosen, with the aim of providing the best trade-off between the minimization of the loss rate and the maximization of the throughput.

Table 3.3: States table

R_{loss} \ r_i	$< 10\%$	$> 10\%$ $< 20\%$	...	$> 80\%$ $< 90\%$	$> 90\%$
6Mbps	$S_t = 0$	$S_t = 1$	...	$S_t = 8$	$S_t = 9$
9Mbps	$S_t = 10$	$S_t = 11$	...	$S_t = 18$	$S_t = 19$
...	...	...	...	...	...
54Mbps	$S_t = 70$	$S_t = 71$	...	$S_t = 78$	$S_t = 79$

$$R_{t+1} = \quad \beta \cdot (-R_{loss_{t+1}} + R_{loss_t}) + (1-\beta) \cdot \frac{Rate_{t+1} - Rate_t}{Rate_{t+1} + Rate_t} - 1; \tag{3.3}$$

Additional special management of the boundary rates was foreseen, choosing the minimum rate when the loss does not decrease and it is bigger than 50%, and rewarding the maximum rate when the loss does not increase and it is smaller than 50%. Finally the State Action Reward State Action (SARSA) algorithm was chosen thanks to its simplicity and to fact that the policy was updated by means of an online procedure. This technique is realized performing the evaluation of the action–value function expressed by Equation 3.4, whose value is then stored in the Q State Table (QST).

$$QST[S_t][A_t] = QST[S_t][A_t] + \alpha * (R + \gamma * QST[S_{t+1}][A_{t+1}] - QST[S_t][A_t]) \tag{3.4}$$

The experiments were simulated in OMNeT++ with two devices moving within the environment at different speeds, continuously varying their relative packet loss. A further attenuation was generated by introducing a simulated obstacle between the elements of the network. The Ieee80211Nist error model, already implemented in OMNeT++, was used to introduce a stochastic error model, allowing to explore every possible state. Finally, the Two Ray Interference model, presented by Sommer and Dressler (2011), was used for the path loss.

The simulations carried out lasted 100 s. In this scenario, a node sent packets to the other with a period of $T = 1$ ms. The obtained results, as throughput performances, are shown in Figure 3.6. Even if both RRAA and the proposed RL-based algorithms are all able to face effectively the channel impairments due to the obstacle, it can be observed that RL algorithms perform quite better in terms of throughput. From an industrial network perspective, a further performance evaluation needed to be carried out considering the transmission delay experienced by the packets in the network. To this aim, Table 3.4 reports the outcomes of this analysis, in terms of both the mean and standard deviation values. In addition, the Cumulative Distribution Function (CDF) of

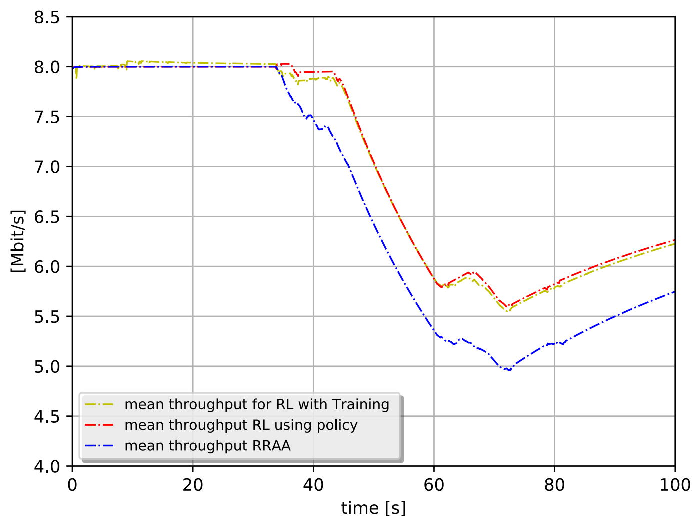


Figure 3.6: Throughput comparison.

the delay is presented in Figure 3.7.

Table 3.4: Simulation results (100.000 packets sent)

Algorithm	**Received packets**	R_{loss} [%]	**Delay [ms] Mean**	**Std. Dev.**
RRAA	71084	28.016	35.255	140.970
RL with training	76191	23.809	22.742	129.773
RL best policy	77401	22.599	16.122	123.917

Both Table 3.4 and Figure 3.7 demonstrate that the RL techniques performed better than the RRAA algorithm. Firstly, RL techniques had a higher number of successful transmissions stemming from the lower packet loss. Secondly, the introduction of RL permitted to decrease the End–to–End delay in terms of mean and standard deviation. This revealed that RL–based RA algorithms were able to converge to a better trade–off between rate maximization and loss minimization. As a final observation, the RL with the training policy, characterized by a randomized rate selection in the 10% of times, performs slightly worse than the RL using the best policy.

3.2.2 Calibration and improvements

The preliminary simulation assessment previously presented showed how RL introduces appreciable improvements over RRAA. Despite that, RRAA is not an industrial-specific

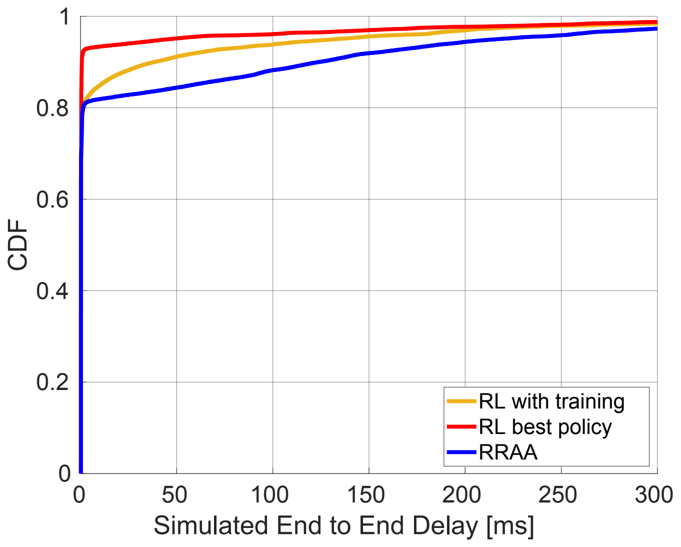


Figure 3.7: End to End Delay CDF.

RA algorithm and the state of art proposes many other RA algorithms. In this context, the idea of applying RL to a technique specifically conceived for industrial real–time, such as RSIN, came immediately as consequence. Moreover, to have more realistic simulations, precise calibration of the used OMNeT++ NIST Error Model (Miller) was done. This crucial calibration phase was focused on the relationship between packet error rate (PER) perceived at the Data-Link Layer, which hence depends on the outcomes of the NIST error model, with respect to the SNR at the receiver. To this aim, referred to the experimental setup proposed by Tramarin, Vitturi, Luvisotto, and Zanella (2016), the PER-SNR relationship has been determined experimentally. Reproducing these measurements data as a reference, the main parameters of the OMNet++ NIST error model were tuned, in a typical calibration procedure. The results are reported in Figure 3.8, where the PER–SNR curves obtained with OMNeT++ are compared with the experimental data.

The RLRA algorithm presented in the previous section had been hence properly modified to take into account the channel behavior by means of the perceived SNR level. This value was used to properly modify States, Actions, and the Rewards given to the agent by the Markov Decision Process (MDP).

Specifically, States S_i were defined considering

- the SNR,
- the chosen Rate,
- the frame loss rate R_{loss}.

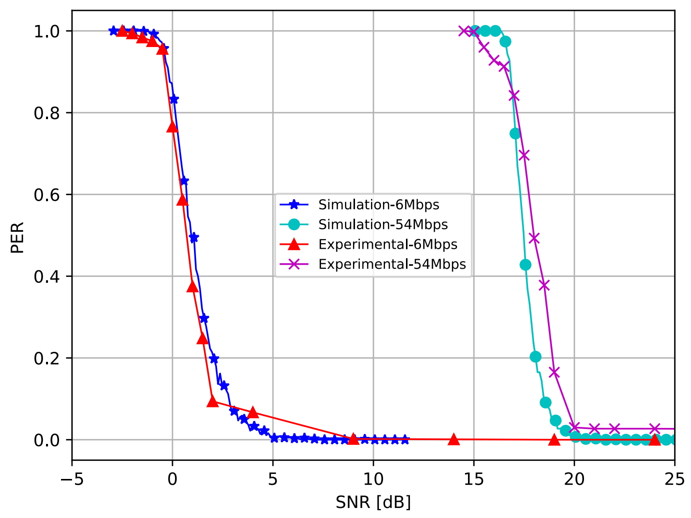


Figure 3.8: Experimental and simulated PER–SNR curves after calibration.

The SNR was divided into 6 different regions SNR_L_i, whose width is 5 dB, while the range of R_{loss} was divided in 10 regions $loss\_L_i$, as described in Figure 3.9. Rates r_i were indexed, for simplicity, using the Modulation and Coding Scheme notation (MCS_i) from 0 to 7, corresponding to 6 Mbps and 54 Mbps, respectively.

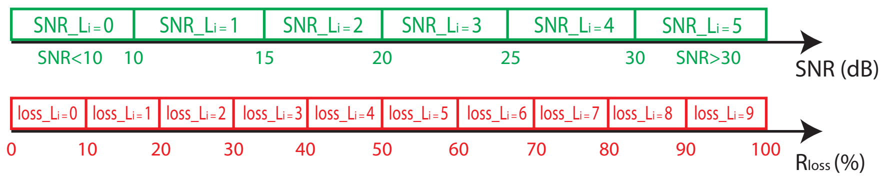


Figure 3.9: SNR and R_{loss} discretization.

Those three terms were then suitably combined, giving rise to 480 different states S_i. Each state had been then univocally indexed by means of Equation 3.5.

$$S_i = loss\_L_i + MCS_i * 10 + SNR\_L_i * 80 \tag{3.5}$$

The rewards function had been adapted as in Equation 3.6 to take into account the SNR level in the RA scheme, in order to provide a better reward to actions which increase the rate when the SNR is high (channel in a good condition), and vice-versa.

$$\begin{aligned} R_{t+1} = \quad & \beta \cdot (-R_{loss_{t+1}} + R_{loss_t}) \\ & + (1-\beta) \cdot \frac{Rate_{t+1} - Rate_t}{Rate_{t+1} + Rate_t} \cdot \frac{SNR_t}{40} - 1; \end{aligned} \tag{3.6}$$

As far as Actions are concerned, two different algorithms had been developed.

- RLRA-SNR: This first and simpler algorithm defined 8 different actions A_t corresponding to the specific rate chosen at the instant t for next step, that is $A_t = 0$ (which corresponds to $r_{t+1} = 6$ Mbps) to $A_t = 7$ ($r_{t+1} = 54$ Mbps). RLRA-SNR, hence, did not differentiate between the first transmission attempt and the possible re-transmissions. Indeed, each packet retransmission is associated with a new chosen Action.

- RLRA-SNR-RC: The second algorithm, conversely, aimed at providing a comprehensive prediction of the whole frame transmission process including also eventual re-transmissions due to bad channel conditions. Specifically, RLRA-SNR-RC determined two different rates to be used for the packet transmission, namely r_t and r_{t+1} with $r_{t+1} \leq r_t$, and performed n_t and n_{t+1} re-transmission attempts with the first and the second rate, respectively. The algorithm further defined a maximum number of transmission attempts n_{max}, performing the remaining $n_{max} - n_t - n_{t+1}$ re-transmissions at the minimum available rate. All the possible Re-transmission Chains (RC) resulting from the aforementioned parameters can be computed in advance to avoid any computation load at runtime.

The simulations were set up with two devices exchanging a total of 100.000 packets. Each packet carried a payload of 50 Bytes, a typical length for time-critical real-time measurement systems. The two nodes moved within the environment: in the first period of time, they were in a plain line of sight, with a good channel status, whereas in the second part, an obstacle started hindering the line of sight path, increasing the path loss and worsening the SNR. The performance index considered in this study was the end-to-end delay, a typical indicator for time-critical or real-time networks. The assessment of the two proposed algorithms was carried out as a comparison of their performance with those obtained by other known RA strategies. In particular, in this analysis the following algorithms were considered:

1. RSIN;

2. RL–based RA algorithm (RLRA);

3. RL–based RA algorithm with SNR (RLRA-SNR);
4. RL–based RA algorithm with SNR and Retransmission Chains (RLRA-SNR-RC).

The outcomes obtained from the first set of simulations have been reported in terms of the experimental cumulative distribution function (ECDF) of the end-to-end delay in Figure 3.10. A more in-depth analysis of the outcomes in terms of number of received packets, R_{loss} and end-to-end delay statistics can be found in Table 3.5.

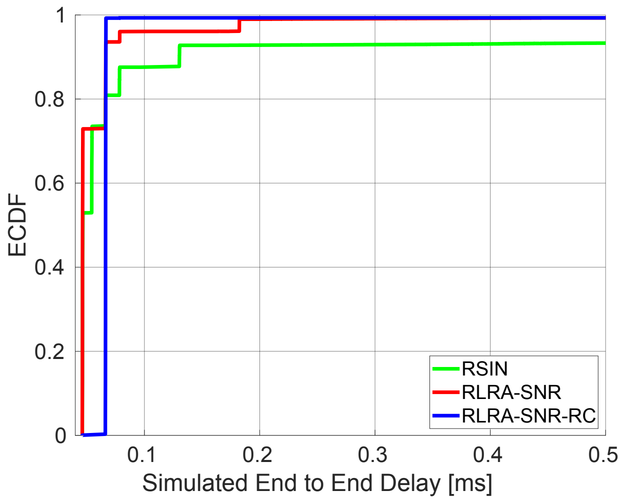


Figure 3.10: Experimental Cumulative Distribution Function for the E2E Delay.

Table 3.5: Simulation results

Algorithm	Received packets	R_{loss} (%)	Delay (ms) Mean	Delay (ms) Std. Dev.
RSIN	82858	17,142	8,407	93,511
RLRA	78835	21,165	15,327	98,162
RLRA–SNR	83494	16,506	4,106	87,272
RLRA–SNR–RC	83426	16,574	4,338	90,944
RLRA–SNR (T)	80868	19,132	7,536	90,528
RLRA–SNR–RC (T)	82472	17,528	8,887	93,340

The analysis of Fig. 3.10 allows to observe that RLRA-SNR-RC had a more conservative behavior and tends to choose lower rates with respect to both RSIN and RLRA-SNR, hence providing slightly higher end-to-end delays. Moreover, this algorithm also presented a steeper curve, indicating that it was able to settle a suitable rate faster than the other algorithms. Table 3.5 provides some more insights. Importantly, all the algorithms adopting a measurement of the SNR were able, as expected, to perform better than

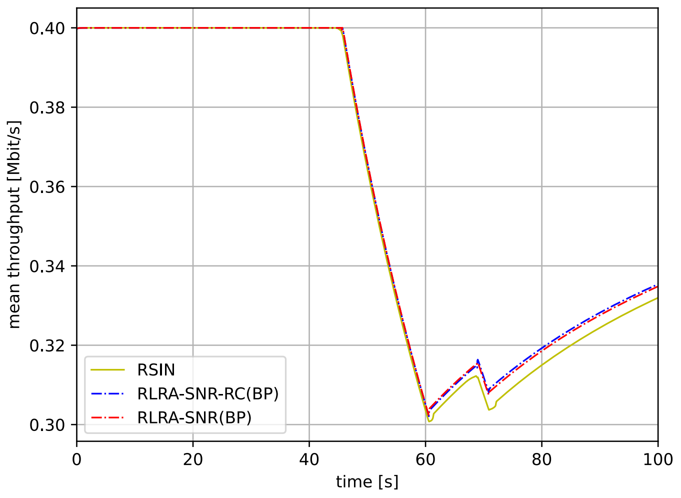


Figure 3.11: Comparison of the Mean Throughput of different RA techniques.

RLRA, which instead learned from the past transmission history. This applied both in terms of average end–to–end delay and standard deviation, and allowed to conclude that the accurate knowledge of the channel status enables more appropriate decisions on the transmission rate yielding to an improved determinism. Another significant result was that both RLRA–SNR and RLRA–SNR–RC performed better than RSIN, indicating their ability to find a better trade-off between the reliability (i.e.minimization of the loss) and the use of high transmission rates. Moreover, RLRA–SNR and RLRA–SNR–RC provided rather similar performance, with RLRA–SNR showing a slightly lower average and standard deviation than RLRA–SNR–RC, mostly thanks to the higher adopted rates. The last two rows of Table 3.5 report the network performance during the training phase of both the proposed algorithms. As expected, the performances during this phase were generally worse than those obtained using the Best Policy π. Clearly, this turns out to indicate that the training activity of RL algorithms, necessary to define the final best policy, had been effective.
Another interesting outcome from this simulation study was relevant to the average throughput that the different algorithms allow, which is represented in Fig. 3.11. While both RSIN and the proposed RL-based algorithms were all able to face effectively the channel impairments due to the obstacle, it can be observed that RLRA–SNR and RLRA–SNR–RC performed slightly better in terms of throughput. This means that, given the small payloads used in the considered scenario, the RL-based algorithms experienced a lower number of packet losses.

Finally, this study allowed to draw some further conclusions. On one hand, the computational efforts, at run-time, required by the proposed algorithms are low compared to that of RSIN, since the latter need to periodically run an optimization problem and update the LUT while the RL–based algorithms directly use the Best Policy. On the other hand, the proposed algorithms are intrinsically able to handle payload variations, whereas RSIN, in those cases needs both an updated PER–SNR map for the new payload length and to run the optimization problem again.

3.3 Adaptive Wi-Fi Smart Sensing and sensor-selection in Resource-Constrained Edge Computing

Possible Wi-Fi applications were inspected also through the study of a centralized problem for sensor selection and elaboration selection of wireless nodes. Moreover, the use of Reinforcement Learning techniques and of OMNeT++ was applied.
In particular, firstly, a new model for a *processing network*, which tailors realistic sampling by resource-constrained smart sensors was proposed. The latter can adapt their local computation and exploit the trade-off online to maximize global network performance, by choosing to either transmit raw data or refine them locally through some on-board data processing algorithms. In addition, motivated by the performance-driven selection proposed by Ballotta, Schenato, and Carlone (2020), sensors could also temporarily set a stand-by mode in order to alleviate the computational burden at the base station. Specifically, this can help because data fusion may degrade the monitoring quality if the amount of sensory data transmitted to the base station overwhelms its computational resources. In fact, the latter design feature has the remarkable side effect of saving energy consumption while at the same time improving global performance. Once formulated the model, which is proposed in section A.1, a Reinforcement Learning approach was applied to solve the problem of choosing the optimal sensing policy in order to minimize the uncertainty of estimation. Finally, the proposed approach was validated through two simulations motivated by smart sensing for a self-driving vehicle and an application of the Internet of Drones, showing that accounting for processing delays can improve performance through a careful allocation of computational resources. In particular, as said, to address wireless communication in a realistic way, OMNeT++ was used.

3.3.1 Reinforcement Learning Framework

By assuming complete knowledge of delays and measurement noise covariances affecting sensors in the different modes, both Problem 1 and 2 become analytically tractable. However, the computation of the exact minimizer requires to keep track of all starts and stops of data transmissions for each sensor, resulting in a cumbersome procedure which

admits no closed-form expression, and requires to solve a combinatorial problem that scales poorly with the number of sensors.
Moreover, in real-life scenarios, the assumptions considered in the formulation of the problem may be too conservative, and the latter method does not allow for relaxation. Indeed, as long as either delays or covariances are not explicitly known or have some variability, *i.e.,* they can be modeled by proper random variables, the minimization becomes intractable. This is true even if the expectations of these random variables are known, since the dynamics in Problem 2 lead to a non-linear behavior for the quantity to be minimized.
Hence, the problem of choosing the optimal sensing policy in order to minimize the uncertainty of estimation is tackled through a Reinforcement Learning algorithm, which allows to easily and successfully address the general formulation of the problem.

3.3.1.1 General Scenario

The Reinforcement Learning (RL) framework (Sutton and Barto (2018)) consists of an agent interacting with an environment without having any prior knowledge of how the latter works and the impact that actions have on it. By collecting samples of the reward signal, which can be thought as a measure of the effect of the interaction, the agent will learn to maximize this user-defined quantity, under the dynamics of the unknown environment. This can be formalized through the Markov Decision Process framework, in which the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$ characterize all needed elements, *i.e.,* the state space $\mathcal{S}$; the set of all possible actions $\mathcal{A}$; the transition probabilities $\mathcal{P}$ regulating the underlying system, which may also be deterministic; the reward function r, which is assumed to be a function of the state and the action; and the discount factor γ, weighting future rewards. The agent selects actions to be performed on the environment through a *control policy*, which is here assumed to be a deterministic function π mapping states to actions, $\pi : \mathcal{S} \to \mathcal{A}$. The goal of the learning procedure would be to identify an optimal policy $\pi^*(\cdot)$, *i.e.,* a map achieving the highest possible return G over the long run, according to the unknown transitions imposed by the environment. The return is simply defined as the expectation of the sum of rewards in an episode of K steps, *i.e.,* $G = \mathbb{E}_\pi \left[\sum_{k=1}^{K} r_k \right]$.
The general problem can then be written as

$$\max_{\pi \in \Pi} \quad \mathbb{E}_\pi \left[\sum_{k=0}^{K-1} \gamma^k r\left(s_k, a_k\right) \right] \tag{3.7a}$$

$$\text{s.t.} \quad s_{k+1}^{\pi} = g\left(s_k, a_k\right) = g\left(s_k, \pi\left(s_k\right)\right) \doteq g_\pi\left(s_k\right), \tag{3.7b}$$

for a generic state-transition function $g : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ representing environment dynamics that embeds the transition probabilities $\mathcal{P}$. One of the simplest and most popular approaches to solve the maximization in (3.7) is the Q-learning algorithm (Watkins and Dayan (1992)).

Q-learning is an iterative *model-free* algorithm which aims at finding an optimal policy to maximize the expected value of the return, without learning an explicit model of the environment.

This is done by focusing solely on the *values* of actions at different states, which correspond to the return. In the tabular setting, this method actually builds a lookup table with the action-value for each state-action pair, and updates it with an *optimistic* variant of the temporal-difference error (Sutton (1988)) at every step, weighted by a learning rate α. The action-value function is defined as:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{K-1} \gamma^k r(s_k, a_k) \middle| s_k = s, a_k = a\right], \tag{3.8}$$

and it expresses the expected return obtained by performing action a at state s and then following policy π afterwards. By defining the optimistic variant of the temporal-difference error as

$$\zeta_k^{\pi}(s,a) = r(s,a) + \gamma \max_{a'} \left[Q_k^{\pi}(s', a')\right] - Q_k^{\pi}(s,a), \tag{3.9}$$

the update for the Q-learning algorithm is given by

$$Q_{k+1}^{\pi}(s,a) = \begin{cases} Q_k^{\pi}(s,a) + \alpha\, \zeta_k^{\pi}(s,a) & \text{if } k < K-1, \\ (1-\alpha)\, Q_k^{\pi}(s,a) + \alpha\, r(s_k, a_k) & \text{otherwise.} \end{cases} \tag{3.10}$$

The Q-function is updated at each step $k = 1, 2, \ldots$ of one episode. The maximization in (3.9) characterizes the algorithm as an off-policy method since the learned policy is different from the one actually employed in the environment by Sutton and Barto (2018). The behavioral policy which runs the episodes and collects the reward is often chosen as the ϵ-greedy policy according to the current lookup table, to emphasize the explorative behavior of the algorithm.

In a finite MDP, this approach is known to converge to the *optimal* action-value function under the standard Monro-Robbins conditions (Jaakkola, Jordan, and Singh (1993)).

3.3.1.2 Optimizing Latency-Accuracy Trade-off

With regard to Problem 1, policy π_i is composed of categorical variables corresponding to sensing modes, and characterizes the potential for intervention in the operations of the ith sensor. The constraints due to the centralized implementation in Problem 2 allow to consider a simplified policy $\pi_{\mathrm{net}} : \mathcal{S} \to \mathcal{A}$ describing how many sensors are required to process or sleep. Specifically, each action $a \in \mathcal{A}$ is specified by pairs of integers (cf. Theorem A.1.2) dictating the amount of sensors which send data and process their measurements within each group $\mathcal{V}_m$ after a is applied. For example, if $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$ with $|\mathcal{V}_1| = 4$ and $|\mathcal{V}_2| = 5$, action $a = \{(2,1),(3,0)\}$ means that two sensors transmit (the other two sleep), with one out of these in processing mode, within $\mathcal{V}_1$, and similarly for $\mathcal{V}_2$.

