Università degli Studi di Napoli Federico II
Ph.D. Program in
Information Technology and Electrical Engineering
XXXV Cycle

Thesis for the Degree of Doctor of Philosophy

Performance of wireless networks for IoT and CPS: empirical evaluation and cybersecurity applications

by

Giovanni Stanco

Advisor: Prof. Giorgio Ventre

Co-advisor: Prof. Alessio Botta

Co-advisor: Ing. Flavio Frattini

Scuola Politecnica e delle Scienze di Base
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

To all the people that changed my life

Performance of wireless networks for IoT and CPS: empirical evaluation and cybersecurity applications

Ph.D. Thesis presented
for the fulfillment of the Degree of Doctor of Philosophy
in Information Technology and Electrical Engineering
by

Giovanni Stanco

October 2022

Approved as to style and content by

_____

Prof. Giorgio Ventre, Advisor

_____

Prof. Alessio Botta, Co-advisor

_____

Ing. Flavio Frattini, Co-advisor

Università degli Studi di Napoli Federico II
Ph.D. Program in Information Technology and Electrical Engineering
XXXV cycle - Chairman: Prof. Stefano Russo - http://itee.dieti.unina.it

Candidate's declaration

I hereby declare that this thesis submitted to obtain the academic degree of Philosophiæ Doctor (Ph.D.) in Information Technology and Electrical Engineering is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.
Parts of this dissertation have been published in international journals and/or conference articles (see list of the author's publications at the end of the thesis).

Napoli, January 15, 2023

_____

Giovanni Stanco

Abstract

The next industrial and technological revolution will be prompted by the connection of countless sensing Cyber-physical systems (CPSs) to the Internet, realizing the so-called Internet of Things (IoT). A promising innovation in this context is the introduction of network technologies known as Low Power Wide Area Network (LPWAN) that enable the connection of compact sensing devices deployed in outdoor environments to the Internet. Several are the domain fields in which IoT is tested and deployed, and several are the companies that are trying to take advantage of this revolution, projected to be worth at least $5.5 trillion by 2030. There are still some obstacles to fully exploit this new opportunity: the main of them is the absence of full knowledge about the possible application and usage of these new networks. Safety and security are two factors that should always be considered, for providing protection to customers and for preventing financial losses to companies.

This work aims to study if and how it is possible to have secure CPSs connected in IoT networks for security-critical applications. In order to fill this gap, we perform an empirical analysis of the overall performance of a system based on LPWAN, comparing the three most popular technologies nowadays available on the market. The information collected from this analysis is exploited for tuning a monitoring platform that collects data about the status of a CPS. Furthermore, the continuous monitoring of the status of the network connection enables the capability of discerning components under attack and excluding these malicious entities from a secure system. In this way we are able to realize secure communications between legitimate components of a CPS.

Keywords: IoT, CPS, wireless networks, LPWAN, industrial IoT, security assessment.

Sintesi in lingua italiana

La prossima rivoluzione industriale e tecnologica sarà spinta dalla connessione di innumerevoli CPS a Internet, realizzando il cosiddetto IoT. Un'innovazione promettente in questo contesto è l'introduzione di tecnologie di rete note con il nome di LPWAN, che permettono la connessione a Internet di dispositivi compatti che vengono installati in spazi aperti, capaci di raccogliere dati sensoriali. Numerosi sono gli scenari applicativi in cui viene testato e impiegato IoT, e numerose sono le aziende che stanno cercando di trarre vantaggio da questa rivoluzione, stimata di valere almeno 5,5 trilioni di dollari nel 2030. Ci sono ancora degli ostacoli che impediscono di sfruttare a pieno questa nuova opportunità: il principale è l'assenza di una conoscenza completa sul possibile uso di queste nuove reti. L'incolumità e la sicurezza sono due fattori che devono essere sempre considerati, per garantire protezione agli utenti finali e per evitare perdite economiche alle aziende.

Questo lavoro ha lo scopo di dimostrare che è possibile collegare in modo sicuro sistemi cyber-fisici in una rete IoT anche per applicazioni che richiedono sicurezza. Per realizzare ciò, in primo luogo eseguiamo un'analisi empirica delle prestazioni di un sistema basato su LPWAN, confrontando le tre tecnologie più popolari tra quelle disponibili al momento sul mercato. Le informazioni raccolte da questa analisi possono essere sfruttate per regolare una piattaforma di monitoraggio che raccoglie dati sullo stato di un CPS. Il monitoraggio continuo dello stato della rete permette inoltre di discriminare le componenti sotto attacco e di escludere queste entità malevole da un sistema sicuro. In questo modo è possibile realizzare comunicazioni sicure tra componenti legittime di un CPS.

Parole chiave: IoT, CPS, reti wireless, LPWAN, industrial IoT, valutazione di sicurezza.

Contents

Acknowledgements

The research presented in this dissertation has been supported by RisLab – Laboratorio di Ricerca e Innovazione per la Sicurezza S.r.l.
The author's work has been partially carried out at School of Computing and Communications at Lancaster University (United Kingdom), from November 2021 until April 2022, under the supervision of Dr Matthew Bradbury.

List of Acronyms

The following acronyms are used throughout the thesis.

3GPP	3rd Generation Partnership Project
ACL	Access control list
BLE	Bluetooth Low Energy
CCM	Counter with CBC-MAC
CPS	cyber-physical system
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
GFSK	Gaussian Frequency Shift Keying
GOT	Goodness of throughput
IoT	Internet of Things
LPWAN	Low Power Wide Area Network
LTE	Long term Evolution
MIC	Message integrity code
MitM	Man in the middle

NB-IoT	Narrowband Internet of Things
OTAA	Over-The-Air-Activation
ROS	Robot Operating System
TDMA	Time division multiple access
UE	User equipment
WSN	Wireless Sensor Network

List of Figures

List of Tables

Chapter 1

Introduction

Information and communication systems are starting to be embedded in the environment around us, with the goal of collecting and storing an enormous quantity of data. The convergence of the digital and physical worlds will not only change the operations of enterprises, but will also have a revolutionary impact on the lives of consumers. A system able to integrate both cyber and physical components using computing and communication technologies is known as a CPS in literature [11]. The interaction between cyber and physical elements is very emphasised, since the physical components are monitored and controlled using sensing, computing, and communication technologies.
The combination of several CPSs capable of measuring environmental parameters led to the birth of Wireless Sensor Network (WSN) technologies. A WSN is made of a large number of interconnected nodes, that exploit wireless connectivity to transmit sensor data. An example of CPS typically employed in a WSN consists of a radio transceiver, an antenna, a microcontroller, a sensor, and a power supply. The widespread diffusion of this technology and the creation of a system of devices connected in a communicating network lay the foundations for IoT. Sensors and actuators are interconnected in IoT and share information, realizing the revolution of an ubiquitous computing Internet [12].
The application settings and domains that can be impacted by IoT are very different [13], starting from smart homes, which provide the capability of monitoring appliances and services such as heating, air conditioning, light-

ning sensors, and security defenses. The use of IoT in industrial automation will allow machines to produce more efficiently, and to send immediate maintenance requests when needed. A better resource managing will also be enabled by smart grids: energy distribution and consumption can be monitored by connecting smart meters to the network of energy providers. Other useful services for the community come from a smart city environment, in which several systems are interconnected to provide intelligent utilities. Cities are also starting to be equipped with road-side equipment that can interact with intelligent vehicles for the aim of safe, efficient, and enjoyable driving and transportation. Logistics is also a promising domain for IoT: freight movement scheduling can be improved by monitoring travel times, routes, and queue lengths. Sensors and actuators are used not only on machines, but on people as well: an example is their use in medical patients for collecting information about their physiological status. These several application domains show that IoT is not limited to the extent of a Personal Area Network, but it creates a global infrastructure that connects devices able to move even to long distances. Special attention has been recently dedicated to the LPWAN technologies, that enable communication between resource-constrained devices employed in wide-area applications with the Internet.

The variety of application highlights the opportunity of revenue that comes with IoT. According to [14], IoT could enable at least $5.5 trillion in value globally, considering also the value captured by IoT customers. The setting that will account for the largest amount (26%) of potential economic value are environments such as manufacturing factories and hospitals: the day-to-day management of assets and people could become much more efficient in such settings. The second most profitable field would be human-health applications that affect the human body, that will represent around $10 - 14\%$ of the value. IoT solutions in this setting have been used not only by individuals, but also by institutions and governments, stimulated by the COVID-19 pandemic.

The new challenges addressed by IoT brought the development of new technologies to satisfy these needs. One example is the introduction of wireless communications that could enable applications in which devices are no longer in a domestic environment, but are deployed at a large distance from an access point or gateway, for use cases that require mobility of the

devices and the assets. **A complete knowledge about the possible application and usage of these networks is currently not available. Investigation about the usage and the performance of the newly designed wireless technologies is however very important, since their deployment in commercial solutions means that they are the foundation of important economic return for companies.** A failure in the communications of these systems can represent a relevant economic loss in business-critical systems in which constant availability is a key requirement. Correct communications are also extremely important in safety-critical systems, in which the security and the health of customers must never be in danger.

The openness of these networks makes IoT devices an easy target for several attacks at different layers. Physical attacks require the access to the devices, so they are harder to perform, but attacks to the network layer and to the routing protocol are very common. Well known attacks to wireless systems, such as eavesdropping and replay, are also applicable in these scenarios. There is then a need of investigating several aspect of the new communication technologies, in order to know that our application can always be available and work properly and that security is granted.

Our goal is to study if and how it is possible to have secure CPSs connected in IoT networks for applications that have strict security requirements. The approach we follow in our work focuses on three complementary aspects that contribute to the fulfilment of secure communications. A preliminary aspect is the overall assessment of the performance of LPWAN communications, focusing on the practical feasibility of these networks. Aspects considered in our analysis regard the availability of our systems, monitoring the sending and receiving of messages and the correct functioning of devices. The evaluated performance metrics give developers and customers useful insight on how LPWAN can satisfy the communication requirements of safety-critical CPSs. After the performance analysis, we show how to exploit the gathered knowledge for tuning a monitoring platform for a CPS. The continuous monitoring of the resources and the status of a CPS enables the possibility of exploiting the constrained resources typically involved in such systems. Another benefit provided by the introduction of continuous monitoring is the possibility of making informed decision on the basis of the gathered information. In par-

ticular, we exploit the information on network performance to understand which are the entities within the system that are corrupted and subject to a malicious attack. The analysis of the effect that network attacks have on the behaviour of the components of our system enables to discriminate the entities that can be trusted in a task offloading application and which ones have to be excluded. Being able to discriminate and isolate the malicious components, we can reach a security level, that is absolutely necessary in the safety-critical systems of our interest.
Chapter 2 provides a broad survey about the several networking possibilities, with special attention on their security features, the most common attacks and the countermeasures proposed in literature.
Chapter 3 is dedicated to the description of the tools used in our tests: we describe the actual devices, the commercial platforms, and the software employed in our tests. We then illustrate the methodology of our investigations.
Chapter 4 presents the results of our experiments. Our empirical evaluation of the performance of LPWAN focuses on the correct functioning and availability of an IoT prototype, evaluating the duration of the supply of a battery and the delivery of messages in terms of losses and latency. An interesting outcome of our tests is the variability of latency depending on the network employed: the delivery time of a message can significantly vary, from few seconds to more than 100 seconds. In this chapter we also describe what are the approaches followed for the development of a monitoring platform for CPSs, and for making decisions in the classification of corrupted communications during a task offloading activity.
Chapter 5 concludes, summarizing the thesis and its experimental validation.

Chapter 2

Background and related work

This chapter is dedicated to the analysis of literature about related topics of our work. The first contribution of this chapter is a thorough report on the several networking possibilities currently available for IoT, their security features, and the well-known attacks and countermeasures. Among these possibilities, we provide a comparison of the three LPWAN technologies, which gained the attention by industrial corporations thanks to their capability to connect a constrained CPS deployed in hard-to-reach locations to the Internet.
We then present the state of the art regarding three aspects that must be considered in order to realize a safe and secure IoT system. The first of them is the performance evaluation of the functioning of an IoT system based on LPWAN connectivity: in this section we highlight the critical gaps currently present in literature that prevent the successful deployment of IoT in business-critical systems. We also illustrate the status of research on resource monitoring solutions for a resource-constrained CPSs. The last part of the chapter focuses on the approaches that have been proposed for evaluating the goodness and the level of trust of the components of an IoT system.

Table 2.1. Short range networks technical specifications.

	BLE	**802.15.4**	**Z-Wave**	**ZigBee**
Modulation	GFSK	O-QPSK, BPSK	GFSK	OQPSK
Spreading	FHSS	DSSS	DSSS	DSSS
Band	2.4 GHz ISM band	868/915 MHz 2.4 GHz	868/908 MHz; 2.4 GHz	868/915 MHz 2.4 GHz
Data rate	1-2 Mbps	20, 40, 250 kbps	9.6, 40, 100 Kbps	20, 40, 100, 250 kbps
Range	5 m	10 m	30 m	10-100 m
Topology	Point-to-point Broadcast Mesh	Star, peer-to-peer	Mesh	Star, tree mesh
Scalability	-	65K nodes	232 nodes	> 65K nodes
MAC access	TDMA	TDMA, CSMA/CA	CSMA/CA	Freq. select.

Table 2.2. References about BLE.