Since we aim to minimize the time-averaged error variance (A.9a), a straightforward metric to be chosen as reward function is the negative trace of matrix $P_k^{\pi_{\mathrm{net}}}$, which evolves according to the Kalman predictor with delayed updates. In the considered framework the base station is allowed to change sensing configuration (corresponding to a new action) at each time $k^{(\ell)}$, therefore a natural way of defining the reward is to take the average of the negative trace of the covariance during the interval between times $k^{(\ell)}$ and $k^{(\ell+1)}$, so that the base station can appreciate the performance of a particular sensing configuration in that interval.

This leads to the following instantiation of the maximization in (3.7),

$$\max_{\pi_{\mathrm{net}} \in \Pi_{\mathrm{net}}} \quad -\mathbb{E}\left[\sum_{\ell=1}^{L} \frac{\gamma^{\ell}}{k^{(\ell+1)} - k^{(\ell)}} \sum_{k=k^{(\ell)}}^{k^{(\ell+1)}} \mathrm{Tr}\left(P_k^{\pi_{\mathrm{net}}}\right)\right] \tag{3.11a}$$

$$\text{s.t.} \quad P_k^{\pi_{\mathrm{net}}} = f_{\mathrm{Kalman}}\left(\mathcal{Y}_k^{\pi}\right), \tag{3.11b}$$

$$P_{k_0}^{\pi_{\mathrm{net}}} = P_0, \tag{3.11c}$$

with $k^{(L+1)} \doteq K$. The quantity of interest is the trace of the error covariance and thus a straightforward approach would suggest to take $\mathcal{S} = \mathbb{R}^+$, however, to keep the Q-learning in a tabular setting the state space has been discretized through the function $\mathfrak{d} : \mathbb{R}^+ \to \mathbb{N}^+$. In particular, the image of $\mathfrak{d}\left[\cdot\right]$ is given by M bins, which were manually tuned in our numerical experiments to yield a fair representation of the values of $P_k^{\pi_{\mathrm{net}}}$ observed along episodes. Then, based on the bin associated with $\mathrm{Tr}\left(P_{k^{(\ell)}}\right)$, the agent outputs a sensing configuration $a \in \mathcal{A}$ through $\pi_{hom}\left(\cdot\right)$ at each time $k^{(l)}$, given by $a_\ell = \pi_{hom}\left(\mathfrak{d}\left[\mathrm{Tr}\left(P_{k^{(l)}}\right)\right]\right)$.

Also, choosing $\gamma = 1$ and time intervals $[k^{(\ell)}, k^{(\ell+1)}]$ of equal length allows (3.11) to match objective cost (A.9a) exactly.

3.3.2 Numerical Simulations

We validate the performance of our proposed approach against baseline design choices with two numerical experiments.
In 3.3.2.1, we consider smart sensors monitoring an autonomous vehicle to get insight into the allocation of processing. In 3.3.2.2, we address drones for target tracking and see how online sensor selection helps improve performance. Finally, in 3.3.2.3 we discuss the role of Reinforcement Learning in conjunction with Kalman predictor, arguing in favor of a hybrid algorithmic-learning estimation framework.

3.3.2.1 Smart sensing for self-driving vehicle

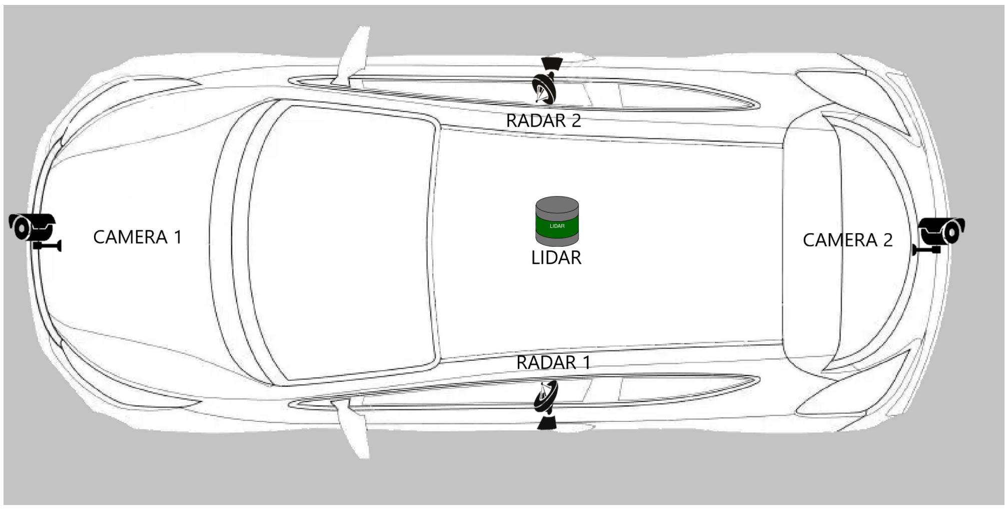


Figure 3.12: Simulation setup for autonomous-driving scenario: sensors measure the car position, and a centralized microcontroller tracks its trajectory.

For the first scenario, we addressed a self-driving car traveling at an approximately constant speed. Specifically, we considered its transversal position with respect to the center of the lane, measured by sensors that transmit to a microcontroller (base station) which orchestrates communication and tracks the car trajectory (Figure 3.12). The car position was modeled with (A.1) as a double integrator, which is a flexible choice used for uncertain dynamic with direct control of accelerations (Tzoumas, Carlone, Pappas, and Jadbabaie (2021); Barreiro, Baños, and Delgado (2021); Oral, Mallada, and Gayme (2021); Rao and Bernstein (2001); Ren (2007)). In this example, it could mimic a lane shift at a sustained speed (*e.g.,* for passing or on a highway). Given such a model, Kalman predictor arguably is an effective and robust estimator, assuming that lateral movements are limited compared to the car speed.

We simulated two radar devices, two cameras, and one lidar, which are commonly employed in self-driving applications (Feng, Haase-Schütz, Rosenbaum, Hertlein, Gläser, Timm, Wiesbeck, and Dietmayer (2021)). Many techniques used in autonomous driving exploit lidar point clouds, such as segmentation, detection, and classification tasks (Li, Ma, Zhong, Liu, Chapman, Cao, and Li (2021)). Also, radars are emerging as a key technology for such systems. Some of today's self-driving cars, *e.g.,* Zoox, are equipped with more than 10 radars providing 360° surrounding sensing capability under any weather conditions (Sun and Zhang (2021)). Finally, camera images are essential to enable the commercialization of self-driving cars with autonomy at level 3 (Kim and Lee (2022)). The sensor parameters (Table 3.6, with $V_{raw} = v_{raw}I$ and $V_{proc} = v_{proc}I$) were chosen based on real-world experiments (Sun, Petropulu, and Poor (2020)), with sampling period $T = 1$ms to ensure real-time vehicle control.

Table 3.6: Sensor parameters for autonomous driving scenario.

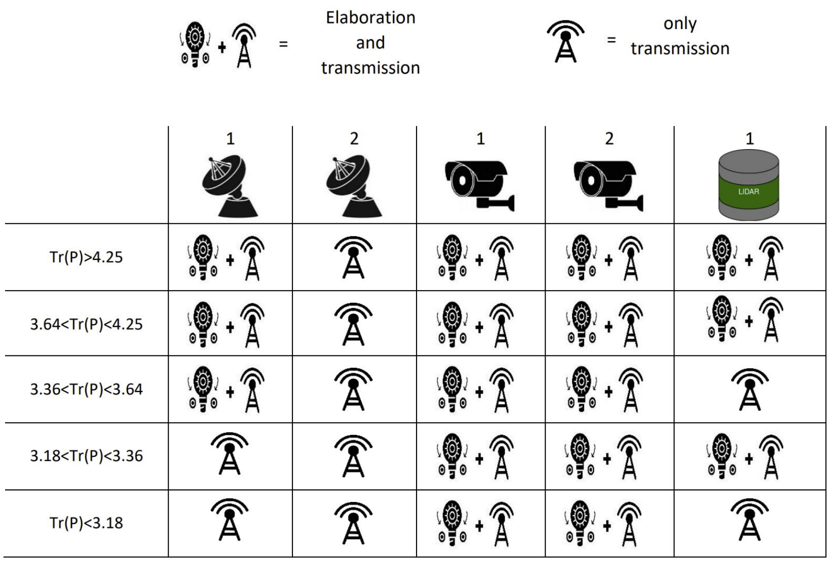
	Freq.	τ_{raw}	τ_{proc}	v_{raw}	v_{proc}
Radar	50Hz	20ms	30ms	0.45	0.40
Camera	25Hz	40ms	100ms	0.3	0.05
Lidar	10Hz	100ms	110ms	0.09	0.054

Table 3.7: Learning hyperparameters for autonomous-driving scenario.

M	α	ϵ_0	$\epsilon_{\min}$	ϵ_t	γ
5	0.1	0.2	0.01	$\max\left\{\frac{\epsilon_0}{\sqrt{t+1}}, \epsilon_{\min}\right\}$	1

The communication protocol was simulated through the discrete-event simulator Objective Modular Network Testbed in C++ (OMNeT++ (2020)).
OMNeT++ is widely adopted to simulate wireless networks, because it already includes common communication protocols and also allows to create customized ones exploiting existing modules. Further, it enables realistic simulations by accurately modeling both the electromagnetic environment and the bottom portion of the protocol stack (from physical to transportation layers). In our simulation, sensors carried IEEE 802.11 (so-called Wi-Fi) communication boards.
As for the training, we set five time windows, each 300ms long, training the system for 100000 episodes with hyperparameters shown in Table 3.7, where t refers to the tth episode.

Table 3.8: Learned network sensing policy for autonomous-driving scenario.



⚙ + 📡 = Elaboration and transmission

📡 = only transmission

	1	2	1	2	1 LIDAR
Tr(P)>4.25	⚙ + 📡	📡	⚙ + 📡	⚙ + 📡	⚙ + 📡
3.64<Tr(P)<4.25	⚙ + 📡	📡	⚙ + 📡	⚙ + 📡	⚙ + 📡
3.36<Tr(P)<3.64	⚙ + 📡	📡	⚙ + 📡	⚙ + 📡	📡
3.18<Tr(P)<3.36	📡	📡	⚙ + 📡	⚙ + 📡	⚙ + 📡
Tr(P)<3.18	📡	📡	⚙ + 📡	⚙ + 📡	📡

As it is possible to infer from Table 3.8, the learned policy requires almost all sensors to process when the error variance is high (top row), while the need for processing diminishes with the variance, turning to raw mode both lidar and radars at the smallest values (bottom row). Interestingly, processing mode is always chosen for cameras, revealing that refining of image frames overhangs the additional computational delay. Note that in this case, given the small amount of sensors, the fusion delays induced at the base station are negligible and sleep mode is never selected, namely, sensors always transmit. The learned policy was tested against two standard design choices: all sensors send raw measurements (*all-raw*), or all process (*all-processing*). The outcomes over the optimized horizon are plotted in Figure 3.13 and summarized in Table 3.9.

Table 3.9: Mean error variance in autonomous-driving simulation.

Q-learning (proposed)	All-raw	All-processing
3.67	3.89	3.88

As it is possible to appreciate from Figure 3.13, the proposed policy (*Q-learning*) cleverly allocates computational resources according to the current estimate accuracy. During the transient phase (till 600ms), when the error variance is larger, processing mode is selected for lidar, cameras, and one radar according to the first two rows in Table 3.8.

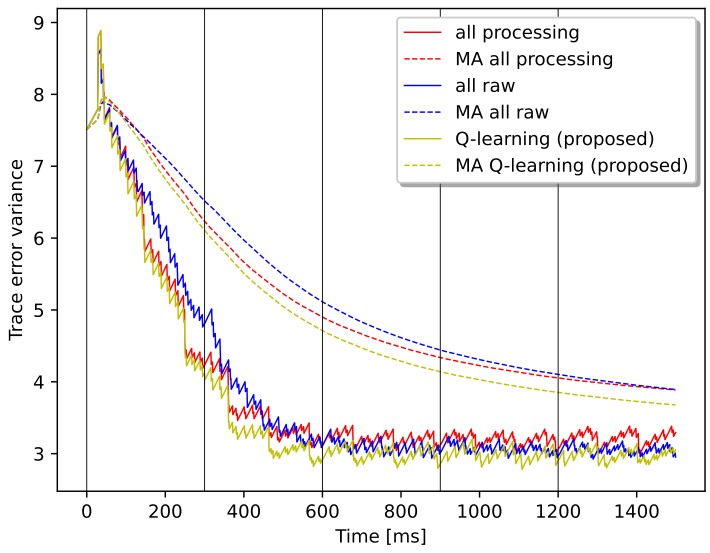


Figure 3.13: Error variance in autonomous-driving simulation.

Indeed, such a choice performs very close to all-processing (red curve), while the all-raw configuration is clearly disadvantageous (higher blue curve). Conversely, at steady state only the cameras are left in processing mode: this resembles more closely the all-raw policy, which performs better (lower blue curve) than all-processing. Overall, we can see from Table 3.9 that the proposed approach leads to a total improvement of about 6% compared to baseline policies. While this result may look marginal, we note that the improvement is rather small over the main transient phase, because the Kalman predictor is able to drop the error variance very quickly for all sensing configurations, but is way larger (about $15 - 20\%$) when the curves settle about small values. Also, while the objective cost (A.9a) refers to the whole horizon, we note that in fact, the learned policy performs better than the baselines nearly at each point in time, as Figure 3.13 shows, with the curve obtained with the Q-learning policy being almost always below the others. Further, the moving average (MA) is consistently smaller than both all-raw and all-processing, highlighting an even better performance of the proposed approach with respect to the targeted optimization.
Finally, we can state that, in this context, the advantage of using the proposed approach is clear: the learned policy exploits knowledge of system dynamics to select the best sensing configurations at different points in time, while baselines cannot adapt to transient and steady-state regimes, which are optimized by different processing configurations.

3.3.2.2 Team of drones for target tracking

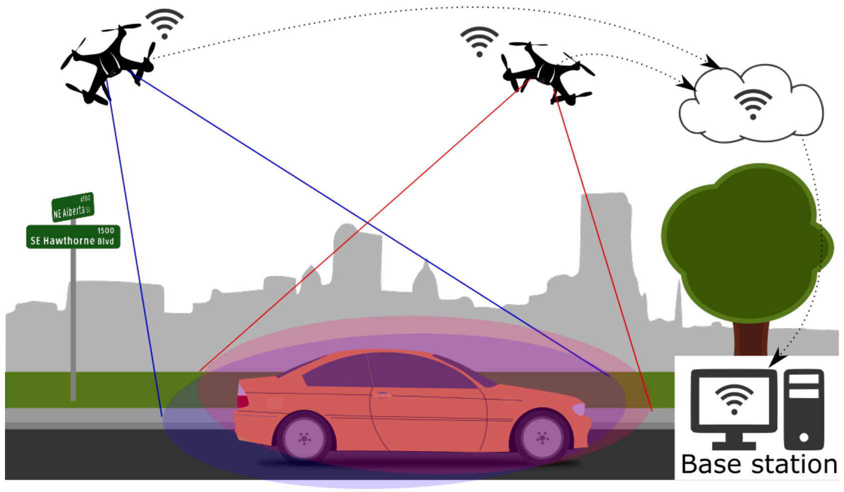


Figure 3.14: Simulation setup for drone-tracking scenario: the base station monitors the target (car) trajectory based on visual updates from drones.

For our second simulation, we considered a team of 25 drones tracking a vehicle on the road (Figure 3.14), modeled as a double integrator akin Ballotta et al. (2020). Each drone is equipped with a smart camera and can either transmit raw images or perform neural object detection on-board and send fairly precise bounding boxes. The used parameters (Table 3.10) are based on experiments proposed by Allan (2019); Hossain and Lee (2019). In this case, given practical limitations of handling many network nodes in OMNeT++, we implemented Python scripts with communication delays $\delta = 10$ms. Also, we set the fusion delay ϕ_k to be proportional to the number of data that are processed by Kalman predictor to compute $\hat{x}_k$. Finally, we addressed an optimization horizon with ten 500ms-long windows and Q-learning hyperparameters reported in Table 3.11.

Table 3.10: Sensor parameters for drone-tracking scenario.

T	τ_{raw}	τ_{proc}	v_{raw}	v_{proc}
10ms	40ms	140ms	10	1

Table 3.11: Learning hyperparameters for drone-tracking scenario.

M	α	ϵ_0	$\epsilon_{\min}$	ϵ_t	γ
5	0.002	0.9	0.1	$\max\left\{\frac{\epsilon_0}{\sqrt{t+1}}, \epsilon_{\min}\right\}$	1

The comparison between our proposed approach and baselines is shown in Figure 3.15 and Table 3.12. In this case, performing an online selection through the sleep mode improves overall performance: in detail, 10 drones are active in raw mode during the transient (first window) and 20 through the rest of the horizon. This means that, with this setup, both data processing is not convenient because of drone resource constraints that cause overlong computational delays, and requiring all drones to send data deteriorates the performance through resource constraints at the base station that hinder information abundance.

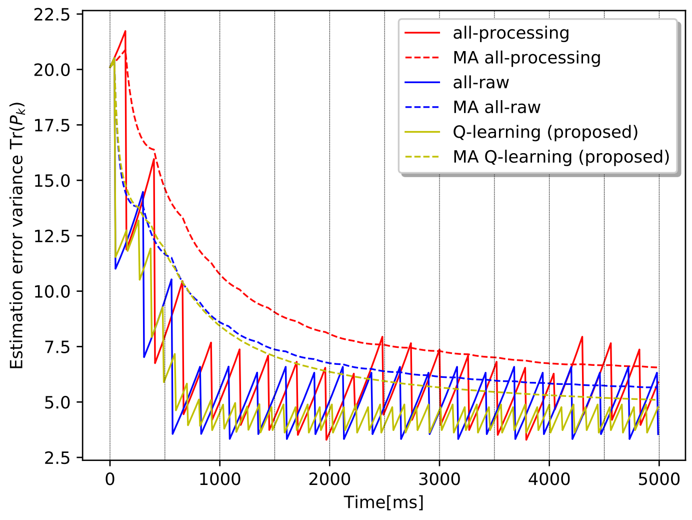


Figure 3.15: Error variance in drone-based tracking simulation.

Table 3.12: Mean error variance in drone-tracking simulation.

Q-learning (proposed)	All-raw	All-processing
5.10	5.69	6.58

Similarly to the first scenario, both baselines are outperformed with respect to optimization (A.9), with an improvement of over 10% (see Table 3.12), and the variance MA is consistently smaller for the learned policy, with the exception of some short intervals within the first two windows (see Figure 3.15). Also in this case we observe that the largest improvement is registered at steady-state, while during the transient all curves are very close, with the all-raw configuration performing best at times. This may have two causes: the transient phase is more difficult to explore for the Q-learning, but also, that seemingly sub-optimal behavior during the first two windows might be necessary given that the learning procedure targets the whole horizon. Indeed, the optimal solution need not patch together the policies that optimize different segments of the horizon.

Remark 1 (Energy consumption). As an interesting side effect, the online selection also reduces the total energy consumption, potentially increasing the overall lifespan of the system besides enhancing performance. Considering industrial devices such as Genie Nano cameras (Dalsa (2021)), with typical power consumption of 3.99W for sampling and transmission (tx), and assuming an increase of 0.15W for data processing (Casares, Pinto, Wang, and Velipasalar (2009)), the energy consumption under the confronted sensing policies is shown in Figure 3.16 and Table 3.13. In particular, the proposed learning-based policy only uses 76% of the energy needed to acquire images and transmit them over wireless.

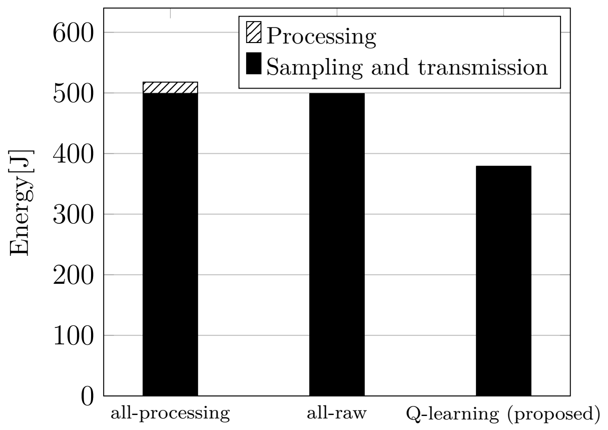


Figure 3.16: Total energy consumption in drone-tracking simulation.

Table 3.13: Energy [J] consumption breakdown during transient (trans.) and at steady state (ss.) in drone-tracking simulation.

	Q-learning (proposed)		All-raw		All-processing	
	Tx	Proc.	Tx	Proc.	Tx	Proc.
Trans.	**59.6**	0	99.8	0	99.8	3.8
Ss.	**319.2**	0	399.0	0	399.0	15.2
Total	**378.8**		498.8		517.8	

3.3.2.3 Discussion: the role of learning in algorithmic estimation

The proposed simulations show that the studied approach can improve performance of smart sensor networks dealing with estimation tasks. In particular, it is noteworthy how a learning framework such as Q-learning can effectively drive the sensing design, leading to improvement with respect to baselines, even with an estimation tool as effective and robust as the Kalman predictor. Indeed, the characteristics of such algorithm, applied to the chosen dynamical systems, are such that one can expect even trivial choices (such as all-raw and all-processing) to yield satisfactory performances. Conversely, a careful design given available options is far from trivial. In fact, even the most simple sensing design bears a combinatorial structure which makes it computationally infeasible to derive the optimal solution. Moreover, submodularity and supermodularity, which allow to analytically bound suboptimality gaps of greedy algorithms (Tzoumas et al. (2021)), may be hard to meet in realistic scenarios – for example, in the presence of delays, out-of-sequence message arrivals, or multi-rate sensors (Ballotta et al. (2020)).

Given these premises, the performance improvements obtained via the studied learning method are encouraging not only with regard to the proposed applications, but mostly in supporting the contribution of such tools to classical estimation and control frameworks, which can benefit from the power of data to circumvent classical computational bottlenecks associated with an optimization-based design. Hence, rather than looking at the two domains of model-based and data-driven control as alternative approaches, we hope with this work to reinforce arguments supporting a unified, hybrid framework that picks the best of both worlds and combines them together.

3.4 Ultra-Wideband for Distance Measurement and Positioning in Functional Safety Applications

UWB technology has attracted substantial attention due to its characteristics and during the Ph.D. it results useful to analyze and implement this technology. Besides the centralized usage of anchor-based systems proposed by Li, Bi, Li, Wang, Lin, and Chen (2018), also point-to-point ranging started receiving much more attention, as proposed by Güler, Abdelkader, and Shamma (2021). Possible topologies were analyzed by Seo, Kim, Noh, and Seo (2017) and by Nguyen, Hanif Zaini, Wang, Guo, and Xie (2018). In addition, the scalability of UWB-based localization was analyzed by Ridolfi, Van de Velde, Steendam, and De Poorter (2018) showing the huge impact of the coordination protocol on scalability.

Also the synchronization algorithms have an important impact on the UWB systems, and it was deeply analyzed by McElroy, Neirynck, and McLaughlin (2014) and by Hamer and D'Andrea (2018). Large-scale networks were analyzed by Macoir, Ridolfi, Rossey, Moerman, and De Poorter (2018) with an anchor-based TDOA strategy. Finally, also the impact of TWR between all nodes in a real-time network was analyzed by Han Sangjin (2010). Once obtained the distances between nodes, trilateration (Yang and Liu (2008)) and iterative multilateration (Hadzic and Rodriguez (2011)) could be done. As proposed by Cao, Chen, St-Onge, and Beltrame (2021) least square optimization could be used to find the coordinate of the nodes by selecting reference nodes to do them. Nguyen et al. (2018) propose a model which was designed and an extended Kalman filter was used for tracking a drone. Moreover, many other machine learning approaches can be used, as proposed by Che, Ahmed, Ahmed, Zaidi, and Shakir (2020), and in particular, deep learning, which has proven its effectiveness in a variety of fields (Cheng, Zhao, Wang, Li, Wang, and Chang (2020), Lu, Sheu, and Kuo (2021)). Despite all these localization techniques through Ultrawideband, there are not many implementations of this technology for safety goals. Indeed many algorithms use a machine learning approach, or they are particularly computational demanding, being not in accordance with IEC 61508. In this context, a simple deterministic algorithm, which is usable in safety critical systems, with small computational resources, for tracking moving targets through UWB was designed. The proposed technique was simply to configure, allowing to invert anchors and distances between them and not requiring a training dataset, as for a machine learning approach. In addition, it was designed for a specific target application of mobile robots involving the presence of human operators. In this scenario, functional safety plays a fundamental role, since it is of prominent importance to avoid the presence of people in the working area of robots during operation. To achieve such a goal, a safety procedure has been designed based on real-time measurement of the distance between robots and human operators as well as on the calculation of the operators' position. Particularly, the safety procedure has to stop robots when the distance with human operators becomes lower than a given threshold. Moreover, the safety procedure exploits positioning to assess whether or not human operators are located in specific safety zones.

A UWB system is used to implement distance measurement and positioning. Specifically, some UWB anchors are located on the robots (which actually implement the safety procedure), whereas human operators are equipped with small UWB tags. This allows distance measurements with techniques like those described in the previous section. Positioning is then achieved via an algorithm that elaborates the distances measured by each anchor.

The described scenario has some requirements, mostly derived from the functional safety features, that are different from those of traditional tracking and positioning systems. Indeed, on one hand, it is fundamental to have a sample rate able to guarantee the timely intervention of the safety procedure. Specific reaction times in this respect are strictly related to the application, nonetheless, typical values are in the range of 200–300 ms. On the other hand, requirements on the accuracy of distance measurement and positioning may be relaxed, since safety areas do not need to be delimited with centimeter precision. Finally, for the considered applications it is essential to limit the complexity of the safety procedures since they are often implemented on low-cost devices (for example, in a prototype application we developed, we used an Arm Cortex–M0 processor).

3.4.1 First test set: two nodes distance measurements

The first work provided an experimental campaign aiming at a precise characterization of two commercially available Ultra-Wideband sensors measuring the distance between each other. The obtained results are presented, discussed, and compared with those of one possible industrial commercial implementation, such as Terabee Follow-meTerabee (2020). Moreover, the obtained results have been analyzed in light of the application requirements. For these tests, two modules including Decawave/Qorvo Ultra-Wideband DWM1001c were used. They have a 6.5 GHz band, with 5 possible channels, all FCC/IEC RF Certified for permanent indoor and outdoor usage. In these tests, the time of flight was calculated using the simplest algorithm (Two-way-ranging) and then the distance was obtained by multiplying the obtained ToF for the speed of light. The tests were carried out to characterize the system in an indoor environment, analyzing the impact of possible obstacles and different distances. The sampling time of the sensors has been set to $25\mu s$. For every performed test, we acquired the measured distance for a time interval of 25 s, leading to the acquisition of 10000 unique measurements. The first test has been executed with the sensors positioned at a distance of 1.23 meters, inside a room with some other objects, but with no direct obstacles. Results are provided in Figure 3.17 and in the first row of Table 3.14, respectively. As can be seen in Figure 3.17, measurements are affected by rather high noise, and a quantization step of about 18–19 mm can be detected. Despite that, the measured accuracy is in agreement with the reference technical specification (which indicates 25 cm) and it is definitely suitable for the targeted applications. Moreover, measurements showed good repeatability, as can be evinced by the low value of the standard deviation. A second set of test has been performed to evaluate the precision of the measurement at different distances, with the sensors in the same room and with some obstacles. Results are provided in Figure 3.18 and Table 3.14, respectively. In particular we are referring to test numbers 2, 3 and 4. In

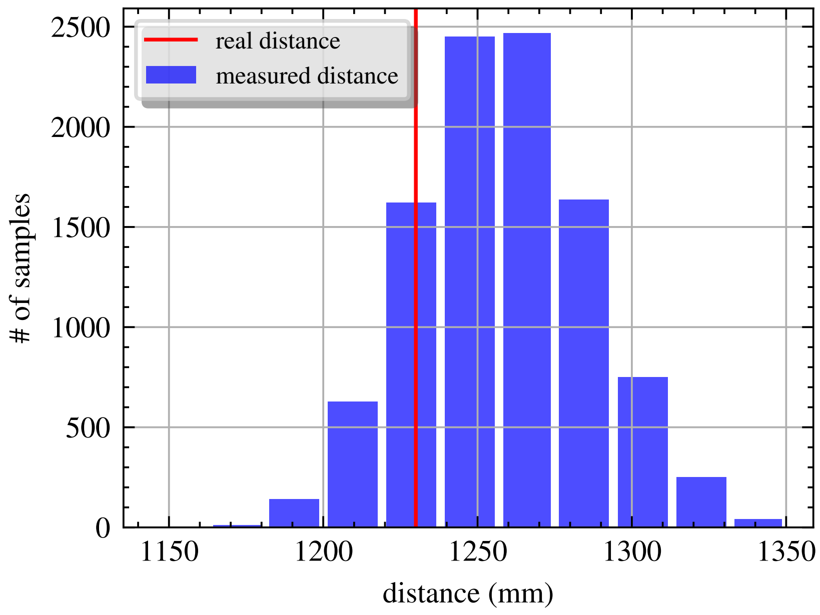


Figure 3.17: Distances measured Test 1

this case, Figure 3.18 plots the Probability Density Functions (PDFs) of the distances.

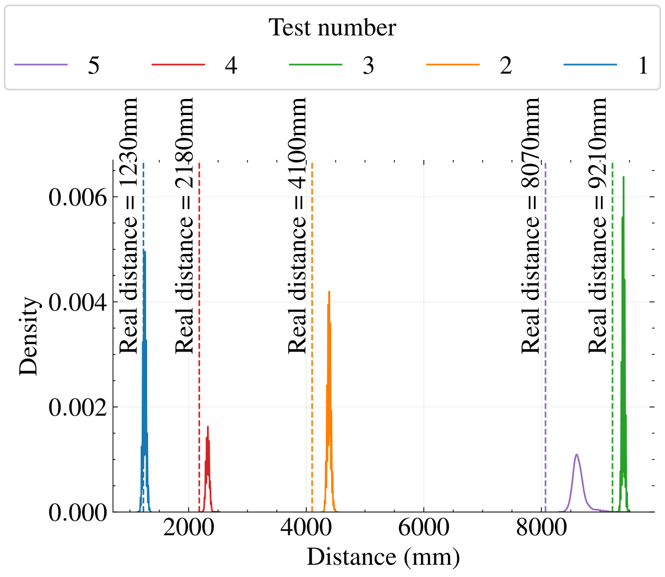


Figure 3.18: Probability density function of the measured distance. Dashed vertical lines represent the real distance.

As can be seen, in some cases, particularly for tests number 2 and 4, the mean measured distance is affected by a non-negligible bias error (about 7% and 6.5%, respectively).

Table 3.14: Experimental results

Test (#)	**Real dist.** (m)	**Mean m. dist.** (m)	**Std. Dev.** (m)	**Mean Err** (m)	**Max Err** (m)	**Obst.** (s/m)	**Diff. rooms** (bool)
1	1.23	1.248	0.028	0.018	0.120	NO	NO
2	4.1	4.387	0.031	0.287	0.421	some	NO
3	9.21	9.394	0.025	0.184	0.283	some	NO
4	2.18	2.323	0.029	0.143	0.334	many	NO
5	8.07	8.636	0.110	0.566	1.329	many	YES

This is most likely due to the presence of objects between the sensors which did not allow a direct path to the signal. However, even if this is an unwanted condition, the overestimation can be properly addressed by the safety procedure. In any case, the detected error is in agreement with the considered technical specifications.
Finally, test number 5 was performed with the sensors in two different rooms, with walls, desks, doors, and many other interfering objects placed among them. From Figure 3.18 and Table 3.14 it is possible to see that measurements remain affected by the bias error, confirming that the absence of a direct path may worsen the measurement accuracy.

3.4.2 Second test set: 2D localization for a possible safety scenario

3.4.2.1 System and algorithm presentation

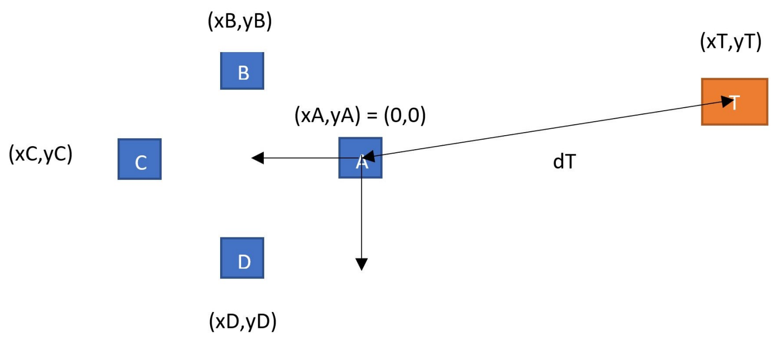


Figure 3.19: Network Topology

In the second session of tests, an Ultra-Wideband band network including four well-known positioned and synchronized anchors and a device to localize were considered. In particular, the localization was done through the TDoA algorithm, with the standard synchronization algorithm implemented by the used module, which has factory closed source firmware. A representation of the network is given in Figure 3.19 where A, B, C, and D are the anchors, whereas T is the device to localize. The distance of each couple $(anchor_i, Device_T)$ was measured in a configuration with some obstacles. Results are presented in Figure 3.20 and Table 3.15), respectively.

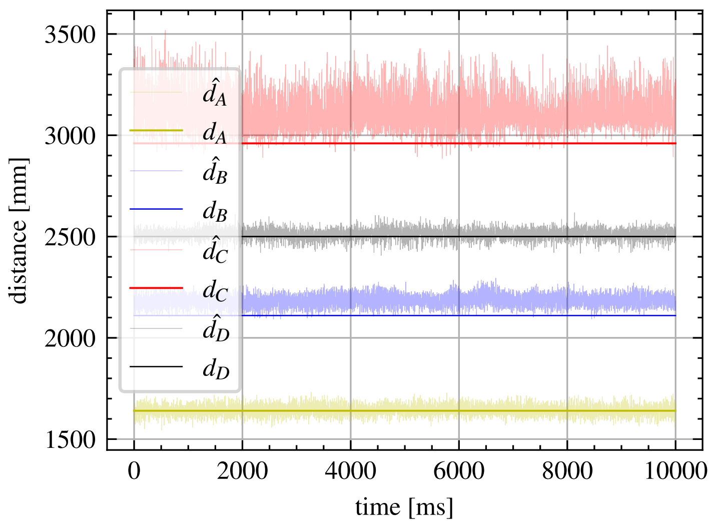


Figure 3.20: Characterization of each sensors with some obstacles

Table 3.15: characterization of each sensors with some obstacles

Sensor (#)	Real dist. (m)	Mean m. dist. (m)	Std. Dev. (m)	Mean Err (m)	Max Err (m)
A	1.64	1.641	0.024	0.001	0.092
B	2.11	2.183	0.028	0.073	0.185
C	2.96	3.108	0.082	0.148	0.558
D	2.50	2.512	0.023	0.012	0.118

From the above measurements, it is not possible to identify the location of the device by simply finding the intersection of spheres with a radius equal to the distances returned by the device and with the center equal to the (known) position of each anchor. An appealing alternative technique is represented by machine learning. However, for the intended applications, this is not a viable option. Indeed, the limited resource devices that are supposed to be used prevent the adoption of machine learning that, moreover, is discouraged in the context of functional safety IEC 61508. Furthermore, the tight reaction times require a fast calculation of the device's position. In addition, the application was designed to be configurable, with possibly different known distances between the anchors.

For these reasons, a machine learning approach would require an important dataset and a long training session. For these reasons, an efficient deterministic algorithm for determining a 2D global position(algorithm 1) was proposed. In particular, the system is simplified in 2D, because the measurement errors shown with many obstacles and walls would cause an even larger error in the third dimension, and also to simplify the computation to ensure real-time performance.
The proposed algorithm, once obtained all the distances between anchors and tag, finds the 2D points which are the intersection of all circumferences with a radius equal to the distances returned by the sensor and with the center equal to the well–known position of each anchor. The position of the intersections is shown in Figure 3.21.

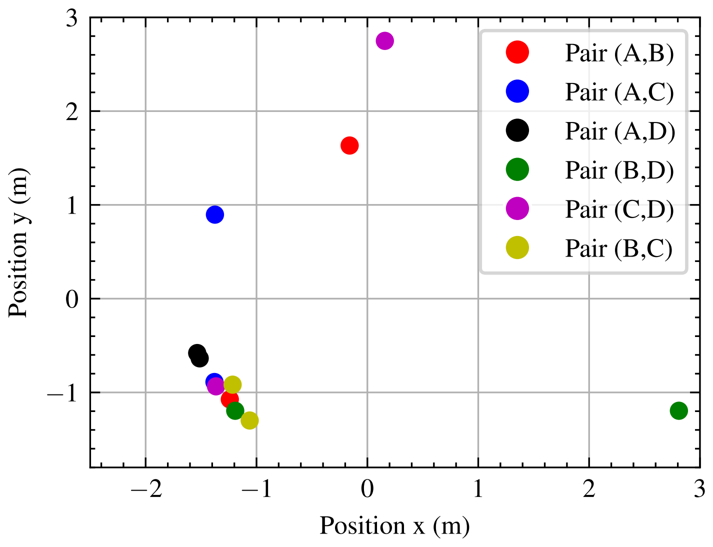


Figure 3.21: Intersection points of the circumferences for every sensors pairs

After that, found at maximum n points, where $n = 2 * \binom{4}{2} = 2 * \frac{4!}{(4-2)!2!} = 12$, the algorithm finds the neighborhood, with center one of those points and with radius r, which contains the maximum number of points (Figure 3.22). If all neighborhoods, with the described properties, contain one point, then the algorithm returns the neighborhood containing at least 2 points with the center one of those points and with the smallest radius $r_n > r$. Once found the neighborhood, the mean point of those contained in the selected collection is returned.

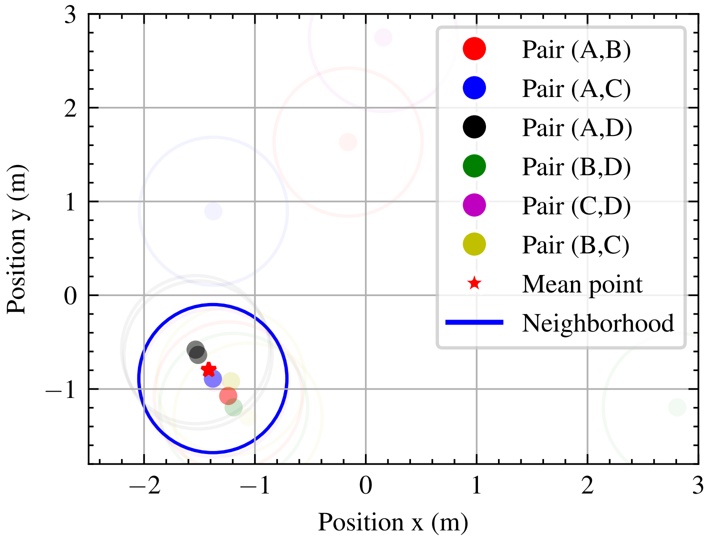


Figure 3.22: Example of the neighborhood chosen with the algorithm

Algorithm 1: Positioning(radius)

```
1  initialized $A_x, A_y, B_x, B_y, C_x, C_y, D_x, D_y$
2  measure(dAT,dBT,dCT,dDT)
3  $listPoints \leftarrow []$
4  for each couple of sensors $(I, J)$ do
5      $Points_{ij} \leftarrow calculatePoint(I_x, I_y, J_x, J_y, dIT, dJT)$
6      $listPoints.append(Points_{ij}[0])$
7      $listPoints.append(Points_{ij}[1])$
8  $numPointInNeigh \leftarrow []$,$pointInNeigh \leftarrow []$
9  maxNumNeigh $\leftarrow 0$, maxIndexNeigh $\leftarrow -1$
10 for i=0;i<len(listPoints);i++ do
11     numPointInNeigh.append(1)
12     pointInNeigh.append(listPoints[i])
13     for $p$ in $(listPoints \setminus listPoints[i])$ do
14         $dist \leftarrow calcDist(listPoints[i], p)$
15         if $dist<r$ then
16             numPointInNeigh[i]++
17             pointInNeigh.append(p)
18     if numPointInNeigh[i]>maxNumNeigh then
19         $maxNumNeigh \leftarrow numPointInNeigh[i]$
20         $maxIndexNeigh \leftarrow i$
21 if maxNumNeigh>1 then
22     return $calcMeanPoint(pointInNeigh[maxIndexNeigh])$
23 else
24     return $Positioning(radius + 0.1)$
```

3.4.3 Experimental results

The algorithm was tested in an experimental campaign with some obstacles between the anchors and the target device. Results are presented in Figure 3.23 and in Table 3.16 respectively.

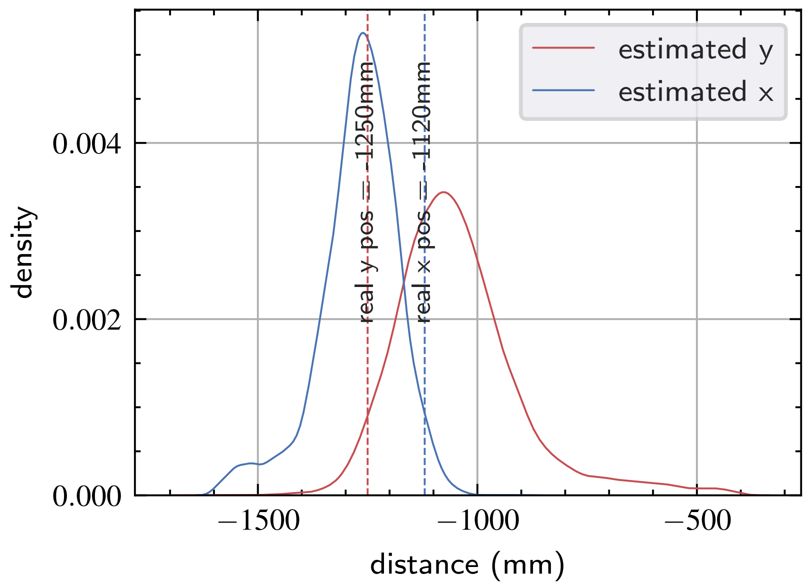


Figure 3.23: x and y coordinate estimation with some obstacles

As can be seen in Table 3.16, the mean estimated position on each axis has a rather limited error resulting in a good estimation of the real position. Unfortunately, both maximum error and standard deviation are definitely too high. Indeed, as can be seen in Figure 3.23, this effect is due to the superimposition of noisy peaks in the measured distance leading to overestimation of the real distance and thus high error and standard deviation. It appears evident that the algorithm itself does not provide satisfactory results in these terms and a further filtering procedure should be introduced. In particular, we assumed that a maximum distance, equal to a reasonable distance (0.5 m) that a person could travel in a sampling time (25 ms), between two consecutive points, could be defined and that it is possible to discard a single anchor distance if identified as too noisy. The results filtering are shown in Table 3.16, where the beneficial effects are evident. As can be seen, standard deviation, mean error, and maximum error are decreased, leading to a more accurate estimation of the real distance with a maximum error lower than 50 centimeters, which results acceptable for the targeted applications.
Results were not compared with any machine learning approaches for the reasons proposed

Table 3.16: coordinates estimation with some obstacles

Test	**Variable dist. (m)**	**Real dist. (m)**	**Mean m. Dev. (m)**	**Std. Err (m)**	**Mean Err (m)**	**Max Err (m)**
no filter	x	-1.12	-1.253	0.076	0.133	0.526
	y	-1.25	-1.073	0.120	0.176	0.896
	distance	1.678	1.655	0.023	0.022	0.189
with filter	x	-1.12	-1.218	0.063	0.098	0.318
	y	-1.25	-1.134	0.094	0.115	0.484
	distance	1.678	1.668	0.026	0.010	0.161

in the previous section and since, cause of the configurability of the system and the small dataset available, those techniques would probably overfit data, proposing an accurate and precise system for this configuration and data, but not for all the possible ones.

4 The Fail Safe over EtherCAT implemented over IEEE802.11

In the era of Industry 4.0 and particular of the Industrial Internet of Things (IIoT), many manufacturing units can be physically located in different distinct places or production areas. In this context, the cabling is increasingly complex and branched, and its cost is always more important. Moreover, wires are difficult to install and maintain. In this scenario, the possibility of removing cables and replacing them with wireless links seems not only a very attractive opportunity but also a fundamental step to incorporate particular applications into the IIoT context. Indeed, traditional Fieldbus communication is not able to provide a complete solution and efficient integration in contexts such as, for example, mobile autonomous collaborative robotics (Refaat, Daoud, Amer, and Makled (2010); Frotzscher, Wetzker, Bauer, Rentschler, Beyer, Elspass, and Klessig (2014)). Wireless extensions of automation networks and fieldbuses have been researched in different forms (Vitturi, Carreras, Miorandi, Schenato, and Sona (2007)), but the new challenge is the use of these systems in a safety critical environment. Therefore in such applications, ensuring the correct transmission of information is a fundamental requirement and this could be difficult with the uncertainty of a unique shared transmission medium. Despite that, the state of art is moving in this direction. For example in both Hashemian (2010) and Hashemian (2011), WSNs have been exploited to monitor safety critical devices in nuclear plants, while they were applied for fire rescue applications by Sha, Shi, and Watkins (2006). Instead, Ikram, Jansson, Harvei, Fismen, Svare, Aakvaag, Petersen, and Carlsen (2013) successfully exploited the black channel to implement a SIL compliant wireless hydrocarbon leak detection system based on ProfiSafe (IEC 61784-3-3). The proposal of using Profisafe over a black channel including wireless technology is proposed also by both Åkerberg, Reichenbach, and Björkman (2010) and Åkerberg, Gidlund, Lennvall, Neander, and Björkman (2011), through a proof-of-concept for the use of WirelessHART (WirelessHART) in safety-critical communications based on ProfiSafe. In addition also Yang, Ma, Xu, and Gidlund (2018) investigated the subject proposing a similar work focusing mainly on an enhanced WirelessHART scheduling to minimize the Safety Function Response Time (SFRT). Indeed, since WirelessHART is based on Time Division Multiple Access (TDMA) and all devices are time synchronized and communicate in prescheduled 10 ms time slots, it lacks asynchronous frames. This fact represents a bottleneck in cases where high priority frames, such as alarms, have to be sent. Despite that, also this problem has been widely addressed with the proposal of several modified Medium Access Control (MAC) layers able to offer additional QoS features. Priority MAC has been proposed by Shen, Zhang, Barac, and Gidlund (2014) which introduces a novel medium access method that enables higher priority traffic to hijack the dedicated transmission bandwidth of the low priority traffic. Zheng, Gidlund, and Åkerberg (2016) proposed WirArb, which was specifically conceived for industrial WSNs, supporting multiple users by pre-assigning a specific arbitration frequency to each node allowing to decide the order of channel access. With this mechanism WirArb ensures that the node with the highest priority will immediately gain channel access, guaranteeing a deterministic behavior. These solutions improve the responsiveness and reliability of the networks but they create ad-hoc non-standardized solutions, in contrast with the IIoT paradigm. To successfully embody Industry 4.0, the proprietary solution must be replaced by an open and standardized solutions (Weyer, Schmitt, Ohmer, and Gorecky (2015)). For this reason, in our work, we decided to follow a more standardized approach. Indeed, in our contribution, we propose an implementation of the Fail Safe over EtherCAT (FSoE) protocol on the top of IEEE 802.11 WLAN. In this way, this implementation can be adopted in a large variety of devices that are equipped with a WiFi interface. Moreover, the future introduction of TSN Fedullo, Morato, Tramarin, Rovati, and Vitturi (2022) on Wi-Fi (Adame, Carrascosa-Zamacois, and Bellalta (2021)), will bring seamlessly all the advantages introduced by the custom MAC layers but with the advantage of high interoperability.