BLE	
General description	[15, 16, 17, 18, 19, 20, 21, 16, 22]
Security features	[23, 20, 18, 16, 24]

2.1 IoT network technologies and security aspects

2.1.1 Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a wireless technology that operates in the unlicensed ISM band at 2.4 GHz, designed by the Bluetooth Special Interest Group for application in healthcare, fitness, home and security scenarios [15]. BLE offers the same range as Bluetooth, but with a reduced power consumption and a lower data rate transfer [16]. BLE is designed to broadcast in a range between -30 dBm to 0 dBm, with an operating range of up to five meters [17]. BLE utilizes a Gaussian Frequency Shift Keying (GFSK) modulation and occupies 40 RF channels with 2 MHz spacing [22]. 37 of these channels are used to transfer data while the remaining 3 are used to broadcast device information and establish connections. BLE transmits across the channels ranging from 2404 MHz to 2478 MHz using Frequency Hopping Spread Spectrum. BLE protocol is similar to Bluetooth but they are not compatible [18]. Every BLE packet begins with an 8 bit preamble, followed by a 32 bit access address (AA), the variable length Protocol Data Unit and the 24 bit CRC.
A BLE device can communicate in two modes: broadcasting mode or communication mode. In broadcasting mode, advertising packets containing information about the broadcaster available for pairing are sent. A master device interested in initiating a connection sends a Scan Request to the peripheral asking for more parameters and it is later possible to establish a connection. Once a BLE peripheral is connected to a master device, the communication is carried out over the 37 data channels using adaptive frequency hopping. When two devices are connected, data is sent in bursts in order to save energy. Point-to-point communications in BLE operate under a master/slave model, where the master is the user and the slave is the device that waits for the connection [20]. In most cases the master is the client while the slave is the server. The master coordinates the medium access using a polling mechanism based on Time division multiple access (TDMA), that periodically polls the slaves [21].
BLE mesh networks that enable many-to-many communications have been proposed for overcoming coverage limitations [16]. Mesh networks can cover a wide physical area and contain a large number of devices. They use flooding as routing technique, so that every packet is replicated to all

the other nodes in range, increasing network reliability. Packets have a TTL counter while devices have a message cache in order not to retransmit old messages.

Security

BLE makes use of three keys [23]. The first key is the Temporary Key (TK), a 128-bit key used along with two random numbers from the master and the slave for encrypting the messages for the generation of the second key, the Short Term Key (STK). The STK is a 128-bit key used to encrypt a connection after pairing and it is used to establish a link-layer encrypted session over which the Long Term Key (LTK) is exchanged. The LTK is a 128-bit key used to encrypt the connection which is saved and used for all other communications.

Two processes occur during the initial connection and protect against malicious eavesdroppers and other basic attacks: pairing and bonding [20]. Pairing involves an exchange of security features and capabilities. Bonding is the second step, after the keys have been generated and exchanged. Bonding is a more permanent encryption method that saves the key for use in future connections. The pairing process consists of three phases [18]. In the first phase the device which wants to establish a connection will send a pairing-request to the device it wants to communicate to. The devices will then exchange their requirements about authentication and bonding and will choose the pairing method that best suits their capabilities. All the data in this phase are not encrypted, so an eavesdropper could begin its attack from this phase and then perform a Man in the middle (MitM) attack. In the second phase the devices will start generating or exchanging the TK and they will generate the STK used in the encryption. The third phase is optional and it is used to share keys specific to transport.

Three modes are available for the pairing process.

1. Just Works: the pairing method used by devices which do not have a display or a keypad. The devices do not ask for authentication to the other device, so anyone can connect. The TK is set to all zeros in this mode, so any eavesdropper can easily guess it.

2. Passkey: one device displays a 6-digit passkey and the other device has to enter the displayed key. The connection will be established using this 6-digit PIN as TK to provide authentication and to partly

prevent MitM attacks. Passkey entry provides slightly more protection but previous research showed that it can be brute forced.

3. Out of Band: the TK is shared between the devices by other technologies, for example NFC. This method is more secure than the others because it employs a 128 digit TK and, if the OOB channel is secure, it prevents attacker from eavesdropping. Another method is to send the TK over BT Classic, but it is not a common mode: the best option is to use a out-of-band channel.

An added function when using pairing and bonding is link layer encryption [20]. The encryption method used in versions 4.0 and 4.1 is derived from the devices being paired initially. Version 4.2 of BLE does not use a short-term key, but uses a key derived from an Elliptic-curve Diffie–Hellman (ECDH) key. The LTK requires the ECDH key, two random numbers from the master and from the slave, and the two Bluetooth address of the master and the slave. The ECDH key is never transmitted and is computed independently to protect against eavesdropping. The connection is encrypted and authenticated after the LTK is established.
Application layer encryption is one of the most popular methods for securing BLE devices [20]. New devices do not need to be paired, instead user and device establish key to encrypt and decrypt credentials. Using application layer encryption can be more complicated due to the managing of keys, however it adds an extra layer of security.
As mentioned before, BLE also offers the possibility to deploy mesh networks, which provide an improvement to the more common star topology since they do not have a single point of failure [16]. Every message is encrypted and authenticated at two different levels: network and application. Network encryption protects from outside access, while application encryption defines which are the operations allowed for a device. Three types of keys are used in a BLE mesh network: Network Key (NetKey), shared across all the nodes in the network; Application Key (AppKey), shared across the nodes that participate in a given mesh application; and Device Key (DevKey), only known to the network configuration device and the device itself. To setup a mesh network, a provisioner generates a NetKey and provisions devices with the NetKey and a unique network address. The provisioning of these keys is done following the Diffie-Hellman

Table 2.3. References about IEEE 802.15.4.

IEEE 802.15.4	
General description	[25, 26, 27, 28]
Security features	[29, 30, 31, 27]

key exchange algorithm to encrypt the transmitted parameters. The DevKey is also derived from the ECDH exchange on provisioner and device side. The messages are encrypted twice, once with the AppKey and a second time with the NetKey. Encryption and authentication are achieved using the AES-128 block cipher in Counter with CBC-MAC (CCM) operation mode, which provides both authentication and confidentiality. Each encrypted message with AES-CCM contains an authentication Message integrity code (MIC). Encryption and authentication of the destination address and the transport Protocol Data Unit are achieved using encryption keys derived from NetKeys.
To prevent replay attacks, every device of the mesh network increases the sequence number, so every message is sent with a unique pair of sequence number and source address. If a device does not work any more, the Bluetooth mesh provides a key refresh procedure to update the network and application keys to which the device was part. The key refresh procedure should be performed periodically and not only when a device is no longer available.

2.1.2 IEEE 802.15.4

IEEE 802.15.4 is a protocol designed for the physical (PHY) layer and the MAC layer of low rate wireless personal area networks (LR-WPAN) [25]. The main goals of LR-WPANs include ease of installation, short range operations, low cost, long battery life, and a simple protocol [28]. IEEE 802.15.4 is the dominant technology for WSN, and the basis for more complex networks, such as ZigBee, WirelessHART, etc. The PHY layer is responsible for the activation of the radio transceiver, energy detection, link quality indicator for received packets, and channel frequency selection. The MAC layer is used for reliable single hop communication among devices: it supports PAN association and disassociation and Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

for channel access. MAC also uses acknowledged frame delivery, performs frame validation, maintains network synchronization, and schedules the time slots.
Long battery life is achieved by using synchronization to their neighbors and to the gateways. When in sleep mode, the node turns off its radio to save power and stores all messages that it needs to send at the next transmission opportunity. When receiving, it turns on its radio before the scheduled receiving time, receives the data, sends an acknowledgement, turns off its radio, delivers the data to the upper layer, and goes back to sleep.
IEEE 802.15.4 works on the 868/915 MHz and 2.4 GHz bands and provides a bitrate of 20, 40 and 250 Kb/s respectively. The low band (868/915 MHz) adopts binary phase shift key (BPSK) modulation: it offers one channel with a raw data rate of 20 kbps when operating in the 868 MHz band, 10 channels with a raw data rate of 40 kbps in the 915 MHz band. The high band (2.4 GHz) adopts offset quadrature phase shift key (O-QPSK) modulation and has 16 channels with a raw data rate of 250 kbps [26].
Two network topologies are available for IEEE 802.15.4: star topology with a PAN coordinator and devices identified by a 64-bit address, and peer-to-peer topology with a PAN coordinator and the possibility for devices to communicate directly [28]. Two kind of devices can participate in a 802.15.4 network: Full Function Devices (FFDs), which work as a controller responsible for creation and maintenance of the network, and Reduced Function Devices (RFDs), simple nodes with restricted resources [27]. The FFD acts as a PAN coordinator and can initiate, terminate, and route a communication, while an RFD can perform a logical role of end devices with simple applications [26]. Three types of data transfer are allowed by this standard. The first one is the data transfer from a device to a coordinator; the second is the data transfer from a coordinator to a device; the third is the data transfer between two peer devices. The first two methods are used in star topology, all three methods can be used in a peer-to-peer technology. Four types of frames are available: beacon frames used by coordinators, data frames used for transfers, acknowledgment frames for confirming reception and MAC command frames used for handling MAC peer entity control transfers. The data frame has variable length and indicates its type, if security is enabled, the addressing modes,