4.1 Background: Functional safety Fieldbuses

The term Functional Safety refers to those (re)active systems able to automatically identify potentially dangerous conditions, thus triggering corrective actions to reduce the level of risk in a system. In this sense, safety is defined as the absence of unacceptable risk, where unacceptable is correlated to the probability and the dangerousness of the risk. Functional safety represents a decisive and crucial requirement in several industrial systems and will play an ever increasing role in the factory of the future, as addressed by the Industry 4.0 paradigm. Indeed, the functional safety global market, including several device types such as safety sensors/actuators, PLCs, is expected to grow with a Compound Annual Growth Rate (CAGR) greater than 8% until 2025, as reported in Functional-safety-market report. Traditional safety systems, based on dedicated hardware circuits or electromechanical parts, will be replaced and effectively improved by Functional Safety Networks (FSNs), namely communication systems used for the transmission of safety-relevant messages, that are designed to implement distributed functional safety systems, as proposed by Buja and Menis (2012).
As a matter of fact, industrial automation systems have made extensive use of communication networks to connect their components, deployed over (possibly large) distributed plant areas, as suggested by Vitturi et al. (2019). This trend started roughly at the beginning of the 90's and progressively enforced over the years, thanks to the improvements achieved by such networks in terms of performance indicators like timeliness, reliability, dependability, and scalability. Nowadays, this scenario is further revolutionized by the adoption of the Industrial Internet of Things (IIoT) (Lu (2017); Jasperneite, Sauter, and Wollschlaeger (2020)) where a pervasive communication infrastructure allows the connection of cloud systems, controllers, industrial equipment, sensors, and actuators to drastically improve the performance of manufacturing systems in terms of product quality, production efficiency, safety, and security (Sisinni, Saifullah, Han, Jennehag, and Gidlund (2018)), Girs, Sentilles, Asadollah, Ashjaei, and Mubeen (2020).
From the architectural point of view, FSNs exploit functional safety protocols logically placed on top of the protocol stacks of *wired* industrial networks. This is the case, for instance, of the ProfiSafe protocol used in Profibus and Profinet, or of the Fail Safe over EtherCAT (FSoE) in EtherCAT systems. FSNs are expected to be ever more deployed and integrated into the factory communication infrastructures, so that plant safety data and information will become part of IIoT ecosystems and, as such, they will be accessed and elaborated using the new services and tools made available in such a context (Mumtaz, Alsohaily, Pang, Rayes, Tsang, and Rodriguez (2017)). Notwithstanding, FSNs are required to ensure the transmission of safety-related information among nodes with

extremely low error probability and bounded reaction times: as a reference, to achieve a SIL 3 (Safety Integrity Level) as specified by IEC 61508, an FSN has to ensure a residual error rate less than $10^{-9}/h$, that is largely lower than what is typically provided by typical industrial networks. This is needed to cope with the strong requirements imposed by safety applications such as motion control (Jeppesen, Rajamani, and Smith (2019)), automotive (Xie, Zeng, Liu, Zhou, Li, and Li (2018); Xie, Li, Han, Xie, Zeng, and Li (2020)) and nuclear power energy plants (John and Bhattacharjee (2020)), to mention some.
However, IIoT ecosystems are characterized by ubiquitous connectivity, as well as by high flexibility of the communication infrastructures, that have to be easily reconfigurable to cope with the dynamic changes of the production schedule. These features may be satisfactorily addressed by *wireless* communications. Thus, it is envisaged that in IIoT ecosystems industrial wireless networks will be ever more deployed. However, the available functional safety protocols have been designed and tested for wired networks, and hence their suitability for typical wireless-based IIoT applications, for example, collaborative robotics and transportation via automatic guided vehicles (Robinson (2019)), is still to be assessed.
The most popular Functional Safety Networks are defined by the IEC 61784-3 International, which calls them Functional Safety Fieldbuses, even if the protocols it introduces are (or can be) adopted by any kind of industrial networks, e.g. Real–Time Ethernet networks, and wireless networks. Despite that, in the standard, there is a specific reference to the Communication Profile Families (CPFs) introduced by the Fieldbus standardization framework. In these terms, Table 4.1 lists the CPFs for which functional safety protocols have been defined along with their commercial names.
The functional safety protocols defined by IEC 61784-3 have been designed assuming the underlying communication system behaves like a *black channel*. With such an approach, the safety nodes are not aware of the channel features, nor of the industrial network they rely on, with the exception of the service primitives necessary for data transmission.
Additional FSNs are available. Two of them, namely OPC UA Safety and CANOpen Safety, are noteworthy. OPC-UA Safety (OPC UA Safety), has been recently defined in agreement with the guidelines of IEC 61784-3, even if it is not (yet) included in such standard. CANOpen Safety (EN 50325–5) is a popular European Standard, designed with a different approach, referred to as *white channel* from that of IEC 61784-3. From a protocol architecture point of view, the nodes of a FSN are characterized by a Safety Communication Layer, placed on top of their protocol stacks, if a *black channel* approach is used. Instead, if a *white channel* approach is used, it is assumed that the functional

Table 4.1: Communication Profile Families (CPF) and Functional Safety Protocols defined by IEC 61784-3

CPF	Commercial Name	Functional Safety Protocol
1	FOUNDATION Fieldbus	FF-SIS
2	Common Industrial Protocol (CIP)	CIP Safety
3	PROFIBUS & PROFINET	PROFIsafe
6	INTERBUS	INTERBUS Safety
8	CC-Link	CC-Link Safety
12	EtherCAT	Fail Safe over EtherCAT (FSoE)
13	Ethernet POWERLINK	Ethernet POWERLINK Safety
14	EPA	EPASafety
16	SERCOS	CIP Safety
17	RAPIEnet	RAPIEnet Safety
18	SafetyNET p	SafetyNET p

safety protocol is well aware of the underlying communication system and makes use of its services and protocols, at all layers, to implement the safety functions. The representation of these two stacks is presented in Figure 4.1

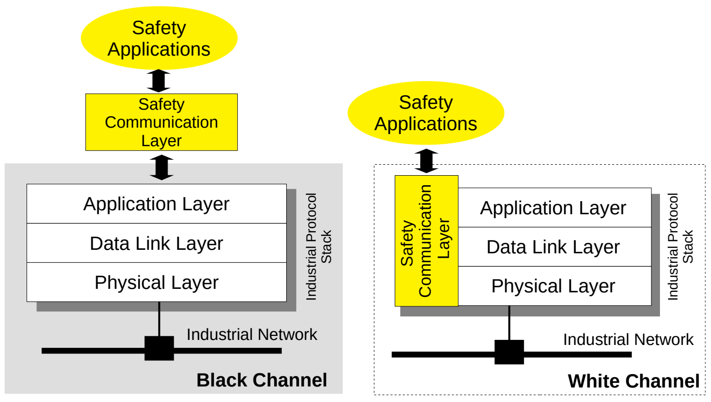


Figure 4.1: Functional Safety Protocols: Black and White Channel Approach

Regardless of the channel approach, functional safety protocols have to deal with several types of communication errors. The most typical ones, as specified by IEC 61784-3, are corruption, loss, and delay of the transmitted safety messages. To tackle such impairments, the protocols adopt some countermeasures, the most effective being the use of enforced CRC, message numbering, and timestamping. The complete list of errors and countermeasures can be found in the standard documents.
When a functional safety network is used (in a plant, machinery, equipment, etc.) it definitely contributes to the risk analysis, which has to determine the overall Safety

Integrated Level (SIL). In particular, IEC 61508 recommends that the adopted communication facilities may influence with a maximum percentage of 1% the average frequency of dangerous failures per hour of the safety functions (assuming that every single error could lead to a dangerous failure). This reflects on the performance figures that FSNs have to provide, in terms of residual error probability per hour (REP). Clearly, the higher the SIL, the lower the REP. In general, REP is strictly related to the communication behavior and depends on:

- the bit error rate of the underlying channel;
- the number of safety messages transmitted per hour;
- the countermeasures against errors adopted by the functional safety protocols;

4.1.1 Fail Safe over EtherCAT

As previously seen, FSoE is referred to as Functional Safety Communication Profile 12–1 by IEC 61784–3, and it has been conceived for deployment in conjunction with EtherCAT.
FSoE is a Master-Slave protocol, with a unique device referred to as FSoE master, and several FSoE slaves. During normal operation, the FSoE master cyclically polls the FSoE slaves. The data exchange takes place over FSoE connections, which are virtual communication channels established between the FSoE master and each FSoE slave in the initialization phase.
The FSoE PDU has two different formats depending on the amount of safe data bytes that has to be exchanged. The simplest format is shown in Figure 4.2 and it is used to transfer a single byte of safety data from master to slave and vice versa. The first byte contains the command that identifies the specific state of the FSoE connection. This allows to determine the meaning of the safety data. The command field is followed by the data field, by the CRC (2 bytes), and then by the Connection Id (2 bytes).

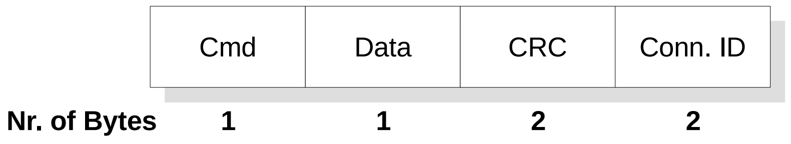


Figure 4.2: Basic FSoE frame

The CRC is calculated in a more elaborated way, with respect to traditional protocols. Indeed, it is determined on a data structure, which includes more fields than those of the Safety PDU. They are reported in Table 4.2.

Table 4.2: FSoE: Fields used for CRC Calculation

Field Nr.	Field
1	received CRC (bit 0-7)
2	received CRC (bit 8-15)
3	ConnID (bit 0-7)
4	ConnID (bit 8-15)
5	Sequence Number (bit 0-7)
6	Sequence Number (bit 8-15)
7	Command
8	SafeData[0]
9	0
10	0
11	0

As can be seen, there are two fields that refer to the sequence number. This is a 16–bit

counter ranging from 1 to 65535, re–initialized to 1 when the maximum value is reached, which represents the progressive number of the PDU transmitted by a safety device. Each safety device handles its own sequence number, which represents the progressive number of the safety PDU it transmits. A device also maintains the sequence number of the Safety PDU it expects to receive from its partner. The comparison between these two values (that are expected to coincide) is achieved by inserting the sequence number in the structure used to calculate the CRC. Moreover, as described by IEC 61784–3, three additional null octets are included in the structure used for CRC calculation.
The second PDU format, which is described in Figure 4.3, is used when more than a single byte of safety data has to be transferred.

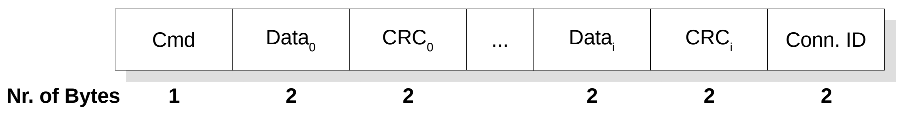


Figure 4.3: FSOE extended frame

As can be seen, safety data is structured in blocks of two bytes, each followed by the relevant CRC. From the above analysis, it emerges that FSoE uses the same countermeasures adopted by PROFIsafe against communication errors: Sequence Number (implemented via the progressive number assigned to PDUs), Time Expectation (via the watchdog timer), Connection Authentication (via the rigid assignment of Connection Ids) and Data Integrity Assurance (via the CRC).

4.2 Implementation

The FSoE stack has been developed and implemented on two different device types, namely a Raspberry Pi board and a PC running the Linux operating system. Both devices have Wi–Fi interfaces and are equipped with the full UDP/IP protocol stack. The first implementation of FSoE has been made on top of the User Datagram Protocol (UDP), practically including the FSoE frame in the UDP one as proposed in Figure 4.4 With this type of implementation, as evidenced in Figure 4.4, the black channel comprises the communication medium, the WiFi modules of the adopted devices, and their UDP/IP suites (indicated as protocol stacks) up to the interface with the respective FSoE stacks. As can be seen always in Figure 4.4, the transmission of safety data is started by a safety application running on the master that issues a request to the FSoE protocol stack. The arrival of the safety data is notified via an indication primitive to the safety application of the slave, which issues the response primitive carrying the response data. The service is then concluded with the arrival of the confirmed primitive to the master.

Clearly, if during the elaboration within a node (either master or slave) an error is introduced during the transmission through the black channel, data are protected by all the counter-measurements of the FSoE protocol, which detects the problem.

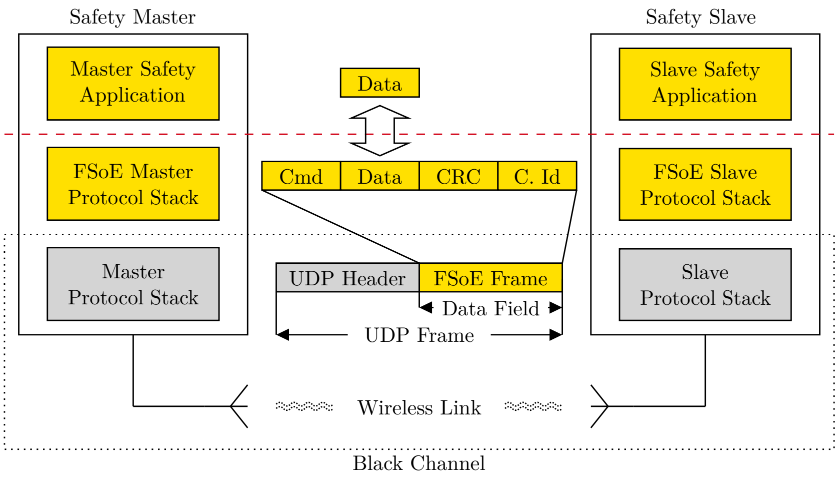


Figure 4.4: Implementation of the FSoE Protocol Stack

In addition, it is clear that a WiFi network has a completely different topology with respect to EtherCAT, thus a different addressing technique needed to be devised. In practice, the FSoE Master hence assigns a “connection ID” to each slave and associates it with the IP address of the slave itself. This allows the delivery of safety data to the right slave. Remarkably, this custom addressing technique has neither impact on FSoE, i.e. it has not required any modification to the FSoE protocol stack, nor on the safety performance. Moreover, this type of addressing technique adds a further useful safety feature, because it allows to obtain a double check of the correct addressing of the slave, thanks to the univocal correspondence between IP address and connection ID. Indeed, both slave and master know both their connection ID and IP address. Therefore, upon receiving a frame, they can check whether the addressing is correct or not.
To test the performance of FSoE over WiFi, an experimental setup has been prepared where the FSoE master protocol was implemented on the PC, whereas the FSoE slave was the Raspberry Pi board. The network was configured to use the IEEE 802.11n at a rate of 72 Mbits/s in the 2.4GHz band with the Request To Send/Clear To Send (RTS/CTS) mechanism enabled. Notably, RTS/CTS is usually disabled, when WiFi is used for industrial real-time applications since it might introduce additional latency. In this case, however, it has been used since it contributes to increase communication reliability (Willig et al. (2005)).
From the IEEE 802.11 network point of view, the node acting as FSoE master is config-

ured as an Access Point (AP), whereas the FSoE slave is a Station (STA), as shown in Figure 4.5.

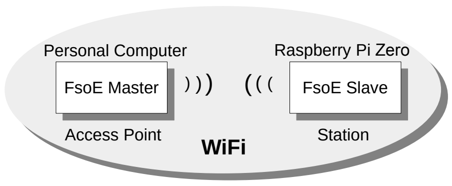


Figure 4.5: Scheme of the Experimental Setup

4.3 Introduction of used performance indicators

For the evaluation of the following results, three meaningful performance indicators were evaluated:

- the polling time of the slave
- the percentage of FSoE lost packets
- the Safety Function Response Time.

Indeed, in practical applications, the delivery of Safety PDUs from the master takes place cyclically with a period, T_{cycle}, determined by the requirements of the application itself. Since all the slaves must be queried within a period, the polling time of a single slave represents a meaningful index of the protocol performance. In addition, also the number of packets lost is an essential parameter for safety protocol, indeed IEC61784-3 correlates this parameter to the maximum achievable SIL. Finally, the Safety Function Response Time, which is explained in dedicated session 4.3.2, is another essential parameter, typical of safety critical protocols.

4.3.1 Polling Time of a FSoE Slave

The polling time of a slave has been calculated as the time that elapses between the generation of a FSoE frame transmission request by the master and the reception of the confirm primitive from the slave. As can be seen in Figure 3.1, the transmission sequence includes the times necessary to

- execute the FSoE protocol stacks in both master and slave;
- execute both master and slave standard protocol stacks;

- transmit the frame from master to slave and vice–versa.

Under the assumption that the stack execution times are the same for both master and slave, the polling time Tp can be expressed as

$$T_p = 2T_{FSoE} + 4(T_{stack} + T_{MAC}) + 2T_{tx} \tag{4.1}$$

where T_{FSoE} is the time to execute the FSoE protocol stack, T_{stack} is the time to execute the protocol stack, T_{MAC} is the execution time of the last two bottom layers of the protocol stack, and T_{tx} is the time to transmit a frame. It is worth remarking that the term relevant to the execution time of the FSoE stack in Equation 4.1 is $2T_{FSoE}$ since the generation of both the response and confirm primitives are made automatically by the stack, i.e. the FSoE protocol stack is executed only once at both the master and the slave units.

4.3.2 Safety Function Response Time

Safety Function Response Time(SFRT) is an interesting index to assess the performance of safety critical automation systems which use functional safety networks. It represents the time elapsed between the detection of an error and the transition of the system into a safe state. In the context of the IEC 61784–3 International Standard, the SFRT indicator has been specifically introduced by one of the protocols within the specifications, namely ProfiSAFE (IEC 61784-3-3). In particular, it is defined as "the worst-case time to reach the safe state of the system in the presence of errors or failures in the safety functions or in the communication medium itself". With respect to the reaction time (T_r), which in a pure master-slave could be defined as:

$$T_r = 2T_c = 2\sum_{i=1}^{N} T_{pi} \tag{4.2}$$

where T_c is the cycle time (i.e. the time necessary to execute the polling of all the slaves), N is the number of slaves, and T_{pi} is the polling time of the i–th slave; SFRT represents a more comprehensive index since it explicitly includes the case of errors in the communication system. Although its specific formulation is related to ProfiSAFE, it has been subsequently characterized by Åkerberg, Gidlund, Lennvall, Neander, and Björkman (2011) and Pimentel and Nickerson (2014) for other protocols. In the following, we provide an analytical description of SFRT for FSoE.
In ProfiSAFE, SFRT is calculated under the hypothesis that during the transmission of a safety PDU there may be at most a single faulty device in the safety network. This means

that the procedure of reaching the safe state by a system has to consider, in the worst case, also the possibility of a fault in one of the devices of the network. The detection of such fault is made possible by watchdog timers used by the safety devices. Thus, since ProfiSAFE is based on a master–slave protocol, if a slave does not respond to the query of the master within a timeout, then the master marks that slave as faulty and moves to the next one. Conversely, if a slave is not polled by the master within a timeout, then it enters the safe state.
Let us assume the safety network is composed by one master and n slave devices. If a slave detects a fault, then the maximum time to reach the safe state is given by

$$SFRT = WCDT + WDTout^{max} \tag{4.3}$$

where $WCDT$ is the Worst Case Delay Time, i.e. the maximum time requested by the communication between the master and the slaves with the exception of the faulty slave. The contribution of this latter to SFRT is given by the term $WDTout^{max}$, which represents the maximum value of the watchdog timeout among the slaves, formally expressed as

$$WDTout^{max} = \max_{i=1,2,\ldots,n}(WDTout_i). \tag{4.4}$$

Moving to FSoE, if a slave detects a fault, in the worst case it will notify to the master after a cycle time (this happens when the fault detection occurs just after the slave has been polled by the master and hence it has to wait for another cycle before it is polled again). Similarly, another cycle is necessary for the transmission of the safety messages by the master to all the other slaves. However, to calculate the SFRT it has to be taken into account that one of the slaves can be faulty. Hence we have

$$SFRT_{FSoE} = T_C^{max} + \sum_{\substack{i=1 \\ i\neq j}}^{n} T_{p_i}^{max} + \max_{i=1,2,\ldots,n}(WDTout_i) \tag{4.5}$$

where T_C^{max} is the maximum FSoE cycle time, $T_{p_i}^{max}$ is the maximum polling time of the i–th slave, and j accounts for the faulty slave. Clearly, the first two terms in in Equation 4.5 account for the term $WCDT$ in Equation 4.3.

4.4 Assessment with different transport layer and the proposal of UDP_c

4.4.1 First experimental assessment

The first test application has been developed in which the FSoE master continuously polls the slave. In practice, the master application invokes a request primitive to the FSoE master stack to send a safety message to the FSoE slave application and, upon reception of the confirm primitive from this latter one, it immediately starts a new request. These tests were executed in a laboratory environment where in–band interference was controlled. Also, in this direction, we accurately selected the Wi-Fi channel that showed the lowest traffic. The experiments lasted 5 hours. Two meaningful performance indicators were evaluated: the percentage of FSoE lost packets and the polling time of the slave. During the test, 10 packets got lost in total, resulting in a ratio of lost packets equal to 1.04×10^{-6}. Concerning the polling time, Figure 4.6 shows its empirical probability density function (to ease readability, the figure reports data comprised in the interval median $\pm$ one standard deviation). As can be seen, the behavior is slightly multi-modal with the main peak centered at about 1200μs. It also denotes a considerable jitter, which might prevent the use of this network in the most demanding applications. The obtained results, particularly the percentage of lost packets, led to conclude that, unfortunately, the SIL3 can not be reached. Indeed, SIL3 requires that a FSoE connection can be re-initialized at most once every 5 hours. Since each lost packet corresponds to one re-initialization, in this experiment 10 re-initializations occurred.

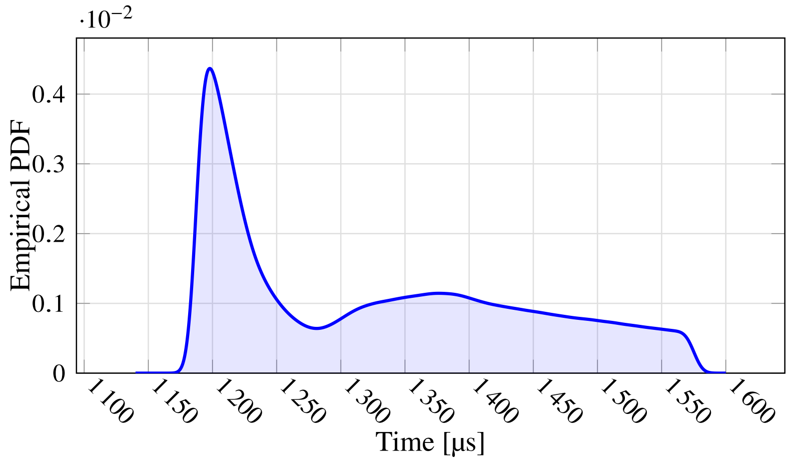


Figure 4.6: Empirical Probability Density Function of the Polling Time

The calculation of SFRT is straightforward from Equation 4.5. Considering that the watchdog time has been set to 250.000μs on both master and slave, it results:

$$SFRT = 17942.40 + 250.000\,\mu s = 267.942,40\mu s \tag{4.6}$$

4.4.2 The application of different transport layer

The implementation discussed so far of FSoE over IEEE802.11 based on the encapsulation of SPDU into UDP frames, in a first analysis, revealed the significance and effectiveness of the proposed approach, particularly concerning the applicability of that implementation for some specific application areas. Despite that, the performance achieved in terms of packet loss was still not sufficient to meet the strict FSoE requirements. These performances could be improved by changing the transport layer. Indeed UDP is a connectionless protocol that does not encompass any QoS mechanism in which the successful delivery of a PDU completely relies on the robustness of the transmission medium. With this lack of flow control and retransmission mechanisms, when a frame is lost or corrupted, there is no possibility of requesting its retransmission. In this part, the usability of different transport layer protocols to carry SPDU is discussed. Moreover, the results obtained with the previous experimental setup and with different transport layers are presented, showing their impact on polling time, on number of lost packets and on SFRT.

4.4.2.1 UDP caching layer

The simplest approach to improve packet loss is to replace UDP with TCP, indeed this protocol provides Quality of Service (QoS) features, through error detection, flow control and re-transmission of undelivered packets. Despite that, there are many features of UDP that could result particularly useful. In detail this protocol provides a broadcast service which allows to smartly manage the network with a faster communication and smaller communication. Moreover, UDP provides a smaller header, improving the effective data rate and the effective time performances. In this context it could be useful to keep UDP and add the QoS features to it through an additional caching layer. In this way the standard protocol would be maintained, with its features, and the additional QoS characteristics are designed. As proposed in Figure 4.7 , the additional intermediate layer, act as an interface between UDP and the FSoE stack, both in Master and Slave implementations. The caching layer was designed to store both the incoming from the protocol stack and outcoming SPDU from the FSoE stack. For every outcoming frame, the caching layer expect to receive an incoming frame with in a predefined deadline T_D.

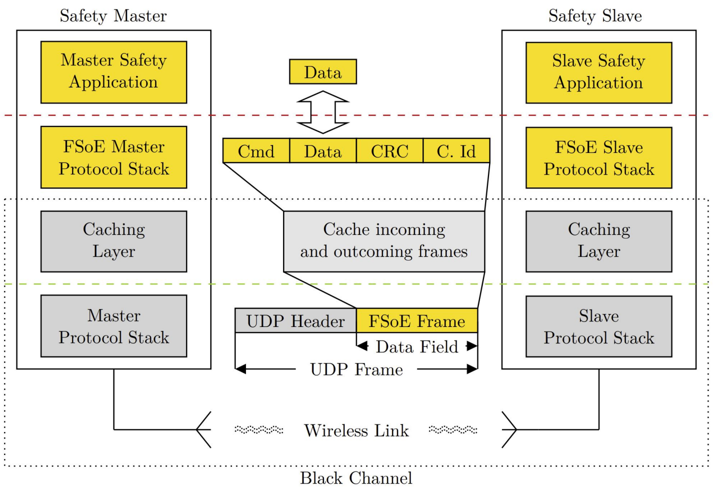


Figure 4.7: Implementation of FSoE stack with UDP and a caching layer

In case of packet loss, as proposed in Figure 4.8 when the deadline T_D expires, the caching layer perform a retransmissions of the stored outcoming frame. This deadline is completely independent to the FSoE watchdog which will continue to monitor autonomously for communication delays and trigger the transition to the fail-safe state, i.e. the reset of the FSoE connection, if a valid SPDU is not passed to the FSoE stack within the preset watchdog time. Defined T_W the FSoE watchdog period and T_p the polling time of a

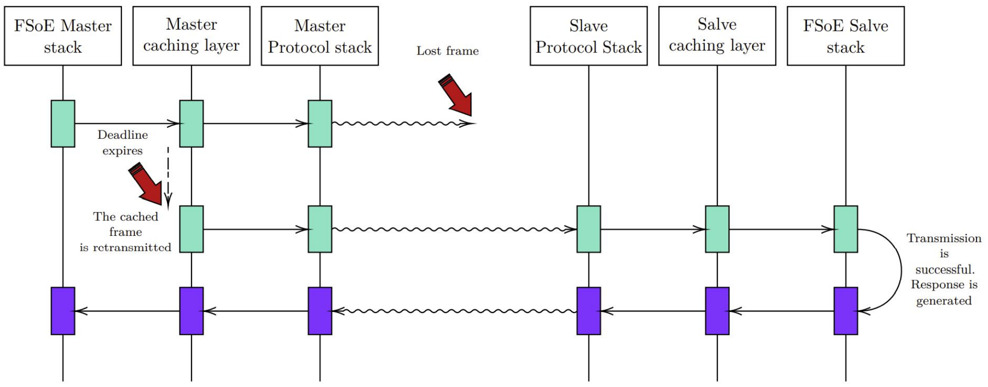


Figure 4.8: Handling of caching layer retransmission in case of packet loss

device, to effectively exploit the retransmission mechanism provided by the caching layer, it must be

$$T_p < T_D < T_W \tag{4.7}$$

Clearly with $T_D \geq T_W$, the FSoE watchdog would be triggered before the retransmission, making the caching layer completely useless and so obtaining a common UDP behavior. In general to avoid it, it is necessary that T_D is smaller than T_W at least of $T_p + \epsilon$, where ϵ is the elaboration time of the caching layer. Indeed if an error is detected, the retransmission mechanism must work and deliver the message before the watchdog, otherwise all the retransmission was useless. At the same time it is mandatory that $T_D > T_p + \epsilon$, otherwise a retransmission start before the pack could be effectively correctly delivered. So this general law is obtained:

$$T_D + T_p + \epsilon < T_W \quad and \quad T_D > T_p + \epsilon \tag{4.8}$$

A possible approach to select the retransmission deadline would be by imposing

$$T_D = \frac{T_W}{M} \tag{4.9}$$

where M represents the number of retransmissions that the caching layer will perform within a FSoE watchdog period, such that $T_D > T_p + \epsilon$. Clearly, the optimal value of M should be statistically determined, in order to fulfill the requirements on the maximum number of resets of the FSoE connection, and in accordance with the condition of the channel, which introduces other delays. Also, the generation of duplicate packets is a condition to be avoided since, as specified in Section 4.1.1, it would result in the FSoE connection being reset. The handling of duplicate frames is done by exploiting one of the fundamental features of SPDUs. Indeed, FSoE is designed to ensure that two consecutive frames should differ at least of one bit. This is ensured by the use of consecutive sequence numbers calculated in the CRC as explained in Table 4.2. Due to this characteristic, if are received two identical frames, those are certainly duplicated frames. In this situation, as can be seen in Figure 4.9, the duplicated frame is detected by caching layer by comparing the received frame with the previously stored incoming frame. Consequently, the previously stored outcoming frame is sent, without executing the FSoE stack.

The working principle of the caching layer is summarized in algorithm 2. The layer continuously polls for an incoming frame from the transport layer, which is eventually acquired. The incoming frame is compared against the previously stored incoming frame, and in case they differ, the FSoE stack is executed and the generated response, as well as the incoming frame, are locally stored for future comparisons. If the incoming frame matches the one received on the previous transmission, the caching layer simply responds with the stored outcoming frame without triggering the execution of the FSoE stack

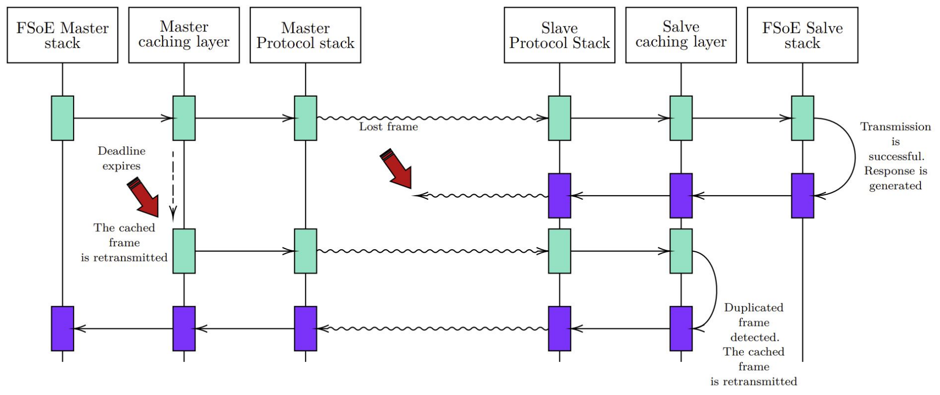

FSoE Master stack
Master caching layer
Master Protocol stack
Slave Protocol Stack
Salve caching layer
FSoE Salve stack
Lost frame
Transmission is successful. Response is generated
Deadline expires
The cached frame is retransmitted
Duplicated frame detected. The cached frame is retransmitted

Figure 4.9: Handling of caching layer retransmission in case of packet loss

thus avoiding the reset of the FSoE connection. When there are no incoming frames, the algorithm monitors whether the deadline for receiving the response is exceeded. If so, then the previous outcoming frame is retransmitted. Moreover, the callbacks from the FSoE stack in case a violations of the watchdog are handled allowing the device to be forced into a safe state.

Algorithm 2: Caching layer

```
1  while true do
2      if has imcoming frame from UDP then
3          incomingFrame = ReceiveFsoeFrameFromUdp()  if incomigFrame !=
            storedIncomingBuffer then
4              reset deadline $T_D$  storedIncomingBuffer = incomingFrame
                outcomingFrame = executeFsoeStack(incomingFrame)
                storedOutcomingBuffer = outcomingFrame
5          else
6              outcomingFrame = storedOutcomingBuffer
7          end
8          TrasmittFsoeFrameToUdp(outcomingFrame)
9      else
10         if deadline $T_D$ is expired then
11             TrasmittFsoeFrameToUdp(storedOutcomingBuffer)
12         end
13         update deadline
14     end
15     update watchdog  if watchdog is expired then
16         trigger fail safe condition  reset FSoE stack
17     end
18 end
```

4.4.2.2 Results with different transport layer

The assessment of the FSoE protocol over WiFi with different transport layers was carried out in a noisier environment to have the possibility of seeing more loss and QoS mechanisms. Specifically, in this case, the tests were conducted in a laboratory with more people and equipment, where there was no control over the sources of interference. The transport layers used were represented by UDP, TCP, and again UDP but with the additional caching layer proposed in the previous part (which following was referred with UDPc). The results of the experimental campaign are reported in Table 4.3 (packet loss), Table 4.4 (polling time statistics) and Figure 4.10 (polling time empirical PDF).

Packet loss			
	Total pkt	pkt lost	% pkt lost
UDP	7817020	25	0.00032
TCP	7457630	1	0
UDPc	7726440	0	0

Table 4.3: Lost FSoE Packet Statistics for different transport layer on IEEE802.11

Polling time (µs)				
	mean	std	min	max
UDP	2277.18	1113.25	1183.89	43128.30
TCP	2366.40	1144.27	1257.90	225480.00
UDPc	2281.71	1113.14	1187.54	51522.90

Table 4.4: Polling time statistics for different transport layer on IEEE802.11

At a first glance, the (negative) effects of the noisier environment appear evident with respect to the results proposed in sec. 4.4.1. Indeed, the performance figures of the protocol when UDP is used are worse than in the previous experiments. Specifically, as proposed by Table 4.4, Table 4.3 and Figure 4.10, there were 25 lost packets (15 more with respect to the previous case) whereas the polling time resulted, on average, about 685 μs longer. However, the two additional strategies considered for the transport layer revealed interestingly better results. As it can be seen, with TCP the number of lost packets was dramatically reduced, even if at the expense of an increased polling time. This is clearly due to the quality of service features introduced by the TCP protocol that ensures a more robust data transmission but requires longer execution times since the protocol stack is definitely more complex than that of UDP. Even better performance figures are provided by UDPc. In this case, no packets were lost and the polling time resulted very close to that obtained with UDP. This is because the caching layer introduces only the

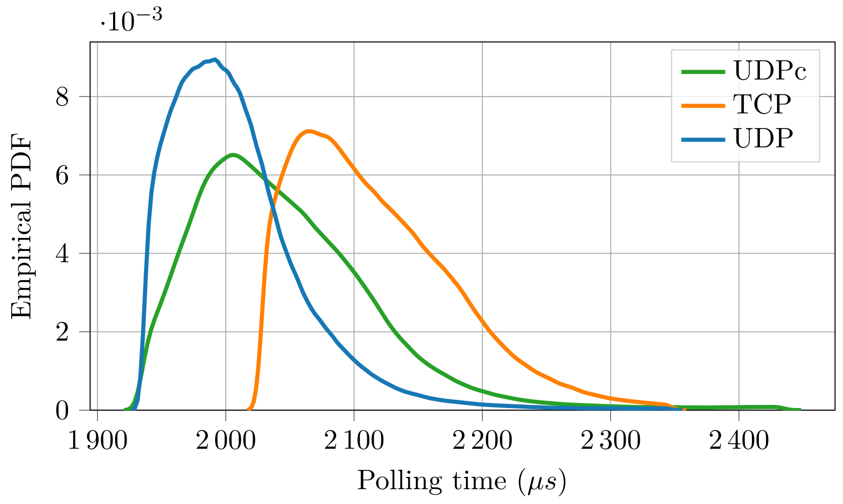


Figure 4.10: EPDF of the polling time for different transport layer on IEEE802.11

services necessary to support the FSoE protocol (handling of packet retransmission and duplication), resulting in an efficient stack, with very limited impact on performance.

The SFRT for the three different transport layers are shown in Table 4.5.

$\mathbf{SFRT_{FSoE}}$ (μs)	
UDP	293128.30
TCP	475480.00
UDPc	301522.90

Table 4.5: SFRT values for the three transport layers

From the results proposed, the watchdog timeout has a noticeable impact on the SFRT. Clearly, by decreasing it the system would result more responsive to communication problems and therefore able to force the entire system to the safe state much more quickly. However, reducing the watchdog timeout could imply further drawbacks. Indeed, especially when wireless networks are used, the use of a short watchdog timeout might lead to erroneously consider a device as faulty (because it does not answer within the timeout) while, conversely, its polling is simply delayed due to temporary decay of the communication channel status. Thus, the choice of the watchdog timeout is a trade–off between the ability of the system to quickly react to fault situations and the necessity of ensuring adequate safety performance.

4.4.3 Discussion

The results of the experiments confirm the potential effectiveness of the black channel approach, as they demonstrate the possibility of implementing the FSoE protocol over WiFi and, more generally, over networks for which it was not explicitly designed, without any modification to the protocol itself. Therefore the designed SIL level is maintained if some base performances could be guaranteed. In this way, it is possible to use safety stacks that have been certified out-of-context (i.e. regardless of the application for which they are used) without the need for a new safety assessment.
Also the adoption of the proposed UDPc does not lead to any UDP modification. Indeed, the caching layer is an additional software component that acts as an interface between the safety layer and the transport layer. Furthermore, the caching layer is both protocol and platform agnostic, i.e. its operation does not depend on the hardware/software platform in which it is used, nor on the protocols used for the PDU exchange. This makes it highly interoperable and can be used in any embedded platform. Analyzing separately the two experimental sessions, some additional comments can be done. Concerning the first one, it highlights a resulting polling time of about 1.2 ms and a ratio of 10 lost packets every 5 hours. This packet loss result does not comply with the limit imposed by the regulation indeed it imposes a maximum rate of 1 lost packet every 5 hours. However, it can be pointed out that the proposed setups of wireless-FSoE can be used in less demanding application scenarios. In this context, the second experimental campaign wanted to propose an implementation that could be compliant with the requirements imposed by SIL3. In this sense, the proposed results with TCP and UDPc are really promising and pave the way to a wide range of applications. Instead, the UDP ones confirm that, with the proposed stack, it is not possible to achieve that safety level. Regarding the timeliness, it has been observed that the overhead introduced by TCP provides also a considerable increase in polling time. In this sense, UDPc maintains the fundamental characteristics of UDP, such as essential broadcasting messages, allowing to obtain superior performance (in the order of hundreds of microseconds) to TCP while ensuring the same degree of reliability in the transfer of SPDUs.
As a general observation, the safety over wireless systems may not satisfy the requirements in terms of SFRT precisely because of the longer transfer times of the SPDUs and the inability to guarantee tight deadlines compared to the wired counterpart. This situation can be mitigated by adopting TSN on WiFi which will allow to achieve wire-equivalent reliability with deterministic time and timeliness performance over wireless (Cavalcanti et al. (2019)). These achievements can likely be extended to other protocols that adopt the same approach, e.g., those defined by IEC 61784-3. Among them, ProfiSAFE seems

a very suitable candidate, since such a protocol has some common features with FSoE (for example, the master–slave strategy as well as some of the countermeasures against communication impairments) and openSAFETY, whose implementation over IEEE802.11 was analyzed by Hadziaganović, Atiq, Blazek, Bernhard, and Springer (2021). Another safety protocol that could easily work over wireless networks is OPC UA Safety. In this case, the safety stack is implemented (via a suitable mapper sublayer) on top of the OPC UA protocol stack which, in turn, relies on the TCP/IP suite. Thus, the safety protocol is not aware of the underlying physical layer. In this respect, the assessment provided in Morato, Vitturi, Tramarin, and Cenedese (2021b) clearly shows that the use of different physical layers for OPC UA applications may actually have an impact on the performance, but the feasibility of such applications is ensured whatever the adopted physical layer.

4.5 Simulator implementation

As proposed by the previous section, the results obtained from the different experimental campaigns showed the effective possibility of implementing FSoE over a network for which it was not explicitly designed, without any modification to the protocol itself. These outcomes revealed appealing capabilities, but at the same time, they highlighted the risk of implementing a stack that is not able to achieve the required safety level. In this context, a performance examination is essential before starting the effective implementation. Despite that, a theoretical analysis could result to be too difficult without the correct instruments. In this sense, a simulator of the designed network should be the perfect tool.
Clearly the implementation of the simulation model should be accurate and able to reproduce realistic representations of the industrial wireless environment behavior. For this purpose, the FSoE over Wi-Fi protocol has been implemented using the widespread, discrete event OMNeT++ simulator. In detail, FSoE was implemented using UDP and the validated Wi-Fi stack made available by OMNeT++ (Bredel and Bergner (2010)). However, as proposed in chapter 3, these models often implement generic calibrations that can simulate a wide range of scenarios but are unlikely to reproduce specific use cases. Therefore, to be able to carefully simulate a Wi-Fi-based FSoE network, it becomes imperative to set up a precise calibration phase of both the channel errors models, which was done through the calibration proposed in 3.2.2, and the polling time, which is representative of the time necessary to complete the communication cycle between two devices.

4.5.1 Calibration of the polling time

After the calibration of the channel errors models, a subsequently focus on the FSoE prototype implementation was done, evaluating the Polling Time (t_P) between a FSoE Master and the FSoE Slave. As previously pointed out in section 4.3.1, t_P includes the time necessary to execute both the FSoE and the underlying protocol stacks in both master and slave, and the time to transmit the safety frame and the slave's answer message. The latter time, assuming no collision during the transmission Tramarin et al. (2016), may be considered deterministic. Conversely, the former time usually depends on the characteristic of the device where protocols and applications are implemented, such as the computational capabilities, as well as operating system calls, memory management, etc., which may introduce random latencies and jitter. OMNeT++ in some way tries to take into account these uncertainties by introducing some uniform random delays, but

obviously, they cannot correspond to real use cases.
Therefore, a further set of tests were relevant to the calibration of the simulator with respect to experimental values. To this goal, a suitable measurement setup has been designed, as shown in Figure 4.11.

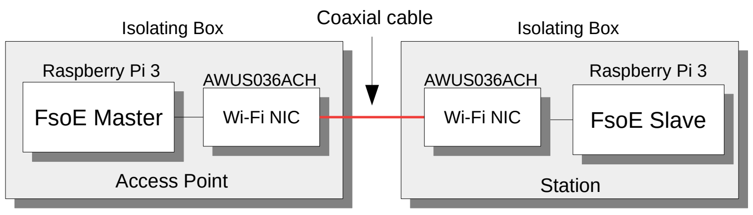


Figure 4.11: Experimental set-up

All the experiments have been carried out on Raspberry Pi Model 3B boards which run a general-purpose operating system Raspbian OS (Kernel version 5.4.83). Each device has been equipped with an external Alfa AWUS036ACH USB Wireless Network Interface Controller (WNIC) set to operate in the 2.4 GHz band with IEEE802.11g modulation standard and output power of 30 dBm. Moreover, the rate adaptation features have been disabled, thus using a fixed 54 MBps rate. Tests have been conducted in an industrial area, whereby it is necessary to minimize as far as possible the influence of external factors, for example, other WiFi stations or background noise. For this reason, the WNICs antennas have been connected via a coaxial cable, thus simulating an ideal transmission medium. To further increase the robustness to external noise, the Raspberry Pis and the WNICs have been embedded into separate shielding boxes, and finally, a variable RF attenuator, with an attenuation range of 50-110 dBm, has been inserted in the coaxial transmission line to properly control the attenuation of the real transmission medium.
The FSoE Master and FSoE Slave have been implemented on two different Raspberry Pi boards, respectively configured as Wi-Fi Access Point (AP) and Station (STA). The communication between the Master and the Slave is hence managed without intermediate devices. The simplest FSoE SPDU (7 bytes) has been encapsulated into a UDP frame, to carry safety data. It is worth noting that the FSoE stack runs on the Raspberry Pi as normal non-prioritized processes. Moreover, SPDUs are sent continuously in a non-scheduled way. In particular, after the reception of the safety frame SF_i from the master, the slave answers with the frame AF_i. Then, the master sends a new safety frame SF_{i+1} immediately after the reception of the answer AF_i from the slave. In each device, a watchdog of $250\mu s$ monitors the FSoE connection cycle to detect possible delays on the network. If a device does not receive any answer from the communication partner within the specified timeout, the frame is marked as lost and the FSoE connection is

re-initialized.
Several experimental sessions have been conducted, to test the medium with different attenuation values, that have been varied in 1dB steps. Correspondingly, for each measurement session, more than 50000 unique values of the polling time t_P have been acquired.
For the calibration of the polling time model, the probability densities obtained from the experimental measurements were used. Each of these was obtained for different channel attenuation values. Using the Inverse Transform Sampling method, the delays to be used in the transmission are sampled from the experimental densities according to the received signal strength. In practice, when the Master issues the transmission of an SPDU, the simulator calculates what will be the channel attenuation and therefore the power of the received signal. Based on this, the simulator chooses the density from which to sample, and schedules the response message from the slave after the delay generated by the sampling. The outcome of the polling time calibration is shown in Figure 4.12 while a more detailed statistic is reported in Table 4.6. As can be seen in Figure 4.12 the trend of the mean, minimum and maximum values of the polling time is rather similar for both the experimental and simulated sessions.

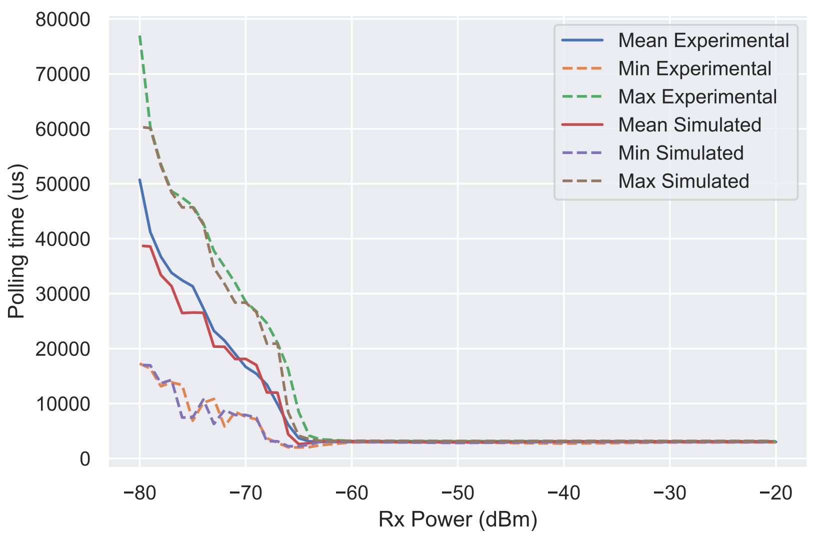


Figure 4.12: Comparison of mean, maximum and minimum polling time on the experimental and simulated setup

Indeed, for high reception powers, the trends are practically identical. For powers lower than -60 dBm the polling time values follow the same trend but with a definitely higher error. This observation is also confirmed by Figure 4.13 which shows the Mean

Table 4.6: Statistics of the polling time and PER

rxPower (dBm)	Polling time (µs)				PER	MSE on mean (%)
	Mean	Std	Min	Max		
	Experimental					
-80.0	50720.27	11510.14	17271.40	76988.70	0.030	-
-76.0	32437.22	6756.42	13383.80	47484.70	0	-
-73.0	23225.14	6050.65	10836.40	37790.80	0	-
-70.0	16663.57	4124.12	7597.51	28525.20	0	-
-67.0	9870.57	3460.87	2855.14	21021.10	0	-
-64.0	3116.34	220.08	2076.29	4144.56	0	-
-50.0	3010.49	23.01	2931.70	3139.15	0	-
-20.0	3010.20	21.32	2930.30	3130.82	0	-
	Simulated					
-80.0	41473.87	8602.50	23940.54	59851.54	0.027	18.23
-76.0	31377.75	6506.41	17041.85	45395.85	0	3.26
-73.0	21427.64	4985.96	13059.43	34340.43	0	7.73
-70.0	16684.58	3954.34	11635.10	27807.10	0	0.12
-67.0	9877.29	3260.08	4049.86	20378.86	0	0.06
-64.0	3106.29	82.38	3035.66	3535.66	0	0.32
-50	3084.09	21.70	3014.20	3173.20	0	2.44
-20	3085.52	19.16	3048.02	3170.02	0	2.50

Square Error calculated on the average polling time.