and if it needs an acknowledgment. A beacon-enabled PAN is used when synchronization or low-latency is required, and it is controlled by a coordinator. The coordinator broadcasts beacons in order to achieve device association and synchronization. The channel contains active and inactive periods, with or without contention. A beacon is received by a node, which then sends request to the coordinator for Guaranteed Time Slot (GTS) in the Contention Access Period. In case of availability of the GTS slots, the coordinator, through the beacon frame, allocates them to the desired nodes. The data is communicated in the GTS slots of the subsequent superframes [32].
In 2008, IEEE 802.15.4e was created to extend 802.15.4 support and support low power communication. IEEE 802.15.4e introduces channel hopping for changing the frequency channel using a pre-determined random sequence, reducing the effect of interference and multipath fading. Other versions include IEEE 802.15.4-2011, whose MAC protocol has a high consume of energy since receivers must be active all the time for multihop routing, and 802.15.4-2012, which retains small duty cycles through the time synchronization and channel hopping techniques.

Security

The 802.15.4 specification addresses the security needs through a link-layer security package [29]. The security layer is handled at the MAC layer, while the application specifies its security requirements and explicitly enables security. A link layer security protocol provides four basic security services: access control, message integrity, message confidentiality and replay protection [30]. Access control means that the link layer protocol should prevent unauthorized parties from participating from the network using an Access control list (ACL). A secure 802.15.4 network provides authentication and integrity introducing a MIC, a cryptographically secure checksum. Computing it requires authorized senders and receivers to share a secret key. The MIC can be either 4, 8 or 16 bytes long: a longer MIC provides increased protection against authentication attacks. IEEE 802.15.4 does not include key management and device authentication schemes. Keys are assumed to be provided by higher layer processes and key management and key establishment are not specified. 802.15.4 offers different security modes by utilizing the *security enabled bit* in the Frame Control field in the header [31]. Acknowledgment packets do not

support security, while other packet types can support integrity protection and confidentiality protection [29].
Three security modes are defined to achieve different security objectives: unsecured mode, ACL mode and secured mode. Unsecured mode does not provide any security service; ACL mode maintains an ACL for limited security services and reception of messages from a limited number of devices, without cryptographic protection; the secured mode provides all the security services according to the security suite in terms of confidentiality, message integrity, access control and sequential freshness. An application can choose among security suites that control the type of protection provided for transmitted data [29]. The 802.15.4 specification defines eight different security suites that offer different security properties and guarantees. The offered properties are: no security, encryption only (AES-CTR), authentication only (AES-CBC-MAC), and encryption and authentication (AES-CCM). Each category that supports authentication is available in three versions according to the size of the MIC: 4, 8, or 16 bytes. For each suite that offers encryption, replay protection can be optionally offered. The Null suite and the AES-CCM-64 suites are required to be implemented, while the others are optional. An application indicates the chosen suite based on source and destination address. Radio chips decide the security suite and the keying information to use on the basis of their ACL.
Devices may support up to 255 ACL entries containing an address, a security suite identifier and security material. If the application requests security, the MAC layer looks up the destination address in its ACL table, looking for a match. On packet reception, the MAC layer determines if security suites have been applied to that packet, and potentially looks for an ACL entry based on the address of the sender and applies the appropriate security suite. A receiver can optionally enable replay protection when using a suite that provides confidentiality protection. The recipient compares the replay counter from the incoming packet to the highest value seen, as stored in the ACL entry: if the incoming packet has a larger replay counter than the stored one, the packet is accepted and the new replay counter is saved. Otherwise, the packet is rejected.

Table 2.4. References about Z-Wave.

Z-Wave	
General description	[33, 34, 35, 36, 37, 38]
Security features	[39, 33, 37, 40]

2.1.3 Z-Wave

Z-Wave, a proprietary standard marketed by the Z-Wave Alliance, is one of the most common implementation of the ITU-T G.9959 recommendation [33]. Z-Wave is a short-term wireless communication technology offering the advantages of low cost and low energy consumption, covering about 30 meters point-to-point communications.
All Z-Wave products adhere to the ITU-T G.9959 Physical, MAC, Segmentation and Reassembly (SAR), and Logical Link Control (LLC) layer specification, ensuring interoperability between vendor devices. They differentiate on the network and application layer [34].
The PHY layer uses either Manchester or Non Return Zero encodings to transmit data [39]. The PHY layer supports three data rates: 9.6 kbps (R1), 40 kbps (R2) and 100 kbps (R3). The PHY Protocol Data Unit consists of the PHY Header, PHY Service Data Unit (PSDU) and the End of Frame (EoF). The maximum PSDU size is 170 B when operating at 100 kbps, 64 B when operating at 9.6 or 40 kbps.
The MAC layer uses CSMA/CA to moderate access to the wireless medium, and it is also responsible for frame acknowledgement, data validation, and retransmissions. MAC frames can be single-cast, multicast and acknowledgemnt. A MAC Protocol Data Unit frames (MPDU) contains a header, consisting of identification and control fields, and a payload, consisting of data pertaining to an application layer command.
The Application layer contains commands and parameters specific to the device and manufacturer. The payload frame specifies if the command is single/multi or broadcast and the command classes.
Two types of nodes are available in Z-Wave networks: control nodes (gateways), that send commands and manage all the devices, and slave nodes. There can be up to 232 slave nodes in a Z-Wave network and they can also forward commands to other nodes that are not directly reachable by the control node, with a hop limit of four nodes.

The controller is the central entity of the network, to which every device is linked. There can be multiple control nodes to cover a larger area, but only one of them can be the primary controller, which carries the reliable information about the network topology, while the secondary controllers get the network routing information from the primary controller. Each controller contains a routing table that represents the full topology information of the network and allows to find the efficient paths [35]. A controller is identified by a unique value named Home ID, set by the manufacturer when the controller is built. It has a length of 32 bits and identifies the network [34]. The secondary controllers contain the same Home ID as the primary and each device in a Z-Wave network uses this Home ID in each message sent, to make it identifiable by every other device in the same network. The primary controller also assigns a 8-bit Node ID to each device during a pairing process [36]. A user wishing to add a device to the network puts the Z-Wave controller and the new device into a pairing mode. While in pairing mode, the controller adds any device found in pairing mode [37]. Association/Pairing is the main feature allowing to create a Z-Wave network. The controller keeps track of the paired nodes and it can not send a message to a device until it is paired. The pairing/inclusion procedure begins when the controller and the device are in inclusion mode, which is triggered by pressing a button on the device. Once in inclusion mode, the controller listens for a Node Information Frame (NIF) emitted by the slave device. The controller replies with the Home ID of the network. The slave sends another NIF to confirm that is part of the network. The device and the central controller must share a network key that allows communication. When a new device is paired via Z-Wave, a specific syncing protocol is executed in order to share this network key with the device. A preamble packet is sent between the receiver and transmitter, containing home ID, and node ID. In this period the protocol becomes susceptible to attack, as unencrypted identifying information is being transmitted.