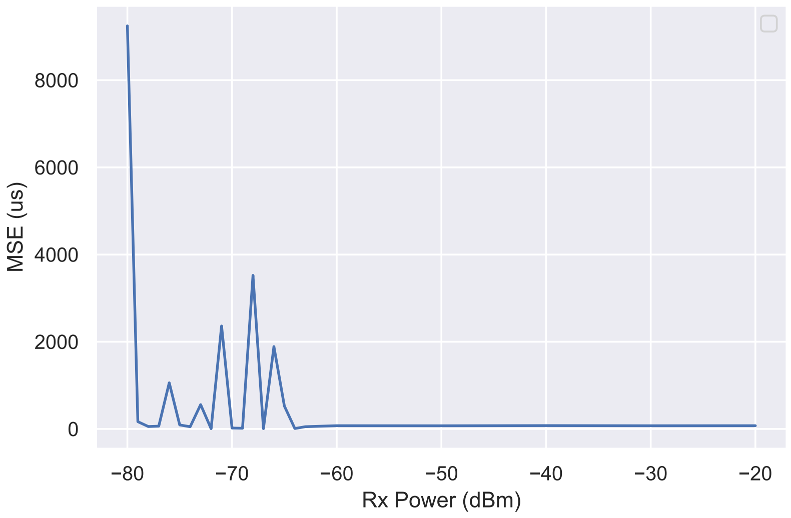


Figure 4.13: MSE

As can be seen, the error is almost zero for relatively high receiving powers, while it tends to increase as the intensity of the received signal decreases. This phenomenon is due to the model of path loss used in the simulator which is the *LogNormalShadowing*. The path loss model not only takes care of simulating the attenuation of the signal and calculating the probability that the signal can be received, but also it adds a certain degree of variability to the signal reception time to simulate the effects of attenuation. Therefore, the uncertainties introduced by the path loss model overlap with the delays introduced by the polling time model causing it to deviate slightly from the experimental trend. The solution to this problem will require the implementation of a completely custom path loss model, which will be left for future work. In the current state of the simulator, the path loss model has been calibrated to minimize these superimposition effects.

In general, the accuracy of the calibration can be confirmed by the MSE which, except for some isolated cases, remains less than 4%. This is also confirmed by Figure 4.14 which shows the comparison of the probability density function of the polling time. As it can be seen, they are definitely very similar.

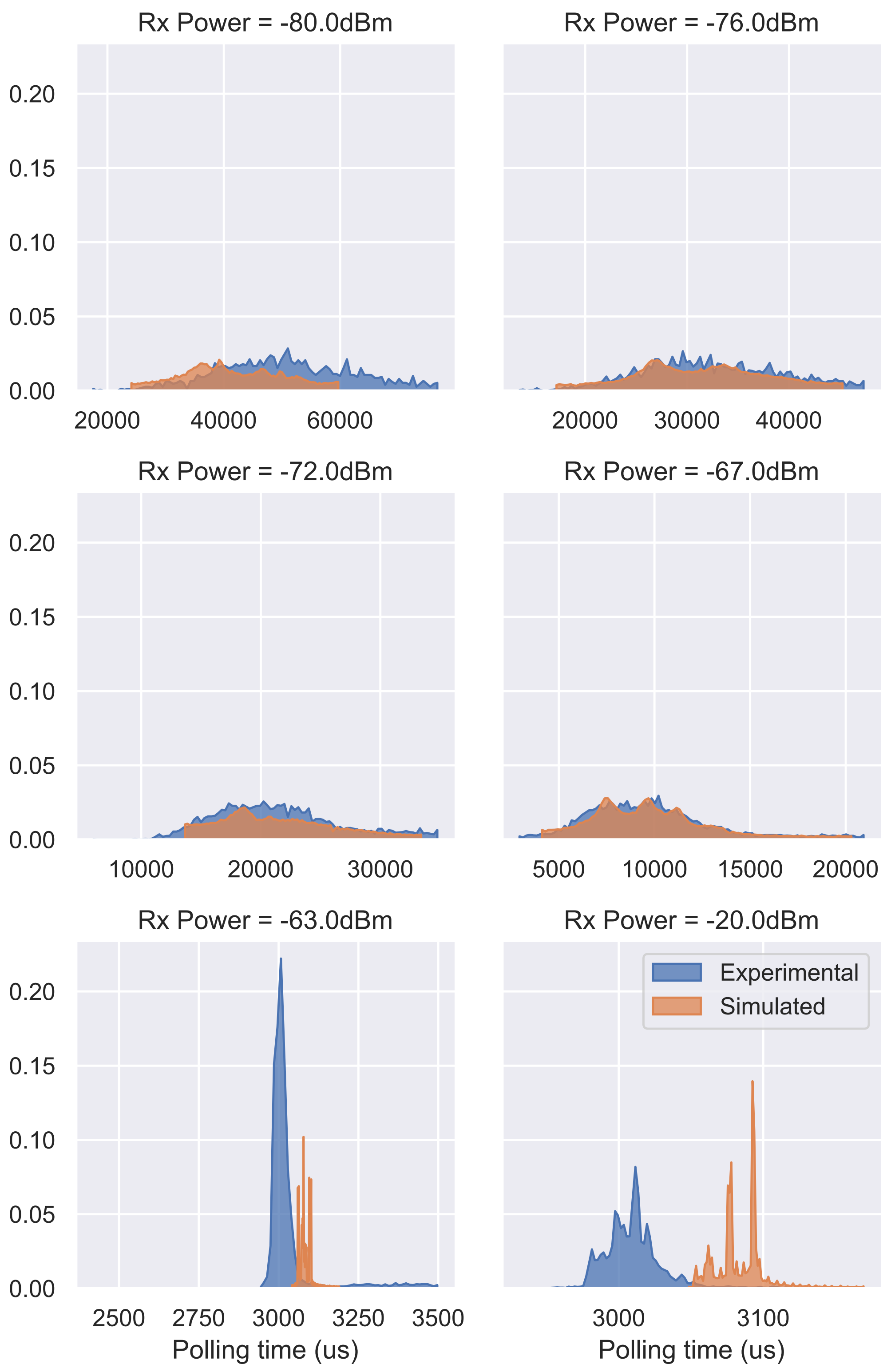


Figure 4.14: Comparison of the probability density function of the polling time in the experimental and simulated setup

4.5.2 Simulation with multiple slaves

The simulation of a realistic multi-node network needs particular care and requires the execution of several tests. In this Section, the outcomes of some simulation sessions carried out towards the assessment of the proposed simulation model were analyzed. In particular, it refers to the prototype network, described in Fig. Figure 4.15, which is composed of one FSoE master and five FSoE slaves. The position of the nodes with respect to each other has been carefully selected to reproduce and test three different communication scenarios. Firstly, aiming at analyzing the protocol behavior in absence of mutual interference, Slaves 1 and 2 are placed relatively distant from each other, thus reproducing an ideal situation. On the contrary, slaves 3 and 4 have been placed close together, to study how possible interference can affect the polling time and the PER. Finally, slave 5 has been placed far away from the master, to analyze the impact of a great attenuation, i.e. a relatively low SNR, on the polling time and the PER.

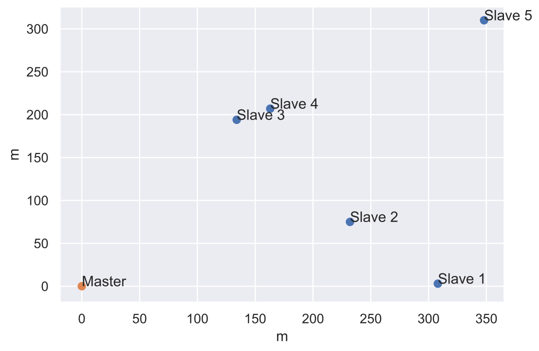


Figure 4.15: Positions of the nodes

Simulation statistics of both the polling time and exchanged packets for each node are reported in Table 4.7.

Table 4.7: Statistics of the polling time in simulation with multiple slaves

	Polling time (µs)		**Packets**			
	Mean	Std	Sent	Acknowledged	Lost	PER
Slave						
1	33932.20	6581.77	71997	71997	0	0.000000
2	27618.30	6540.25	71997	71997	0	0.000000
3	27390.94	6506.25	71999	71999	0	0.000000
4	31431.96	6545.19	72000	71998	2	0.000028
5	105864.38	52708.85	71998	52683	19315	0.268271

Comparing the behavior of Slaves 1 and 2, it is possible to underline that the latter introduces a slightly lower polling time, while both do not experience any packet loss. This is reasonable as they do not have other nodes nearby, but the distance between Slave 1 and the Master is higher than the one with the Slave 2. Conversely, Slaves 3 and 4 introduce mutual interference, as can be noted from both the polling time (t_p) and the Packet Error Rate (PER). Indeed, the polling times of Slaves 3 and 4 are quite similar to those of Slaves 1 and 2, although the latter ones have a greater distance from the master. Furthermore, Slave 4 also experiences some packet loss. Finally, Slave 5 introduces both higher t_p and PER, thus underlying the impact of the attenuation on the communication.

5 Conclusions

Functional safety networks are expected to be increasingly used in IIoT ecosystems, particularly over wireless media. In this respect, during the Ph.D. work, the challenges of designing wireless distributed functional safety systems, possibly real–time were addressed. In particular, there was a focus on those safety protocols described by the IEC 61784–3 International Standard, especially FSoE, and it was investigated their suitability for IIoT applications. The provided analysis, as well as the results of an extensive experimental session carried out on a prototype implementation of FSoE over WiFi, allowed to make some interesting considerations. First, the black channel principle can be successfully exploited to bring safety protocols over communication media different from those for which they were natively designed. Second, although the black channel approach in principle ensures feasibility, it is clear it might introduce limitations, particularly with respect to performance. This aspect derives from the undeniable fact that each protocol has been conceived for a specific network. As an example, referring to wired systems, functional safety protocols designed for Ethernet networks can be difficult to implement on communication systems that use short payloads such as Controller Area Network. Even more evidently, referring to OPC UA Safety, SPDUs may reach large sizes, thus, they need adequate MAC and physical layers to be transferred. Possible performance limitations have been clearly evidenced by the experimental assessment of FSoE over Wi-Fi Indeed, as seen, the behavior of FSoE is strictly related to the protocol stack

included in the black channel. Nevertheless, it has also been shown that there is a large room for improvement and future developments. Focusing on Wi-Fi, performance might be further improved with respect to both reliability and reaction time. Indeed, rate adaptation algorithms specifically designed for industrial WiFi applications, as well as suitable network protocol tuning, may lead to significant benefits in terms of reliability and timeliness. In addition, new forthcoming Wi-Fi versions, such as those based on the IEEE 802.11ax standard, promise considerable performance improvements. Moreover, the upcoming extension of TSN features to Wi-Fi and 5G systems must be remarked, indeed thanks to their built-in mechanisms to improve reliability and timeliness, they promise to improve considerable performances with the possibility of enforcing a rapid and extensive adoption of safety protocols in IIoT ecosystems.
Focusing, instead, on the transport layer, some important outcomes were obtained and some other considerations could be done. Indeed, while TCP seems to result the obvious choice from this point of view, the lack of broadcast addressing could make it difficult to use in some applications. In this sense, a caching layer that implements the retransmission and management of duplicate frames, being agnostic from other protocols and layers, was proposed. In the presented implementation, the layer was interposed between UDP and FSoE. To evaluate the effectiveness of the protocols analyzed, the number of lost packets was measured in a real implemented setup. The polling time was monitored to understand the impact of the different protocols on it. Both TCP and caching layer implementations have outperformed UDP, drastically lowering packet loss. The impact on TCP polling time was more evident, while the introduction of the caching layer is practically irrelevant. The results obtained showed the importance of the transport layer, and more generally of the entire stack. These results are a further step towards the implementation of safety critical wireless systems, as it has been possible to achieve levels of reliability very similar to those of a wired network. Despite all these considerations, the risk of implementing a stack that is not able to achieve the required safety level still exists. In this context, field testing of distributed functional safety systems is not always a feasible operation and for these reasons, the implementation and the calibration of a functional safety over wireless simulation model in OMNeT++ were proposed. The calibration was done using measurements conducted in an experimental setup specifically designed for this purpose. The error channel model and polling time have been calibrated for various transmission channel conditions. The comparison with the experimental measurements and the tests conducted on a simulation with multiple nodes revealed a good calibration of the simulator. However, further extensive tests are required for precise and complete validation.

A
Appendix

A.1 Adaptive Smart Sensing in Resource-Constrained Edge Computing: Setup and problem Formulation

A.1.1 System model

Dynamical System. The process of interest is described by a time-varying discrete-time linear dynamical system,

$$x_{k+1} = A_k x_k + w_k \tag{A.1}$$

where $x_k \in \mathbb{R}^n$ collects the to-be-estimated variables (state) of the system, $A_k \in \mathbb{R}^{n\times n}$ is the state matrix, and white noise $w_k \sim \mathcal{N}(0, W_k)$ captures model uncertainty. Time-varying linear models are widely used in control applications, in virtue of their simplicity but also powerful expressiveness (Medeiros, Park, and Kak (2008); Yang, Zhang, Zheng, and Qian (2020); Jeon and Eun (2019); Radisavljevic-Gajic, Park, and Chasaki (2018); Devos, Kirchner, Croes, Desmet, and Naets (2021)). For example, a standard approach in control of dynamical systems modeled through nonlinear differential equations is to approximate the original model as a parameter- or time-varying linear system, for which efficient control and optimization techniques are known (Devos et al. (2021); Gros, Zanon, Quirynen, Bemporad, and Diehl (2020); Tsai and Gu (2014)).

In view of wireless transmission of sensor sampling, we assume a discrete-time dynamics with time step T, where subscript $k \in \mathbb{N}$ indicates the kth time instant kT. Without loss of generality, we set the first instant $k_0 = 1$. The sampling time T represents a suitable time scale for the global monitoring and, possibly, decision-making task at hand. For example, typical values of T are one or two seconds for trajectory planning of ground robots, while higher frequency is required for drones performing fast pursuit or self-driving applications.

Smart Sensors. The process modeled by (A.1) is measured by N smart sensors (or simply sensors) gathered in the set $\mathcal{V} = \{1, \dots, N\}$, which output a noisy version of the state x_k,

$$y_k^{(i)} = x_k + v_k^{(i)}, \qquad v_k^{(i)} \sim \mathcal{N}(0, \mathbb{V}i, k), \tag{A.2}$$

where $y_k^{(i)}$ is the measurement collected by the ith sensor at time k, for any $i \in \mathcal{V}$, and $v_k^{(i)}$ is measurement noise.

Sensors can either communicate raw measurements or process collected information locally before transmission. Such processing may consist in compression, filtering, or more complex tasks such as feature extraction from visual data or 3D point clouds. Because of limited hardware resources, sensors face a *latency-accuracy trade-off*: raw measurements are less accurate, but local data processing introduces extra computational delay. In particular, uncertainty about the true dynamics, modeled in (A.1) through the noise term w_k, progressively makes outdated measurements less informative about the current state of the system, so that high accuracy alone might not pay off in real-time monitoring.

For example, consider a car that is approximately moving at a constant speed, where w_k captures unmodeled movements such as sudden accelerations: as time goes by, knowledge of the current position of the car through its nominal model (constant speed) becomes more and more imprecise because of all unmodeled terms hidden by w_k (*e.g.,* accelerations), that progressively deviate the car from its nominal trajectory. In this case, a sensor may prefer to sample the system (*e.g.,* acquire images of the road) more often, rather than spending time to obtain precise, but outdated, position measurements.

We formalize the latency-accuracy trade-off as follows.

Assumption 1 (Sensing modes)**.** Each sensor $i \in \mathcal{V}$ can be in *raw*, *processing*, or *sleep* mode.

Raw mode: measurements are generated after delay $\tau_{i,\text{raw}}$ with noise covariance $\mathbb{V}i, k \equiv V_{i,\text{raw}}$.

Processing mode: measurements are generated after *processing delay* $\tau_{i,\text{proc}}$ with noise covariance $\mathbb{V}i, k \equiv V_{i,\text{proc}}$.

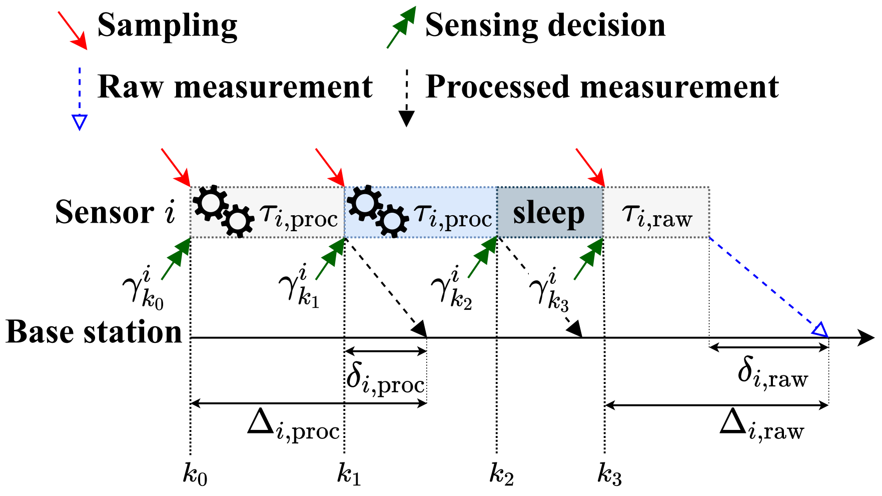


Figure A.1: Data collection and transmission. Computation at the ith sensor is ruled by sensing policy π_i. Here, sensing decisions $\{\gamma^i_{k_j}\}_{j=0}^3 = \{\text{p}, \text{p}, \text{s}, \text{r}\}$ are shown and (A.3b), (A.4b) read $\mathcal{K}_i[0] = s_i^0(k_0) = k_0$, $\mathcal{K}_i[1] = s_i(k_0) = k_1$, $\mathcal{K}_i[2] = s_i(k_1) = k_3$. Measurements are received after delays induced by local computation (rectangular blocks) and communication (dashed arrows). For example, under $\gamma^i_{k_0} = \text{p}$, the sample acquired at time k_0 is firstly processed (with processing delay $\tau_{i,\text{proc}}$), then transmitted at time $k_1 = k_0 + \tau_{i,\text{proc}}$ (with communication delay $\delta_{i,\text{proc}}$), and finally received at the base station at time $k_1 + \delta_{i,\text{proc}} = k_0 + \Delta_{i,\text{proc}}$ (with delay at reception $\Delta_{i,\text{proc}}$).

Sleep mode: the sensor is temporary set idle (*asleep*): neither data sampling nor transmission occur during this mode.

Assumption 2 (Latency-accuracy trade-off)**.** For each sensor $i \in \mathcal{V}$, it holds $\tau_{i,\text{proc}} > \tau_{i,\text{raw}}$ and $V_{i,\text{raw}} \succ V_{i,\text{proc}}$.[1]

Similarly to Ballotta et al. (2020), Assumption 2 models high accuracy as long computation (Figure A.1) and "small" covariance (intensity) of measurement noise, *e.g.,* raw distance measurements may have uncertainty of 1m^2 while processed ones of 0.1m^2.

Next, we define how local operations are ruled overtime.

Definition A.1.1 (Sensing policy)**.** A *sensing policy* for the ith sensor is a sequence of categorical decisions $\pi_i \doteq \{\gamma^i_k\}_{k \geq k_0}$. If $\gamma^i_k = \text{r}$, measurement $y^{(i)}_k$ is transmitted raw; if $\gamma^i_k = \text{p}$, $y^{(i)}_k$ is processed; if $\gamma^i_k = \text{s}$, no measurement is acquired at time k.

According to Theorem A.1.1, different sensing modes can be alternated online. However, because of constrained resources, a sensor cannot acquire measurements

[1]Covariance matrices are ordered according to Löwner order of positive semidefinite matrices. Even though this is a partial order, we require it in our model to express the latency-accuracy trade-off unambiguously.

arbitrarily often. The actual sampling frequency is determined as formalized next. **Wireless channel.** All sensors transmit their data to a common base station using a shared wireless channel. The latter induces *communication latency* that may further delay transmitted updates. We let $\delta_{i,\text{raw}}$ and $\delta_{i,\text{proc}}$ denote the communication delay of raw and processed data transmitted by the ith agent, respectively. In general, $\delta_{i,\text{raw}}$ and $\delta_{i,\text{proc}}$ might differ, depending on possible compression performed by data processing. In case $\delta_{i,\text{raw}} = \delta_{i,\text{proc}}$, we denote both communication delays by δ_i. Finally, the total delay experienced by transmitted data from sampling to reception at the base station (*delay at reception*) is given by $\Delta_{i,\text{raw}} = \tau_{i,\text{raw}} + \delta_{i,\text{raw}}$ for raw and $\Delta_{i,\text{proc}} = \tau_{i,\text{proc}} + \delta_{i,\text{proc}}$ for processed data. Delayed data sampling, processing, and transmission are illustrated in Figure A.1.

Assumption 3 (Sampling frequency)**.** Assume that the ith sensor acquires a sample at time k under either raw ($\gamma_k^i = \text{r}$) or processing ($\gamma_k^i = \text{p}$) mode. Let time k' be defined as

$$k' \doteq \begin{cases} k + \tau_{i,\text{raw}} & \text{if } \gamma_k^i = \text{r} \\ k + \tau_{i,\text{proc}} & \text{if } \gamma_k^i = \text{p}. \end{cases} \tag{A.3a}$$

Then, the next sampling (under any mode) occurs at time $s_i(k)$,

$$s_i(k) \doteq \min_{h\in\mathbb{N}} \left\{ h \geq k' : \gamma_h^i \neq \text{s} \right\}. \tag{A.3b}$$

Finally, the sequence of all sampling instants $\mathcal{K}_i$ is given by

$$\mathcal{K}_i = \left\{ s_i^l(k_0) \right\}_{l\geq 0}, \tag{A.4a}$$

where consecutive sampling times are defined by the recursion

$$\begin{aligned} s_i^{l+1}(k) &= s_i\left(s_i^l(k)\right) \\ s_i^0(k_0) &\doteq \min_{h\in\mathbb{N}} \left\{ h \geq k_0 : \gamma_h^i \neq \text{s} \right\}, \end{aligned} \tag{A.4b}$$

and $\mathcal{K}_i[l]$ denotes the lth element of the sequence, with $l \in \mathbb{N}$.

In words, Assumption 3 states that sensors can acquire a new sample only after the previous measurement is transmitted. This is a realistic assumption if agents have limited storage resources (Mildenhall, Srinivasan, Tancik, Barron, Ramamoorthi, and Ng (2020)). The effect of a sensing policy on sampling and local data processing is illustrated in Figure A.1.

Remark 2 (Multiple processing modes). Smart sensors may have multiple options to process data. For example, a robot equipped with a camera might choose among several geometric inference algorithms to trade latency for accuracy (in general, anytime algorithms Zilberstein (1996)). While we stick to a single processing mode for the sake of simplicity and exposition, our framework can be readily extended in that respect.

Base Station. Data are transmitted to a base station in charge of estimating the state of the system x_k in real-time. Such estimation enables remote global monitoring and decision-making, *e.g.,* coordinated trajectory tracking or exploration. Let $\hat{x}_k$ denote the real-time estimate of x_k. In view of the sequential nature of centralized data processing, when limited computational resources are available at the base station, the real-time estimate of x_k is computed in ϕ_k time (*fusion delay*), needed to process sensory data used in the update by Ballotta et al. (2020). For example, consider Figure A.2: from time k_1 through k_4, new data are received at the base station (green dashed arrows). If the estimation subroutine starts at time k_4, it will take ϕ_{k_5} to process all newly received sensory data (possibly, also old ones if some data arrive out of sequence), and hence the next updated state estimate, $\hat{x}_{k_5}$, will be available at time $k_5 = k_4 + \phi_{k_5}$. Clearly, fusion delays induce longer open loops that further degrade the quality of the computed estimates (similarly to what discussed about local sensor processing) and motivate sleep mode to reduce the incoming stream of sensory data and improve overall performance (Ballotta et al. (2020)).