Security

Security evaluations of the Z-Wave protocol are difficult because developers must sign non-disclosure agreements preventing them to reveal any proprietary information, however it has been shown that the protocol can be reversed engineered [33]. Customers need to implement the provided hardware and the proprietary software protocol stack library, but security

Table 2.5. References about ZigBee.

ZigBee	
General description	[41, 42, 43, 44, 45, 46, 47]
Security features	[48, 40, 46, 47]

issues may rise because of mistakes of programmers during the implementation of the protocol. Other vulnerabilities suffered by Z-Wave devices are due to the interoperability with older devices which do not support encrypted and authenticated communications, making them vulnerable.
Z-Wave utilizes several security mechanisms, which include cryptographic, behavior detection and out of band mechanisms [37]. Z-Wave provides the Security command class for confidentiality, source integrity, and data integrity. Application frames may be encapsulated in a security frame that is both encrypted and signed. The frame is secured through symmetric encryption using AES-128 and three shared keys, known by every node on the network. When devices are paired in secure mode, a key is exchanged between the node and the controller, and it is used to encrypt messages. Although supported, encryption is not required. The security command class encrypts only the application layer message. In Cipher Block Chaining (CBC) mode, the integrity and authenticity of the source and destination address, length of the encrypted payload and encrypted payload are guaranteed by a MIC [40]. Z-Wave devices trust the source and destination fields of the MAC Protocol Data Unit (MPDU) frames: this makes impersonation attacks easy to implement.

2.1.4 ZigBee

ZigBee is a wireless technology designed for short term communications, characterized by low cost, low complexity, low energy consumption and low data rates [41]. It is standardized by the ZigBee Alliance [49]. It operates in unlicensed bands at 2.4 GHz, 915 MHz and 868 MHz for WPANs supporting up to 65000 devices [42]. ZigBee supports different data rates according to the employed frequency: 250 kb/s at 2.4 GHz, 40 and 250 kb/s at 915 MHz, 20, 100, and 250 kb/s at 868 MHz. ZigBee is based on Direct Sequence Spread Spectrum (DSSS) technique and

Offset Quadrature Phase Shift Keying (OQPSK) modulation [43, 44]. A ZigBee network may comprise three type of devices: Coordinator, Router and End Devices. ZigBee can support multiple topologies: star, tree, and mesh. The coordinator is responsible for initiating and establishing the network: it chooses the channels used to communicate, it gives permission to other devices to join or leave the network, and it keeps track of all the routes. The coordinator also works as Trust Center: it enables end-to-end security between devices and stores and distributes the network keys [45]. Routers act as intermediate between the coordinator and the end devices and establish routes using a routing protocol similar to the ad hoc On-Demand Distance Vector.
ZigBee is built upon the physical and MAC layers defined in the 802.15.4 standard, while it defines the network and application layers. The network layer handles network management and routing by invoking actions in the MAC layer, and is responsible for the construction of the network topology. It also transmits messages through multi-hop links, broadcasts route request messages, and processes received route reply messages [46]. The application layer specifies frame formats for transporting data and provides a data service to the applications [47].

Security

The NWK layer is responsible for securely transmitting outgoing frames and receiving incoming frames. When security is applied to a network layer protocol data unit, an auxiliary frame header is present. Upper layers set up the appropriate keys and frame counters, and establish which security level to use. The application layer is responsible for transmitting and receiving frames and establish and manage cryptographic keys. The security level identifier indicates how an outgoing frame is secured and indicates whether the payload is encrypted or not.
ZigBee provides sequential freshness, using an ordered sequence of inputs to reject frames that have been replayed. It also provides frame integrity checking function, using a MIC to protect data from being modified by parties without the cryptographic key. Data encryption uses a symmetric cipher to protect data from being read by parties without the cryptographic key. ZigBee uses Counter mode with Cipher-block chaining message authentication code (CCM*) of AES-128 to provide cryptography. CCM* is a minor modification of CCM, eliminating the need for CTR and CBC-

MAC modes [48]. The CCM mode is a mode of operation only for 128-bit cryptographic block ciphers: it combines the counter mode with the CBC-MAC authentication and uses the same encryption key for both modes. The CCM* mode coincides with the original CCM mode specification for messages that require authentication and, possibly, encryption, but offers support also for message that require only encryption.
The Trust Center works as Trust Manager, authenticating devices that request to join the network, as Network Manager, maintaining and distributing network keys, and as Configuration Manager, enabling end-to-end security between devices. In the Join procedure an end device has to authenticate itself, and to exchange security information items with the Trust Center. ZigBee relies on three types of keys: Master, Link and Network keys. The Master Key (MK) may be pre-installed during manufacturing, installed by a Trust Center or be user-entered. The MK is shared by two devices in the network, and it is used as the basis for long-term secure communication. The MK is used for the generation of the Link Keys, which are used for secret sessions between two communication devices [40]. The Link Key (LK) is shared between two devices and it is used to ensure the security of unicast communication. Communication between application peer entities is secured by 128-bit Link Key shared by two devices, while broadcast communications are secured by a 128-bit Network Key [46]. The Network Key (NK) performs network layer security and it is shared by all the devices in the network. It protects network frames and prevents unauthorized accesses with little resource requirements at devices.
ZigBee provides two security models: Security Mode, designed for lower security residential applications, and High Security Mode, for high security commercial applications [47]. A trade-off between the two standard models is given by the security model of the Smart Energy Profile (SEP), which can be considered as a reference security model for ZigBee applications. SEP provides device descriptions and standard practices for demand-response and load management applications. It also provides standard interfaces and device definitions to allow interoperability among ZigBee devices [47].

Table 2.6. References about Sigfox.

Sigfox	
General description	[50, 51, 52, 53, 54, 53]
Security features	[51, 55, 56, 57]

2.1.5 Sigfox

Sigfox is a widely spread LPWAN technology that provides end-to-end low power, wide area connectivity using an ultra-narrowband IoT communications system designed to support IoT deployments over long ranges. Sigfox is based on patented technologies developed in 2010 by a French company, which has adopted an operator model, essentially creating a cellular network for IoT devices. Sigfox Network Operators (SNOs) around the world deploy the base stations equipped with software-defined radios and connect them to the backend servers using an IP-based network [50]. Sigfox ready devices are not directly connected to the Internet and do not communicate using the internet protocol, but broadcast radio messages which are picked by several access stations and conveyed to the Sigfox Core Network.
When a device needs to send data, it broadcasts a radio message which is then picked up by several access stations and conveyed to the Sigfox Core Network [51]. Each uplink message is transmitted three times on different frequencies to help ensure it will be delivered. The end devices can autonomously choose a random frequency channel to transmit their messages while the base stations can scan all the channels to decode the messages. Since downlink transmissions are rare, there is no acknowledgement functionality, and thus retransmissions are a way to ensure that the message will have a better chance of reaching a base station. Sigfox initially supported only uplink communication and only later evolved in a bidirectional technology, although with a significant link asymmetry. When the application wants to send a message to the node, it will have to wait for the limited receiving time slot to be allocated, so that the IoT application has the opportunity to deliver the response to the device. Sigfox implements 3 diversity schemes to make transmissions robust: time diversity, because payloads are transmitted in 3 consecutive radio frames; frequency diversity, because these three frames are transmitted at different random

subcarrier; space diversity, because the frames are transmitted to multiple base stations [54].
Sigfox offers lower duty cycles, resulting in an improved battery lifetime [57]. In particular, the duty cycle of its frequency band is maximum 1%, allowing a device to transmit only 36 seconds every hour. The number and size of messages over the uplink are limited to 140 12-byte messages per day to conform to the regional regulations on use of license free spectrum, while over the downlink only 4 8-bytes messages per day are allowed. It means that the acknowledgment of every uplink message is not supported and to provide reliability of the uplink communication, time and frequency diversity and redundant transmissions are used.
Every base station deployed by SNOs is connected to the Sigfox Cloud, the central hub where messages and information from the devices are stored. To retrieve the data from the Sigfox Cloud, two methods are available: the Callback API and the REST API. Callback queries are one-way only HTTP requests: when the Sigfox Cloud receives a message from an emitting device, it instantly generates a callback message and sends it to the user's servers. The system forwards any new content when it receives it so there is no need to regularly check for new data. REST API queries are bi-directional HTTP requests employed by the user's server to request and receive data from the Sigfox Cloud [52].
The end devices connect to the base stations in an ultra narrow sub-GHz ISM band carrier, namely the 868 MHz band in Europe and the 915 MHz band in the United States. Sigfox operates in unlicensed bands so its signals must coexist with other signals in the same band. In order to utilize bandwidth efficiently, Sigfox uses Ultra Narrow Band (UNB) modulation types, in which bandwidth occupation is sensibly lower. Among the other benefits provided by UNB there are: low-power consumption, low-cost components in the transceiver part, and very low noise levels resulting in high receiver sensitivity. All these benefits come at the expense of the maximum throughput, only 100 bps, which limits the number of use-cases for this technology. Sigfox operates different uplink and downlink modulations. The scheme adopted for downlink is the Gaussian Frequency-Shift Keying (GFSK) scheme, while differential Binary Phase-Shift Keying (D-BPSK) is employed for uplink. D-BPSK has the advantage of bringing a high efficiency in the spectrum medium access and it is easy to imple-

ment [53]. The spectrum is divided in 400 channels of 100 Hz, starting at 868.180 MHz for channel 0 and ending at 868.220 MHz for channel 400. Channels 181-219 are reserved and not used.
The Sigfox protocol stack consists of Physical layer, MAC layer, Frame layer and, Application layer. The PHY layer implements the Sigfox radio transmission and determines how to synthesize the signals. The MAC layer controls the assembly and disassembly of data and add fields for the identification and authentication of the device (HMAC) and for error detection (CRC). The Frame layer allows the generation of the radio frames starting from the Application layer and it adds a preamble and a sequence number.
The Sigfox technology uses small packets: the frame will have 26 bytes for a maximum 12 bytes payload. The protocol overhead is reduced so less energy is consumed. The packet contains a preamble (4 B), a frame synchronization part (2 B), a device identifier (4 B), a payload of up to 12 B, a Hash code to authenticate the packet in Sigfox network (variable length), and a CRC of 2 B for security and error detection.

Security

Sigfox designers have applied security-by-design principles in the development of the Sigfox protocol and infrastructure. The Sigfox Core Network is a cloud based network hosted in secured data centers in which every component is redundant, monitored and scalable. Sigfox data centers and network architecture are protected against DDoS attacks by using proprietary detection and mitigation algorithms matching Sigfox specific traffic patterns to prevent false positives [51]. The Sigfox Core Network has then the capacity to monitor and detect traffic anomalies. This reduces the impact of DoS attacks based on the radio segment targeting the Sigfox network and rules out Sigfox devices as DDoS attack vectors. Moreover, Sigfox devices do not have the ability to send data to arbitrary entities via internet and are shielded by a very strict firewall [51] and it is not possible to access an end point through Internet maliciously [55].
Each device is provisioned during manufacturing with: the Porting Authorization Code (PAC), a one-time activation code employed for the registration of the device to the network; the Network Authentication Key (NAK), a unique symmetrical authentication key for device authentication that does not change during the device lifetime; and a unique device

Table 2.7. References about LoRaWAN.

General description	[58, 9, 59, 60, 61, 62]
Security features	[63, 64, 65, 66, 67, 68]

identifier, that allows the network server to identify the device [56]. Since the key stored is unique per device, the compromising of one device has a very limited impact on the whole network. However, security practices and secure storage will be implemented by the device designer in order to avoid traffic analysis and eavesdropping.
End-to-end authentication, message integrity and protection against replay attacks are critical aspects in the Sigfox ecosystem. In order to detect replay attempts, Sigfox uses a 12-bit Sequence Number that is transmitted in every uplink message. Sigfox uses a hash-based message authentication code (HMAC) with unique pre-shared symmetrical key for message integrity and authentication [57]. The message authentication code is generated as the result of a symmetric authentication algorithm, function of the message, the NAK and the message sequence number. When the server receives a message, it will retrieve the device ID, the associated NAK and the respective sequence number, verifying whether the message is valid or not.
Sigfox offers resistance to interference and jamming by sending each message on three random frequencies [57]. All the base stations in the range of a device receive the message, thus a high level of redundancy is provided. By default, data is conveyed without any encryption between end devices and base stations. Sigfox gives the option to implement a custom end-to-end encryption solution, or to use Advanced Encryption Standard (AES)-128 in CTR mode. The encryption key (KE) used to encrypt the messages payload, is generated through a Key Derivation Function starting from the NAK.

2.1.6 LoRaWAN

LoRa® is a physical layer technology standardized and maintained by the LoRa Alliance [69]. It modulates the signals in Sub-GHz ISM band using a proprietary chirp spread spectrum (CSS) technique, which spreads

a narrow band input signal over a wider channel bandwidth. The spectrum spreading in LoRa is achieved using a chirp signal that can be described by its instantaneous phase or a specific time function [58]. The resulting signal has low noise levels, low power characteristics, it is difficult to detect or jam. The spread spectrum provides orthogonal separation between signals by using a unique spreading factor (SF) to the individual signal, so that multiple frames can be exchanged in the network at the same time, as long as each one is sent with a different SF. Messages transmitted using different spreading factors can be received simultaneously by LoRa base stations. LoRa uses six spreading factors (SF7 to SF12) to adapt the data rate and range trade-off. Lower spreading factors enable faster transmission at the cost of minimum sensitivity, while higher spreading factors allow longer range and greater sensitivity at the expense of lower data rate [9]. The SF is automatically adapted as function of number of retransmissions if the traffic is acknowledged. Otherwise, devices can decide locally on their rate adaptation [59].