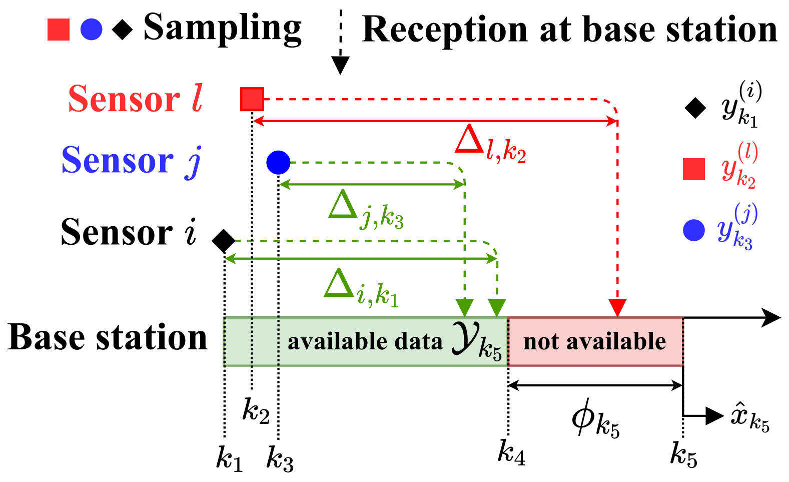


Figure A.2: Data processing at the base station. Resource-constrained centralized processing introduces *fusion delay* ϕ_{k_5} to estimate x_{k_5}. Measurements $y_{k_1}^{(i)}$ and $y_{k_3}^{(j)}$ are received before computation starts at time $k_4 = k_5 - \phi_{k_5}$ and are used to compute $\hat{x}_{k_5}$, *i.e.,* $y_{k_1}^{(i)}, y_{k_3}^{(j)} \in \mathcal{Y}_{k_5}$, while $y_{k_2}^{(l)}$ is received after time k_4 and can be used only in following estimations, *i.e.,* $y_{k_2}^{(l)} \notin \mathcal{Y}_{k_5}$.

Assumption 4 (Available sensory data)**.** In view of Assumptions 1, 3, all sensory data available at the base station and used to compute $\hat{x}_k$ at time k are

$$\mathcal{Y}_k \doteq \bigcup_{i\in\mathcal{V}} \bigcup_{l\in\mathbb{N}} \left\{ \left(y^{(i)}_{\mathcal{K}_i[l]}, \forall i, \mathcal{K}_i[l] \right) : \mathcal{K}_i[l] + \Delta_{i,\mathcal{K}_i[l]} + \phi_k \leq k \right\}$$
$$\Delta_{i,\mathcal{K}_i[l]} \doteq \begin{cases} \Delta_{i,\mathrm{raw}} & \text{if } \gamma^i_{\mathcal{K}_i[l]} = \mathrm{r} \\ \Delta_{i,\mathrm{proc}} & \text{if } \gamma^i_{\mathcal{K}_i[l]} = \mathrm{p}, \end{cases} \tag{A.5}$$

where the lth measurement from the ith sensor $y^{(i)}_{\mathcal{K}_i[l]}$ is sampled at time $\mathcal{K}_i[l]$ and received after overall delay $\Delta_{i,\mathcal{K}_i[l]}$, and ϕ_k is the time needed to compute $\hat{x}_k$ at the base station.

According to Assumption 4, a measurement $y^{(i)}_h$ can be used to compute the estimate of x_k in real time if it is successfully delivered to the base station (with delay at reception $\Delta_{i,h}$) before or at time $k - \phi_k$, where ϕ_k is the amount of time needed to compute $\hat{x}_k$. Data processing at the base station with limited resources and data availability is depicted in Figure A.2.

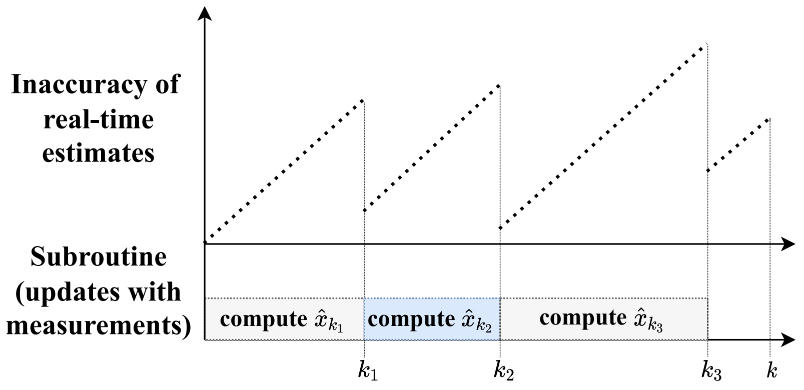


Figure A.3: Real-time estimation at the base station. The state estimate is updated at each point in time (top). Because of limited resources at the base station, open-loop updates are performed whenever fresh sensory data are being processed (bottom), causing performance to degrade overtime through uncertainty in the nominal dynamics (A.1). As soon as the data processing subroutine outputs an updated estimate with new measurements (*e.g.*, $\hat{x}_{k_1}$), the estimation inaccuracy is reduced (*e.g.*, drop at time k_1). **The top plot is qualitative**: the estimate quality does not degrade linearly, in general.

Remark 3 (Actual real-time estimation)*.* Based on the above discussion, the base station cannot provide updated estimates between time k and $k+\phi_k$. In a real system, a real-time state estimate must always be available for effective monitoring. We assume that two parallel jobs are executed. A support subroutine processes received measurements and computes a state estimate at time k in ϕ_k time (cf. Figure A.2). The real-time estimation routine computes one-step-ahead open-loop updates at each point in time according to the nominal dynamics (A.1) (thus degrading estimate quality), and resets whenever the support subroutine outputs an updated estimate with new received measurements (with higher estimate quality).[2] A schematic representation is shown in Figure A.3. Importantly, estimate degradation (top plot in Figure A.3) is not due to lack of new measurements (like in Age of Information literature), but is caused by constrained resources that induce a computational bottleneck in the support subroutine (bottom plot in Figure A.3).

A.1.2 Problem Statement

The trade-offs introduced in the previous section call for a challenging sensing design at network level. In particular, all possible choices of local sensor processing (we address a specific choice for all sensors as a *sensing configuration*) affect global performance in a complex manner, whereby it is unclear which sensors should transmit raw measurements, with poor accuracy and possibly long communication delays, and which should refine their samples locally to produce high-quality measurements. In fact, the authors Ballotta et al. (2020) show that the optimal configuration when considering steady-state performance is nontrivial. Also, the optimal sensing configuration is time-varying, in general. Thus, sensing policies $\pi_i, i \in \mathcal{V}$, have to be suitably designed to maximize the overall network performance.

The state x_k is estimated via Kalman predictor, which is the optimal state observer for linear systems driven by Gaussian noise. It can be readily shown, *e.g.,* via state augmentation, that the Kalman predictor is optimal even with delayed or dropped measurements, whereby it suffices to ignore updates associated with missing data. Out-of-sequence arrivals can be handled by recomputing all predictor steps since the latest arrived measurement has been acquired, or by more sophisticated techniques (Tou and Zhang (2016); Gopalakrishnan, Kaisare, and Narasimhan (2011)).

Let $\tilde{x}_k \doteq x_k - \hat{x}_k$ the estimation error of Kalman predictor at time k, and let $P_k \doteq \mathrm{Var}(\tilde{x}_k)$ its covariance matrix. We formulate the sensing design as an optimal estimation problem.

Problem 1 (Sensing Design for Processing Network)**.** Given system (A.1)–(A.2) and Assumptions 1–4, find the optimal sensing policies π_i, $i \in \mathcal{V}$, that minimize the finite-

[2]We assume that one-step-ahead open-loop steps are computationally cheap.

horizon time-averaged estimation error variance,

$$\begin{aligned}
&\underset{\pi_i\in\Pi_i, i\in\mathcal{V}}{\arg\min} && \frac{1}{K}\sum_{k=k_0}^{K}\mathrm{Tr}\left(P_k^\pi\right) && \text{(A.6a)}\\
&\text{s.t.} && P_k^\pi = f_{\text{Kalman}}\left(\mathcal{Y}_k^\pi\right), && \text{(A.6b)}\\
& && P_{k_0}^\pi = P_0, && \text{(A.6c)}
\end{aligned}$$

where the Kalman predictor $f_{\text{Kalman}}(\cdot)$ computes at time k state estimate $\hat{x}_k^\pi$ and error covariance matrix P_k^π using data $\mathcal{Y}_k^\pi$ available at the base station according to $\pi \doteq \{\pi_i\}_{i\in\mathcal{V}}$, and Π_i gathers all causal sensing policies of the ith sensor.

Remark 4 (Difference with standard sensor selection). What we denote by sleep mode intrinsically represents an online sensor selection. We identify two key elements that make our framework fundamentally different from standard sensor selection in the literature. First, similarly to Ballotta et al. (2020), the use of both "active" modes (raw or processing) and sleep mode targets *optimal performance*, while sensors are classically selected to meet some budget constraints, under the conventional wisdom that the more are selected, the better performance. In contrast, as discussed above, in our framework selection emerges *naturally* as a need to optimize performance, in view of the computational bottleneck at the base station that can increase the objective cost in (A.6). Secondly, rather than a *static, a priori* selection, we allow for *dynamical switching* to and from sleep mode, which enables a much richer design and performance improvement.

A.1.3 System model: a centralized implementation

Problem 1 is combinatorial and does not scale with the size of the system. This raises a computational challenge in finding efficient sensing policies, because the search space may easily explode. On the one hand, the total number of sensing configurations does not scale with the amount of sensors, *e.g.,* 10 sensors yield $2^{10} = 1024$ possible sensing configurations at each sampling instant. On the other hand, Problem 1 also requires to design the policy for each sensor, which is a combinatorial problem by itself that scales exponentially with the time horizon K. Furthermore, each sensing policy π_i not only affects data delay and accuracy but also determines the sampling sequence $\mathcal{K}_i$ for the ith sensor (cf. (A.3)–A.4), augmenting the search space to all possible sampling sequences.

To partially ease the intractability of the problem, and motivated by practical applications, we restrict the domain of potential policies to reduce problem complexity while still enabling useful insights. First, we look at the simple but relevant scenario with a homogeneous network and motivate the design of a centralized policy in subsubsection A.1.3.1. We then go back to the general scenario with a heterogeneous network and formulate a simplified version of Problem 1 in subsubsection A.1.3.2.

Remark 5 (Complexity with multiple processing modes). In the general case where the ith sensor has C_i processing modes, there are $\prod_{i\in\mathcal{V}}(C_i+2)$ sensing configurations in total.

A.1.3.1 Homogeneous Network

Sensor Model. In this scenario, all smart sensors have equal measurement noise distributions,

$$y_k^{(i)} = x_k + v_k^{(i)}, \qquad v_k^{(i)} \sim \mathcal{N}(0, \mathbb{V}k), \tag{A.7}$$

$V_k = V_{raw}$ or $V_k = V_{proc}$ for raw and processed data, respectively. Also, all sensors feature identical computational and transmission resources, given by delays τ_{raw}, δ_{raw} for raw measurements and τ_{proc}, δ_{proc} for processed measurements, respectively (δ in case of no compression). This *homogeneous network* models the special but relevant case where sensors are interchangeable. This happens for example with sensor networks measuring temperature in plants or chemical concentrations in reactors. Also, this model captures smart sensors collecting high-level environmental information, such as UAVs tracking the position of a body moving in space.

Centralized Policy. In this case, it is sufficient to decide *how many*, rather than *which*, sensors follow a certain mode. Accordingly, we focus on the design of a centralized policy that commands all sensors with no distinctions among them.

Definition A.1.2 (Homogeneous sensing policy)**.** A *homogeneous sensing policy* is a sequence of categorical decisions $\pi_{hom} = \{\gamma_\ell^{hom}\}_{\ell=1}^{L}$. Each decision $\gamma_\ell^{hom} = (N - n_{\mathrm{s}}, n_{\mathrm{p}})$ is taken at time $k^{(\ell)}$ such that n_{s} sensors are in sleep mode and n_{p} out of the other $N - n_{\mathrm{s}}$ sensors are in processing mode between times $k^{(\ell)}$ and $k^{(\ell+1)}$, with $0 \le n_{\mathrm{s}} \le N$ and $0 \le n_{\mathrm{p}} \le N - n_{\mathrm{s}}$. Without loss of generality, we set $k^{(1)} = k_0$, $k^{(L)} \le K$.

In words, the base station decides a configuration for all sensors at predefined time instants, which is both practical for applications and convenient to reduce the complexity of the problem. However, decisions may be taken at any time, as long as these are consistent with sensor computational delays (*e.g.,* to guarantee that one sample is collected for each decision).

With a slight abuse of notation, to address the mode of a specific sensor that is following the homogeneous decision γ_ℓ^{hom}, we write $\gamma_\ell^i = \mathrm{m} \in \{\mathrm{r}, \mathrm{p}, \mathrm{s}\}$ meaning that the

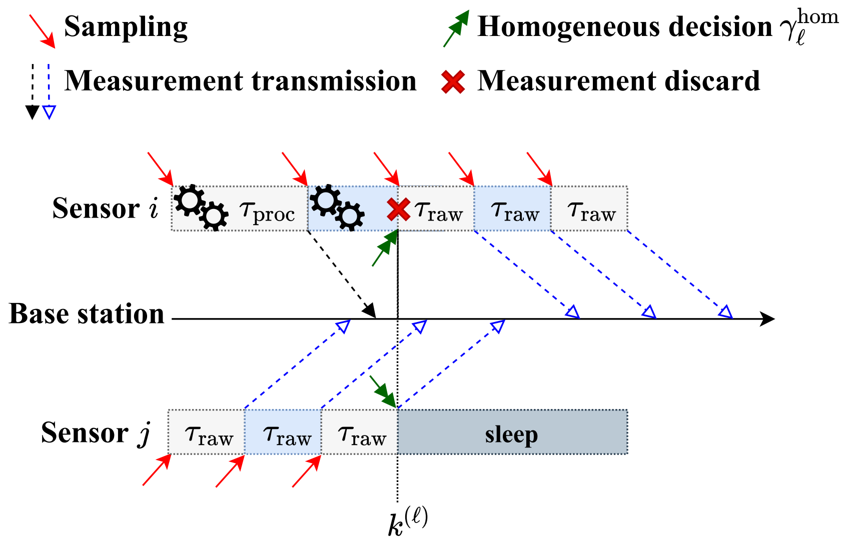


Figure A.4: Homogeneous sensing policy. Sampling and data processing at identical sensors are ruled by policy π_{hom}. Decision γ_ℓ^{hom} is communicated at time $k^{(l)}$ and realized at individual sensors as $\gamma_\ell^i = \text{r}$ and $\gamma_\ell^j = \text{s}$. Concurrently, the ith sensor disregards its current processed measurement (red cross) and switches to raw mode, acquiring a new sample at time $k^{(l)}$

ith sensor is in raw, processing, or sleep mode, respectively. We stress that in this context γ_ℓ^i does not represent a decision of a *single-sensor* sensing policy π_i (see Theorem A.1.1), but all decisions are centralized and γ_ℓ^i denotes the mode that *the base station commands* the ith sensor to obey through decision γ_ℓ^{hom}.

Centralized decisions are communicated regardless of current sensing status. Following common practice in real-time control (Greco, Fontanelli, and Bicchi (2011); Pavlidakis, Mavridis, Chrysos, and Bilas (2020); Lee and Lee (2020); Fedullo et al. (2022); Yu, Jia, Liu, and Ma (2020)), we assume what follows.

Assumption 5 (Sampling frequency with homogeneous sensing policy)**.** Decision γ_ℓ^{hom} switches mode of the minimum amount of sensors possible. If the ith sensor switches mode, the measurement currently being acquired or processed (if any) is immediately discarded. If the new commanded mode is either raw or processing, a new sample is acquired according to such new mode right after the decision γ_ℓ^{hom} is communicated.

Formally, given measurement $y_k^{(i)}$ sampled at time $k < k^{(\ell)}$ obeying decision $\gamma_{\ell-1}^{hom}$, the sampling dynamics (A.3b) becomes

$$s_i^{\text{hom}}(k) \doteq \begin{cases} k' & \text{if } k^{(\ell)} \geq k' \\ k^{(\bar{\ell})} & \text{otherwise,} \end{cases} \tag{A.8a}$$

$$\bar{\ell} \doteq \min_{\ell' \in \{1,\dots,L\}} \left\{ \ell' : \ell' \geq \ell \wedge \gamma_{\ell'}^{i} \neq \mathrm{s} \right\}. \tag{A.8b}$$

Further, $y_k^{(i)}$ is discarded (not transmitted) if $s_i^{\text{hom}}(k) \neq k'$.

The new sampling mechanism is depicted in Figure A.4. According to Assumption 5, a measurement is not transmitted to the base station if it is not ready when a concurrent decision is communicated. In Figure A.4 the ith sensor discards a measurement whose processing is not completed at time $k^{(\ell)}$, when a new decision switches its mode. Formally, a sensor disregards raw (resp. processed) measurements sampled at time $\bar{k} < k^{(\ell)}$ such that $\bar{k} + \tau_{raw} > k^{(l)}$ ($\bar{k} + \tau_{proc} > k^{(l)}$), *i.e.,* their acquisition ends after a *different* mode is imposed by decision γ_ℓ^{hom} (cf. (A.8a)). We denote by $\mathcal{Y}_k{}^{\pi_{hom}}$ all available data at the base station at time k according to (A.8) and such discard mechanism imposed by policy π_{hom}, that excludes some data included in $\mathcal{Y}_k$ (cf. (A.5)).

A.1.3.2 Heterogeneous Network

We now return to the original model (A.2) with heterogeneous sensors. Without loss of generality, assume that the sensor set $\mathcal{V}$ is partitioned into M disjoint subsets $\mathcal{V}_1, \dots, \mathcal{V}_M$, where subset $\mathcal{V}_m$, $m = 1, \dots, M$, is composed of homogeneous sensors of the mth class. From what discussed in the previous section, it is sufficient to specify how many sensors follow a certain mode within each subset $\mathcal{V}_m$. Hence, we finally narrow down the domain of all possible policies according to the next definition.

Definition A.1.3 (Network sensing policy)**.** A *network sensing policy* is a collection $\pi_{\text{net}} \doteq \{\pi_{\text{hom},m}\}_{m=1}^{M}$, where each homogeneous sensing policy $\pi_{\text{hom},m}$ is associated with homogeneous sensor subset $\mathcal{V}_m$, and all homogeneous decisions $\{\gamma_\ell^{\text{hom},m}\}_{m=1}^{M}$ are communicated together at time $k^{(\ell)}$.

In Theorem A.1.3, decision times are fixed like in the homogeneous case, so that decisions are communicated to all sensors at once. At time $k^{(\ell)}$, homogeneous decision $\gamma_\ell^{\text{hom},m}$ involves sensors in $\mathcal{V}_m$, and the overall sensing configuration is given by the ensemble of such decisions. All data available at the base station at time k are collected in $\mathcal{Y}_k^{\pi_{\text{net}}} \doteq \{\mathcal{Y}_k^{\pi_{\text{hom},m}}\}_{m=1}^{M}$.

Finally, we get the following simplified problem formulation.

Problem 2 (Centralized Sensing Design for Processing Network)**.** Given system (A.1), (A.2) with Assumptions 1–5, find the optimal network sensing policy π_{net} that minimizes the finite-horizon time-averaged estimation error variance,

$$\begin{aligned} \underset{\pi_{\text{net}} \in \Pi_{\text{net}}}{\arg\min} \quad & \frac{1}{K} \sum_{k=k_0}^{K} \operatorname{Tr}\left(P_k^{\pi_{\text{net}}}\right) && \text{(A.9a)} \\ \text{s.t.} \quad & P_k^{\pi_{\text{net}}} = f_{\text{Kalman}}\left(\mathcal{Y}_k^{\pi_{\text{net}}}\right), && \text{(A.9b)} \\ & P_{k_0}^{\pi_{\text{net}}} = P_0, && \text{(A.9c)} \end{aligned}$$

where the Kalman predictor $f_{\text{Kalman}}(\cdot)$ computes at time k state estimate $\hat{x}_k^{\pi_{\text{net}}}$ and error covariance matrix $P_k^{\pi_{\text{net}}}$ using data available at the base station according to π_{net}, and Π_{net} is the space of causal network sensing policies.

References

Adame T., Carrascosa-Zamacois M., and Bellalta B. Time-sensitive networking in ieee 802.11be: On the way to low-latency wifi 7. *Sensors*, 21(15), 2021. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/21/15/4954.

Åkerberg J., Gidlund M., Lennvall T., Neander J., and Björkman M. Efficient integration of secure and safety critical industrial wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2011(1):100, 2011. ISSN 1687-1499.

Allan A. The big benchmarking roundup, Aug. 2019. URL https://aallan.medium.com/the-big-benchmarking-roundup-a561fbfe8719.

Ballotta L., Peserico G., and Zanini F. A reinforcement learning approach to sensing design in resource-constrained wireless networked control systems. In *2022 61th IEEE Conference on Decision and Control (CDC)*, pages 1–8, 2022.

Ballotta L., Schenato L., and Carlone L. Computation-Communication Trade-Offs and Sensor Selection in Real-Time Estimation for Processing Networks. *IEEE Trans. Netw. Sci. Eng.*, 7(4):2952–2965, 2020.

Barreiro A., Baños A., and Delgado E. Reset control of the double integrator with finite settling time and finite jerk. *Automatica*, 127:109536, May 2021. ISSN 0005-1098. URL https://www.sciencedirect.com/science/article/pii/S000510982100056X.

Bredel M. and Bergner M. On The Accuracy of IEEE 802.11g Wireless LAN Simulations Using OMNeT++. ACM, 5 2010.

Buja G. and Menis R. Dependability and Functional Safety: Applications in Industrial Electronics Systems. *IEEE Industrial Electronics Magazine*, 6(3):4–12, 2012.

Cao Y., Chen C., St-Onge D., and Beltrame G. Distributed TDMA for Mobile UWB Network Localization. *IEEE Internet of Things Journal*, 8(17):13449–13464, 2021.

Casares M., Pinto A., Wang Y., and Velipasalar S. Power consumption and performance analysis of object tracking and event detection with wireless embedded smart cameras. In *Int. Conf. Signal Process. Commun. Syst.*, pages 1–8, 2009.

Cavalcanti D., Perez-Ramirez J., Rashid M. M., Fang J., Galeev M., and Stanton K. B. Extending Accurate Time Distribution and Timeliness Capabilities

Over the Air to Enable Future Wireless Industrial Automation Systems. *Proceedings of the IEEE*, 107(6):1132–1152, 2019.

CC-Link. Overview of CC-Link Partner Association (CLPA). Technical report, CLPA, 2017.

Che F., Ahmed A., Ahmed Q. Z., Zaidi S. A. R., and Shakir M. Z. Machine Learning Based Approach for Indoor Localization Using Ultra-Wide Bandwidth (UWB) System for Industrial Internet of Things (IIoT). In *2020 International Conference on UK-China Emerging Technologies (UCET)*, pages 1–4, 2020.

Cheng L., Zhao A., Wang K., Li H., Wang Y., and Chang R. Activity Recognition and Localization based on UWB Indoor Positioning System and Machine Learning. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0528–0533, 2020.

CIP. THE COMMON INDUSTRIAL PROTOCOL (CIP). Technical report, ODVA, 2016.

Dalsa T. Genie nano series datasheet, 2021. URL https://www.stemmer-imaging.com/media/uploads/cameras/dalsa/12/122239-Teledyne-DALSA-Genie-Nano-Series-Manual.pdf.

Dang L., Yang H., and Teng B. Application of Time-Difference-of-Arrival Localization Method in Impulse System Radar and the Prospect of Application of Impulse System Radar in the Internet of Things. *IEEE Access*, 6:44846–44857, 2018.

Dawy Z., Saad W., Ghosh A., Andrews J. G., and Yaacoub E. Toward Massive Machine Type Cellular Communications. *IEEE Wireless Communications*, 24(1): 120–128, 2017.

Devos T., Kirchner M., Croes J., Desmet W., and Naets F. Sensor Selection and State Estimation for Unobservable and Non-Linear System Models. *Sensors*, 21 (22):7492, January 2021. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/21/22/7492.

EN 50325–5. Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces – Part 5: Functional safety communication based on EN 50325–4. Standard, European Committee for Electrotechnical Standardization (CENELEC), 2010.

EtherCAT. EtherCAT – The Ethernet Fieldbus. Technical report, EtherCAT Technology Group, 2019.

Ethernet POWERLINK. Powerlink basics. Technical report, (Ethernet POWERLINK Standardisation Group (EPSG), 2015.

Fedullo T., Morato A., Tramarin F., Rovati L., and Vitturi S. A Comprehensive Review on Time Sensitive Networks with a Special Focus on Its Applicability to Industrial Smart and Distributed Measurement Systems. *Sensors*, 22(4), 2022. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/22/4/1638.

Feng D., Haase-Schütz C., Rosenbaum L., Hertlein H., Gläser C., Timm F., Wiesbeck W., and Dietmayer K. Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Trans. Intell. Transp. Syst.*, 22(3):1341–1360, 2021.

Ferrigno L., Miele G., Milano F., Pingerna V., Cerro G., and Laracca M. A UWB-based localization system: analysis of the effect of anchor positions and robustness enhancement in indoor environments. In *2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6, 2021.

Frotzscher A., Wetzker U., Bauer M., Rentschler M., Beyer M., Elspass S., and Klessig H. Requirements and current solutions of wireless communication in industrial automation. In *2014 IEEE International Conference on Communications Workshops (ICC)*, pages 67–72, 2014.

Functional-safety-market report. Mordor intelligence: Functional safety market – growth, trends, and forecasts (2020 – 2025). https://www.mordorintelligence.com/industry-reports/functional-safety-market. [Online; accessed 17-November-2020].

Gast M. *802.11 Wireless Networks: The Definitive Guide.*

Girs S., Sentilles S., Asadollah S. A., Ashjaei M., and Mubeen S. A Systematic Literature Study on Definition and Modeling of Service-Level Agreements for Cloud Services in IoT. *IEEE Access*, 8:134498–134513, 2020.

Gopalakrishnan A., Kaisare N. S., and Narasimhan S. Incorporating delayed and infrequent measurements in Extended Kalman Filter based nonlinear state estimation. *J. Process Control*, 21(1):119–129, 2011. ISSN 0959-1524. URL https://www.sciencedirect.com/science/article/pii/S095915241000199X.

Greco L., Fontanelli D., and Bicchi A. Design and Stability Analysis for Anytime Control via Stochastic Scheduling. *IEEE Trans. Autom. Control*, 56(3):571–585, 2011.

Gros S., Zanon M., Quirynen R., Bemporad A., and Diehl M. From linear to nonlinear MPC: bridging the gap via the real-time iteration. *Int. J. Control*, 93(1): 62–80, January 2020. ISSN 0020-7179. URL https://doi.org/10.1080/00207179.2016.1222553.

Gu Y. and Yang B. Clock Compensation Two-Way Ranging (CC-TWR) Based on Ultra-Wideband Communication. In *2018 Eighth International Conference on Instrumentation Measurement, Computer, Communication and Control (IMCCC)*, pages 1145–1150, 2018.

Gungor V. C. and Hancke G. P. Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches. *IEEE Transactions on Industrial Electronics*, 56(10):4258–4265, 2009.

Güler S., Abdelkader M., and Shamma J. S. Peer-to-Peer Relative Localization of Aerial Robots With Ultrawideband Sensors. *IEEE Transactions on Control Systems Technology*, 29(5):1981–1996, 2021.

Hadziaganović A., Atiq M. K., Blazek T., Bernhard H.-P., and Springer A. The performance of openSAFETY protocol via IEEE 802.11 wireless communication. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2021.

Hadzic S. and Rodriguez J. Utility based node selection scheme for cooperative localization. In *2011 International Conference on Indoor Positioning and Indoor Navigation*, pages 1–6, 2011.

Hamer M. and D'Andrea R. Self-Calibrating Ultra-Wideband Network Supporting Multi-Robot Localization. *IEEE Access*, 6:22292–22304, 2018.

Han Sangjin L. S., Lee Sungjin. Node distribution-based localization for large-scale wireless sensor networks. *Wireless Networks*, 2010.

Hashemian H. Aging management of instrumentation & control sensors in nuclear power plants. *Nuclear Engineering and Design*, 240(11):3781–3790, 2010. ISSN 0029-5493. URL https://www.sciencedirect.com/science/article/pii/S0029549310005285.

Hashemian H. Wireless sensors for predictive maintenance of rotating equipment in research reactors. *Annals of Nuclear Energy*, 38(2):665–680, 2011. ISSN 0306-4549. URL https://www.sciencedirect.com/science/article/pii/S0306454910003324.

Hespanha J. P., Naghshtabrizi P., and Xu Y. A Survey of Recent Results in Networked Control Systems. *Proceedings of the IEEE*, 95(1):138–162, 2007.

Hossain S. and Lee D.-j. Deep Learning-Based Real-Time Multiple-Object Detection and Tracking from Aerial Imagery via a Flying Robot with GPU-Based Embedded Devices. *Sensors*, 19(15):3371, January 2019. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/19/15/3371.

IEC 61158. Industrial communication networks – Fieldbus specifications -. Standard, International Electrotechnical Commission, 2019.

IEC 61508. IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems. Standard, International Electrotechnical Commission, 2016.

IEC 61784-3. Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions, IEC 61784-3. Standard, International Electrotechnical Commission, 2016.

IEC 61784-3-3. Industrial communication networks - profiles - part 3-3: Functional safety fieldbuses - additional specifications for CPF 3. Standard, International Electrotechnical Commission, 2016.

IEEE802.11. Ieee standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, Dec 2016.

IEEE802.15.4z. Ieee standard for low-rate wireless networks–amendment 1: Enhanced ultra wideband (uwb) physical layers (phys) and associated ranging techniques. *IEEE Std 802.15.4z-2020 (Amendment to IEEE Std 802.15.4-2020)*, pages 1–174, 2020.

Ikram W., Jansson N., Harvei T., Fismen B., Svare J., Aakvaag N., Petersen S., and Carlsen S. Towards the development of a SIL compliant wireless hydrocarbon leakage detection system. In *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–8, 2013.

Jaakkola T., Jordan M., and Singh S. Convergence of Stochastic Iterative Dynamic Programming Algorithms. In **Cowan J., Tesauro G., and Alspector J.**, editors, *Adv. Neural Inf. Process. Syst.*, volume 6. Morgan-Kaufmann, 1993. URL https://proceedings.neurips.cc/paper/1993/file/5807a685d1a9ab3b599035bc566ce2b9-Paper.pdf.

Jasperneite J., Sauter T., and Wollschlaeger M. Why We Need Automation Models: Handling Complexity in Industry 4.0 and the Internet of Things. *IEEE Industrial Electronics Magazine*, 14(1):29–40, 2020.

Jeon H. and Eun Y. A Stealthy Sensor Attack for Uncertain Cyber-Physical Systems. *IEEE Internet Things J*, 6(4):6345–6352, August 2019. ISSN 2327-4662.

Jeppesen B. P., Rajamani M., and Smith K. M. Enhancing functional safety in FPGA-based motor drives. *The Journal of Engineering*, 2019(17):4580–4584, 2019.

John A. K. and Bhattacharjee A. K. Qualification of Hardware Description Language Designs for Safety Critical Applications in Nuclear Power Plants. *IEEE Transactions on Nuclear Science*, 67(3):502–507, 2020.

Kim B.-J. and Lee S.-B. Safety Evaluation of Autonomous Vehicles for a Comparative Study of Camera Image Distance Information and Dynamic Characteristics Measuring Equipment. *IEEE Access*, 10:18486–18506, 2022.

Lee H. and Lee J. Limited Non-Preemptive EDF Scheduling for a Real-Time System with Symmetry Multiprocessors. *Symmetry*, 12(1), 2020. ISSN 2073-8994. URL https://www.mdpi.com/2073-8994/12/1/172.

Li J., Bi Y., Li K., Wang K., Lin F., and Chen B. M. Accurate 3D Localization for MAV Swarms by UWB and IMU Fusion, 2018. URL https://arxiv.org/abs/1807.10913.

Li Y., Ma L., Zhong Z., Liu F., Chapman M. A., Cao D., and Li J. Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review. *IEEE Trans. Neural Netw. Learn. Syst.*, 32(8):3412–3432, 2021.

Liu W., Yan Z., Wang J., Ning Z., and Min Z. Ultrawideband Real-Time Monitoring System Based on Electro-Optical Under-Sampling and Data Acquisition for Near-Field Measurement. *IEEE Transactions on Instrumentation and Measurement*, 69(9):6603–6612, 2020.

Lu Y. Industry 4.0: A Survey on Technologies, Applications and Open Research Issues. *Journal of Industrial Information Integration*, 6:1–10, june 2017. ISSN 2452-414X.

Lu Y.-M., Sheu J.-P., and Kuo Y.-C. Deep Learning for Ultra-Wideband Indoor Positioning. In *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1260–1266, 2021.

Luong N. C., Hoang D. T., Gong S., Niyato D., Wang P., Liang Y., and Kim D. I. Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. *IEEE Communications Surveys Tutorials*, 21(4):3133–3174, 2019.

Macoir N., Ridolfi M., Rossey J., Moerman I., and De Poorter E. MAC Protocol for Supporting Multiple Roaming Users in Mult-Cell UWB Localization Networks. In *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pages 588–599, 2018.

Martalo M., Perri S., Verdano G., De Mola F., Monica F., and Ferrari G. Improved UWB TDoA-based Positioning using a Single Hotspot for Industrial IoT Applications. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2021.

McElroy C., Neirynck D., and McLaughlin M. Comparison of wireless clock synchronization algorithms for indoor location systems. In *2014 IEEE International Conference on Communications Workshops (ICC)*, pages 157–162, 2014.

Medeiros H., Park J., and Kak A. Distributed Object Tracking Using a Cluster-Based Kalman Filter in Wireless Camera Networks. *IEEE J. Sel. Top. Signal Process.*, 2(4):448–463, August 2008. ISSN 1932-4553. URL http://ieeexplore.ieee.org/document/4629874/.

Mildenhall B., Srinivasan P. P., Tancik M., Barron J. T., Ramamoorthi R., and Ng R. Nerf: Representing scenes as neural radiance fields for view synthesis. In **Vedaldi A., Bischof H., Brox T., and Frahm J.-M.**, editors, *ECCV*, pages 405–421, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58452-8.

Miller L. E. Validation of 802.11a/UWB Coexistence Simulation. Nist pubs, NIST. URL https://doi.org/10.6028/NIST.WCTG.10-17-2003.

Morato A., Peserico G., Fedullo T., Tramarin F., and Vitturi S. Tuning of a simulation model for the assessment of Functional Safety over Wi-Fi. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, pages 1–6, 2021a.

Morato A., Vitturi S., Cenedese A., Fadel G., and Tramarin F. The Fail Safe over EtherCAT (FSoE) protocol implemented on the IEEE 802.11 WLAN. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1163–1170, September 2019.

Morato A., Vitturi S., Fedullo T., Peserico G., and Tramarin F. A Profinet Simulator for the Digital Twin of Networked Electrical Drive Systems. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1099–1102, 2020.

Morato A., Vitturi S., Tramarin F., and Cenedese A. Assessment of Different OPC UA Implementations for Industrial IoT-Based Measurement Applications. *IEEE Transactions on Instrumentation and Measurement*, 70:1–11, 2021b.

Mumtaz S., Alsohaily A., Pang Z., Rayes A., Tsang K. F., and Rodriguez J. Massive Internet of Things for Industrial Applications: Addressing Wireless IIoT Connectivity Challenges and Ecosystem Fragmentation. *IEEE Industrial Electronics Magazine*, 11(1):28–33, March 2017.

Nguyen D. C., Cheng P., Ding M., Lopez-Perez D., Pathirana P. N., Li J., Seneviratne A., Li Y., and Poor H. V. Enabling AI in Future Wireless Networks: A Data Life Cycle Perspective. 2020. URL https://arxiv.org/abs/2003.00866.

Nguyen T.-M., Hanif Zaini A., Wang C., Guo K., and Xie L. Robust Target-Relative Localization with Ultra-Wideband Ranging and Communication. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2312–2319, 2018.

Nian R., Liu J., and Huang B. A review on reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, 139:106886, 2020.

OMNeT++ . Omnet++ network simulation framework, 2020. URL https://omnetpp.org.

OPC UA Safety. OPC Unified Architecture Part 15: Safety. Standard, OPC Foundation, 2019.

Oral H. G., Mallada E., and Gayme D. On the Role of Interconnection Directionality in the Quadratic Performance of Double-Integrator Networks. *IEEE Trans. Autom. Control*, pages 1–1, 2021. ISSN 1558-2523.

Pavlidakis M., Mavridis S., Chrysos N., and Bilas A. TReM: A Task Revocation Mechanism for GPUs. In *Proc. IEEE HPCC/SmartCity/DSS*, pages 273–282, 2020.

Peserico G., Fedullo T., Morato A., Tramarin F., Rovati L., and Vitturi S. SNR-based Reinforcement Learning Rate Adaptation for Time Critical Wi-Fi Networks: Assessment through a Calibrated Simulator. In *2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6, 2021a.

Peserico G., Fedullo T., Morato A., Tramarin F., and Vitturi S. Wi-Fi based Functional Safety: an Assessment of the Fail Safe over EtherCAT (FSoE) protocol. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2021b.

Peserico G., Fedullo T., Morato A., Tramarin F., and Vitturi S. Ultra-Wideband for Distance Measurement and Positioning in Functional Safety Applications. In *2022 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT)*, pages 1–6, 2022.

Peserico G., Fedullo T., Morato A., Vitturi S., and Tramarin F. Rate Adaptation by Reinforcement Learning for Wi-Fi Industrial Networks. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1139–1142, 2020.

Peserico G., Morato A., Tramarin F., and Vitturi S. Functional Safety Networks and Protocols in the Industrial Internet of Things Era. *Sensors*, 21(18), 2021c. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/21/18/6073.

Pimentel V. and Nickerson B. G. A Safety Function Response Time Model for Wireless Industrial Control. In *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, pages 3878–3884, Dallas, TX, USA, 2014. IEEE. ISBN 978-1-4799-4032-5.

Profibus. PROFIBUS System Description Technology and Application. Technical report, member of PROFIBUS & PROFINET International, 2019.

Profinet. PROFINET System Description Technology and Application. Technical report, member of PROFIBUS & PROFINET International, 2019.

Profisafe. Industrial communication networks - Profiles - Part 3-3: Functional safety fieldbuses - Additional specifications for CPF 3. Standard, International Electrotechnical Commission, 2016.

Radisavljevic-Gajic V., Park S., and Chasaki D. Vulnerabilities of Control Systems in Internet of Things Applications. *IEEE Internet Things J.*, 5(2):1023–1032, April 2018. ISSN 2327-4662.

Rao V. and Bernstein D. Naive control of the double integrator. *IEEE Control Syst. Magazine*, 21(5):86–97, October 2001. ISSN 1941-000X.

Refaat T. K., Daoud R. M., Amer H. H., and Makled E. A. Wifi implementation of wireless networked control systems. In *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, pages 145–148, 2010.

Rekkas V. P., Sotiroudis S., Sarigiannidis P., Wan S., Karagiannidis G. K., and Goudos S. K. Machine Learning in Beyond 5G/6G Networks—State-of-the-Art and Future Trends. *Electronics*, 10(22), 2021. ISSN 2079-9292. URL https://www.mdpi.com/2079-9292/10/22/2786.

Ren W. On consensus algorithms for double-integrator dynamics. In *2007 46th IEEE Conference on Decision and Control*, pages 2295–2300, 2007.

Ridolfi M., Van de Velde S., Steendam H., and De Poorter E. Analysis of the Scalability of UWB Indoor Localization Solutions for High User Densities. *Sensors*, 18 (6), 2018. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/18/6/1875.

Robinson S. H. Living with the challenges to functional safety in the industrial Internet of Things. In *Living in the Internet of Things (IoT 2019)*, pages 1–6, 2019.

Schluse M., Priggemeyer M., Atorf L., and Rossmann J. Experimentable Digital Twins-Streamlining Simulation-Based Systems Engineering for Industry 4.0. *IEEE Transactions on Industrial Informatics*, 14(4):1722–1731, 2018.

Seo D.-J., Kim T. G., Noh S. W., and Seo H. H. Object following method for a differential type mobile robot based on Ultra Wide Band distance sensor system. In *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, pages 736–738, 2017.

Sha K., Shi W., and Watkins O. Using Wireless Sensor Networks for Fire Rescue Applications: Requirements and Challenges. In *2006 IEEE International Conference on Electro/Information Technology*, pages 239–244, 2006.

Shen W., Zhang T., Barac F., and Gidlund M. PriorityMAC: A Priority-Enhanced MAC Protocol for Critical Traffic in Industrial Wireless Sensor and Actuator Networks. *IEEE Transactions on Industrial Informatics*, 10(1):824–835, 2014.

Sisinni E., Saifullah A., Han S., Jennehag U., and Gidlund M. Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734, Nov 2018.

Sommer C. and Dressler F. Using the Right Two-Ray Model? A Measurement–based Evaluation of PHY Models in VANETs. In *MobiCom*, 2011.

Sun S., Petropulu A. P., and Poor H. V. MIMO Radar for Advanced Driver-Assistance Systems and Autonomous Driving: Advantages and Challenges. *IEEE Signal Process. Mag.*, 37(4):98–117, 2020.

Sun S. and Zhang Y. D. 4D Automotive Radar Sensing for Autonomous Vehicles: A Sparsity-Oriented Approach. *IEEE J. Sel. Topics Signal Process.*, 15(4):879–891, 2021.

Sutton R. S. Learning to Predict by the Methods of Temporal Differences. *Mach. Learn.*, 3(1):9–44, aug 1988. ISSN 0885-6125. URL https://doi.org/10.1023/A:1022633531479.

Sutton R. S. and Barto A. G. *Reinforcement Learning: An Introduction.* The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

T. Adame M. C. and Bellalta B. Time-Sensitive Networking in IEEE 802.11be: On the Way to Low-latency WiFi 7. *arXiv:1912.06086*, November 2020.

Terabee . Follow-me-specification-sheet, 2020. URL https://terabee.b-cdn.net/wp-content/uploads/2020/04/Follow-Me-Specification-Sheet.pdf.

Tou R. and Zhang J. Imm approach to state estimation for systems with delayed measurements. *IET Signal Process.*, 10(7):752–757, 2016. URL https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-spr.2015.0345.

Tramarin F., Vitturi S., Luvisotto M., and Zanella A. On the Use of IEEE 802.11n for Industrial Communications. *IEEE Transactions on Industrial Informatics*, 12(5):1877–1886, 2016.

Tsai M.-C. and Gu D.-W. *Robust and Optimal Control.* Advances in Industrial Control. Springer London, London, 2014. ISBN 978-1-4471-6256-8 978-1-4471-6257-5. URL http://link.springer.com/10.1007/978-1-4471-6257-5.

Tzoumas V., Carlone L., Pappas G. J., and Jadbabaie A. Lqg control and sensing co-design. *IEEE Trans. Autom. Control*, 66(4):1468–1483, 2021.

Vitturi S., Zunino C., and Sauter T. Industrial Communication Systems and Their Future Challenges: Next-Generation Ethernet, IIoT, and 5G. *Proceedings of the IEEE*, 107(6):944–961, June 2019. ISSN 0018-9219.

Vitturi S., Carreras I., Miorandi D., Schenato L., and Sona A. Experimental Evaluation of an Industrial Application Layer Protocol Over Wireless Systems. *IEEE Transactions on Industrial Informatics*, 3(4):275–288, 2007.

Watkins C. J. C. H. and Dayan P. Q-learning. *Mach. Learn.*, 8(3):279–292, May 1992. ISSN 1573-0565. URL https://doi.org/10.1007/BF00992698.

Weyer S., Schmitt M., Ohmer M., and Gorecky D. Towards Industry 4.0 - Standardization as the crucial challenge for highly modular, multi-vendor production systems. *IFAC-PapersOnLine*, 48(3):579–584, 2015. ISSN 2405-8963. URL https://www.sciencedirect.com/science/article/pii/S2405896315003821. 15th IFAC Symposium onInformation Control Problems inManufacturing.

Wijethilaka S. and Liyanage M. Survey on Network Slicing for Internet of Things Realization in 5G Networks. *IEEE Communications Surveys and Tutorials*, 23, 2021.

Willig A., Matheus K., and Wolisz A. Wireless Technology in Industrial Networks. *Proceedings of the IEEE*, 93(6):1130–1151, 2005.

Willig A. Recent and Emerging Topics in Wireless Industrial Communications: A Selection. *IEEE Transactions on Industrial Informatics*, 4(2):102–124, 2008.

WirelessHART. HART Field Communication Protocol Specification, Rev. 7.7. 2020 . Standard, 2020.

Wong S. H. Y., Yang H., Lu S., and Bharghavan V. Robust Rate Adaptation for 802.11 Wireless Networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, MobiCom '06, page 146–157, New York, NY, USA, 2006. Association for Computing Machinery.

Xie G., Li Y., Han Y., Xie Y., Zeng G., and Li R. Recent Advances and Future Trends for Automotive Functional Safety Design Methodologies. *IEEE Transactions on Industrial Informatics*, 16(9):5629–5642, 2020.

Xie G., Zeng G., Liu Y., Zhou J., Li R., and Li K. Fast functional safety verification for distributed automotive applications during early design phase. *IEEE Transactions on Industrial Electronics*, 65(5):4378–4391, 2018.

Yang B., Li J., and Zhang H. Resilient Indoor Localization System Based on UWB and Visual–Inertial Sensors for Complex Environments. *IEEE Transactions on Instrumentation and Measurement*, 70:1–14, 2021.

Yang D., Ma J., Xu Y., and Gidlund M. Safe-WirelessHART: A Novel Framework Enabling Safety-Critical Applications Over Industrial WSNs. *IEEE Transactions on Industrial Informatics*, 14(8):3513–3523, 2018.

Yang H., Zhang K., Zheng K., and Qian Y. Leveraging Linear Quadratic Regulator Cost and Energy Consumption for Ultrareliable and Low-Latency IoT Control Systems. *IEEE Internet Things J.*, 7(9):8356–8371, September 2020. ISSN 2327-4662.

Yang Z. and Liu Y. Quality of Trilateration: Confidence Based Iterative Localization. In *2008 The 28th International Conference on Distributed Computing Systems*, pages 446–453, 2008.

Yu W., Jia M., Liu C., and Ma Z. Task Preemption Based on Petri Nets. *IEEE Access*, 8:11512–11519, 2020.

Zhang W., Zhu X., Zhao Z., Liu Y., and Yang S. High Accuracy Positioning System Based on Multistation UWB Time-of-Flight Measurements. In *2020 IEEE International Conference on Computational Electromagnetics (ICCEM)*, pages 268–270, 2020.

Zhang Y. and Duan L. Toward Elderly Care: A Phase-Difference-of-Arrival Assisted Ultra-Wideband Positioning Method in Smart Home. *IEEE Access*, 8:139387–139395, 2020.

Zheng T., Gidlund M., and Åkerberg J. WirArb: A New MAC Protocol for Time Critical Industrial Wireless Sensor Network Applications. *IEEE Sensors Journal*, 16 (7):2127–2139, 2016.

Zilberstein S. Using Anytime Algorithms in Intelligent Systems. *AI Magazine*, 17 (3):73, Mar. 1996. URL https://ojs.aaai.org/index.php/aimagazine/article/view/1232.

Zurawski R. *Industrial Communication Technology Handbook.*

Åkerberg J., Gidlund M., Lennvall T., Neander J., and Björkman M. Efficient integration of secure and safety critical industrial wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2011.

Åkerberg J., Reichenbach F., and Björkman M. Enabling safety-critical wireless communication using WirelessHART and PROFIsafe. In *2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, pages 1–8, 2010.