The LoRa Alliance, a special interest group constituted by several commercial and industrial partners proposed LoRaWAN™, an architecture consisting of layers above the LoRa physical layer. LoRaWAN defines the communication protocol and system architecture for the network while the LoRa physical layer enables the long-range communication link. LoRaWAN is a LPWAN technology offering a fully bidirectional symmetrical link between the endpoint and the gateway. It is a low-power consumption protocol designed for scalable wireless networks with millions of devices. LoRa operates at the ISM radio bands and LoRaWAN defines the operation frequencies in different regions, while telecommunications authorities define duty cycling rules for ISM bands.

A simple ALOHA scheme is used at the MAC layer that in combination with the LoRa physical layer enables multiple devices to communicate at the same time. The nodes in a LoRaWAN network are asynchronous and communicate when they have data ready to send. LoRaWAN does not use the clear channel assessment (CCA) mechanisms and relies exclusively on the end devices duty cycle based channel access mechanism. The end device selects the frequency channel to use in pseudo-random manner for each next packet to be transmitted. The use of duty-cycle based media access mechanism enables a LoRaWAN device to send the data with no

delays, reducing the communication latency and energy consumption. On the other hand, absence of clear channel assessment mechanism increases the probability of packet collisions.
LoRaWAN uses long range star architecture in which gateways are used to relay the messages between end devices and a central core network. The messages transmitted by the end devices are received by not a single but all the base stations in the range, giving rise to a star of stars topology. By exploiting this redundant reception, LoRaWAN improves the successfully received message ratio. End devices communicate with one or many gateways through single-hop LoRa communication while all gateways are connected to the core network server via backhaul IP connections (either cellular, Ethernet, satellite, or Wi-Fi). Gateways scan the spectrum and receive LoRa packets from end devices and forward their data to a network service which handles the packets. Only a small portion of these devices can be located sufficiently far away from the base station. Most of the devices, especially the ones with higher upload traffic needs, should be located in the vicinity of the base station [60]. The base stations connect end devices to network server, the brain of the LoRaWAN system that suppresses duplicate receptions, adapts radio access links and forwards data to the application servers. Multiple receptions of the same message by different base stations however are exploited for localizing end devices. The architecture of a LoRaWAN network is represented in figure 2.1. Two LoRaWAN network types are possible: a Private Network operated by private individuals or companies or a Public Network whose infrastructure is run by mobile network operators [61].

LoRaWAN defines three classes of end devices with different capabilities:

1. Class A: bidirectional end devices, each uplink transmission is followed by two short downlink receive windows, for the lowest power end device systems;

2. Class B: bidirectional end devices with scheduled receive slots, end devices receive a time-synchronized beacon from the base station to open receive windows at scheduled times;

3. Class C: bidirectional end devices with maximal receive slots and almost continuously open receive windows.

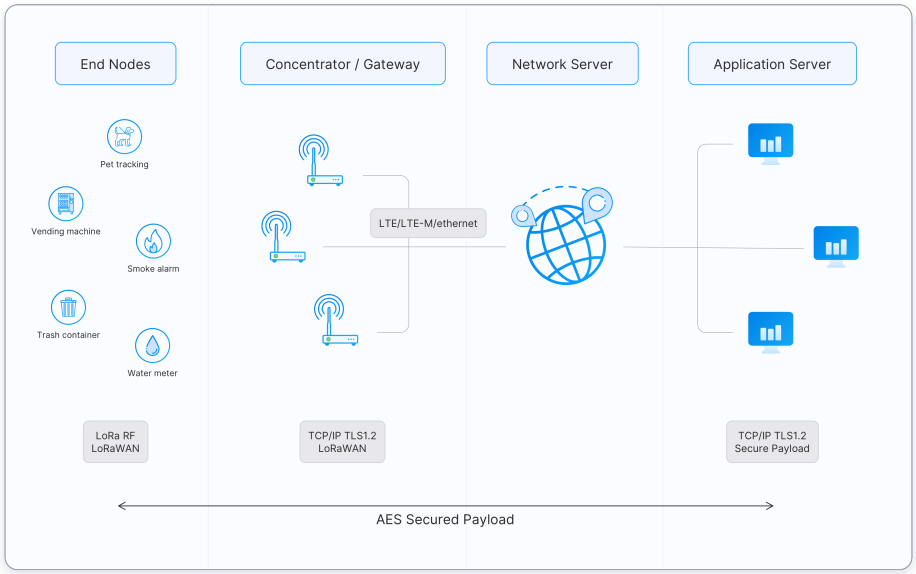


Figure 2.1. LoRaWAN architecture [1].

Uplink messages are sent by end-devices and must include a preamble, a physical layer header and CRC, a physical layer payload and its own CRC. The physical layer payload is formed by MAC layer headers, frame headers, payload and MIC. LoRaWAN packet structure does not include any time based data or signature to validate the time of the message, and this vulnerability might be exploited to perform replay attacks. The structure of a LoRaWAN packet is represented in figure 2.2.

A new version, LoRaWAN v1.1, was released in October 2017, introducing new features to the protocol, including a handover roaming mechanism and a new security architecture [62]. based on a new element, the Join Server, which allows the end devices to connect to the network. The Join Server manages requests to join the network from one or more end devices. In this new architecture the Network Server forwards application payloads to and from the Application Server and requests to join to the network to the Join Server. LoRaWAN v1.1 introduced a roaming architecture where a Network Server can have different roles: Home, Forwarding and Serving. The inclusion of this three servers aims to enable roaming of the devices

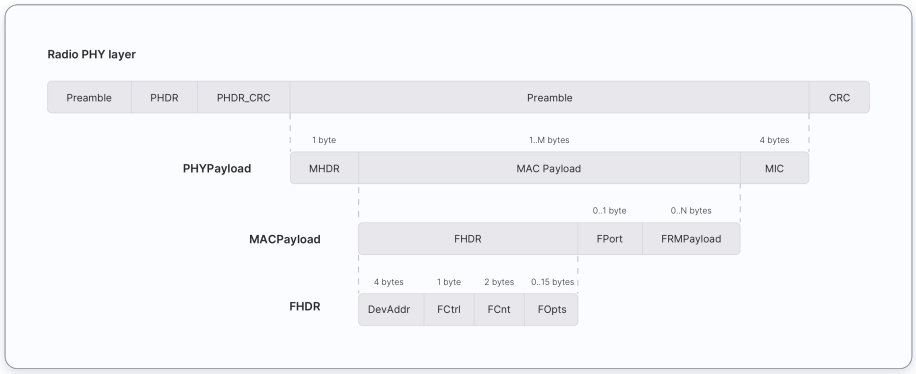


Figure 2.2. A LoRaWAN packet [2].

citywide, country wide or worldwide.

Security

All LoRaWAN end devices have a 64-bit identifier called Device Identifier (DevEUI), set by vendors and developers, while another identifier, the Application Identifier (AppEUI), identifies the application provider of the end device. All communication is done using a 32 bit device address [63]. There are different kinds of keys in LoRaWAN. AppKey and NwkKey are the AES-128 bit root keys that need to be stored in a hardware-secure way and tracked after fabrication and from which the session keys are generated. These root keys are specific to each end device and are embedded during fabrication. The AppKey is used to generate two session keys on which the security of LoRaWAN communication is based, the NwkSKey and the AppSKey [64]. The NwkSKey is used as a message authentication secret session key, while the data confidentiality of the MAC payload is ensured by using the AppSKey with the AES-128 algorithm. The NwkSKey is used to secure the MAC layer communication between the network server and the end node and to generate and verify the 4 byte MIC calculated over the whole LoRaWAN packet. The AppSKey is used for end-to-end encryption between the application server and the end node. Each message is encrypted by using the XOR operation with the corresponding key from the key stream to generate the encrypted payload [63]. When uplink messages arrive at the network server, the server will first check the mes-

sage integrity, discard the packets with an invalid MIC, and then transfer the message to the application server. Without knowing the NwkSKey, an attacker would need to send over 4 billion LoRaWAN packets in order to be sure about the correct MIC. This would take 136 years at a rate of 10 LoRaWAN packets per seconds, but exploiting multiple channels, different spreading factors and multiple gateways, this brute forcing would take less than 10 days [65].
An end device needs to be activated before it is able to communicate with the network server. The activation procedure also defines the way the end node gets its session keys. The two possible methods for end node activation are Activation by Personalization (ABP), which puts the session keys on the end node, and Over-The-Air-Activation (OTAA), which generates the session keys through the join procedure between the end node and the network server.
ABP directly connects end devices to the network without initiating a join procedure. When an end device wants to communicate with the server, it will send messages directly. The device address, NwkSKey and AppSKey are directly defined and stored in the end device, which can encrypt messages using these keys. These parameters are also stored on the server and remain valid throughout the whole lifetime of that end node. If the keys are compromised, all communication can be decrypted for the lifetime of the device.
OTAA provides a more flexible way of establishing session keys with the servers and it is considered more secure since for each session new keys should be generated [66]. In OTAA, the end device and the Network Server (NS) are provisioned with a unique 128-bit AppKey that is used to derive the two session keys. The end device initiates the OTAA procedure by sending a join request message including the AppEUI, the DevEUI and a random number, the DevNonce, to the Network Server. The DevNonce is included in the transmitted packet in order to avoid replay attacks of the join request [70]. Upon reception of a join request, the network server checks if the DevNonce has been already used by the device, comparing the received sequence with the last N sequences, where N is a system parameter to be chosen. If a match is found, two policies are usually implemented: the NS drops the request and waits for further requests with a valid DevNonce, or excludes permanently the end device from the network. The

NS should then store the DevNonces used in previous Join procedures, but the LoRaWAN specification does not specify how to manage the DevNonces and the 16 bits do not exclude that a benign node could generate a previously used DevNonce. The NS checks the integrity of the message and, after validation, it forwards the join request to the Application Server (AS), which checks the entry for this specific end device in the Supported Devices List, matching the DevEUI of the end device to its associated AppKey. After a successful match, the AS responds with a 3-byte AppNonce to the NS. If the device is accepted, the network server will send a Join Accept message containing the App Nonce and also a network identifier (NETID) and some radio and configuration parameters, along with a MIC. The end device validates the MIC and then decrypts the message to obtain the AppNonce, NETID and parameters. Finally, AppNonce and NETID are used to create the session-long keys AppSKey and NwkSKey. The end node can generate the session keys by using these parameters and share the same session keys with the network server. These session keys are used for confidentiality and integrity of the messages exchanged afterwards [67]. An end device must follow this procedure every time it joins a network or loses the session key information.
The Join procedure for OTAA in v1.1 changes [67]. When the end device is deployed, it communicates with the Join Server (JS), whose unique identifier JoinEUI is pre-configured in the end device during fabrication, to initiate the OTAA Join Procedure. The session starts with the join request message sent from the end device. The receiving NS checks the message and forwards the request to the JS which checks the entry for this specific end device in the Supported Devices List, matching the DevEUI of the end device to its associated NwkKey and AppKey. After a successful match, the JS responds with a JoinNonce. The NS then sends a join accept message containing the JoinNonce, a NETID and some radio and configuration parameters. To finalize the OTAA procedure, the JoinNonce, JoinEUI, DevNonce and NwkKey are used by the end device to create network session-long keys: NwkSEncKey, FNwkSIntKey, SNwkSIntKey. The FNwkSIntKey (Forwarding Network Session Integrity Key) is used for the MIC of uplink data messages. Whereas, the SNwkSIntKey (Serving Network Session Integrity Key) is used for the MIC of downlink data messages. NwkSEncKey and AppSKey keys (network and applica-

tion) are used for confidentiality and integrity of the messages exchanged afterwards.
LoRaWAN offers protection against replay attacks by increasing a frame counter for each message sent or received by an end node [68]. The values of the counters are maintained by both end node and network server and they are initialized to 0 every time the node joins the network. When the NS receives a message, it compares the counter stored on the server with the counter received by the end node and decides whether to reject the message. Counters are an important component in replay protection and, as the message counter is used in LoRaWAN to generate the key stream, are also essential to the confidentiality of the communication. For each end device, there are two frame counters named FCntUp and FCntDown. FCntUp is counting uplink messages in the end device, while FCntDown is counting downlink messages in the network server. In order to keep uplink and downlink messages in sync, there is a limit value MAX_FCNT_GAP: if the difference between number of uplink and downlink messages is larger than MAX_FCNT_GAP, subsequent frames will be discarded.
LoRaWAN does not rely on channel sensing or time synchronization for collision avoidance: its main defense is the low data rate of end devices. However, this is not a robust solution in large scale deployments. Collisions matter in LoRa networks when messages share the same SF or when one message is transmitted with significantly more power than the other.

2.1.7 NB-IoT and other cellular technologies

Three enabling technologies in cellular scenarios have been proposed in 3GPP Release 13, according to different requirements and different markets.
LTE enhancements for machine type communications (eMTC)
LTE-M (Long Term Evolution - Machine Type Communication), also called LTE Cat-M1, or Cat-M or enhanced MTC, is an evolution for machine-type communications on a Long term Evolution (LTE) network. It is a cellular LPWAN technology introduced in the 3GPP Release 13 standardization which intends to minimize modem complexity and cost and power consumption. Cat-M1 User equipment (UE) operates within a limited bandwidth of 1.08 MHz out of the available 1.4 MHz, allowing Cat-M1 UE to use only six physical resource blocks out of the eight avail-

Table 2.8. 3GPP standards for the IoT [8].

	eMTC	EC/GSM	NB-IoT
Deployment	In-band LTE	In-band GSM	In-band, Guard-band, standalone
Coverage	155.7 dB	164 dB	164 dB
Downlink	OFDMA	TDMA/FDMA	OFDMA
Uplink	SC-FDMA	TDMA/FDMA	SC-FDMA
Bandwidth	1.08 MHz	200 kHz	180 kHz
Power saving	PSM, ext. I-DRX, C-DRX	PSM, ext I-DRX	PSM, ext I-DRX, C-DRX

able 180 kHz LTE Physical Resource Blocks (PRBs). Cat-M1 devices are expected to achieve a maximum throughput of up to 1 Mbps in both uplink and downlink operations for massive IoT. To reduce the cost while being compliant to LTE systems requirements, 3GPP reduces the peak data rate and the complexity of modem and antenna design. eMTC is characterized by a reduced transmission power to have more cost-efficient and low-power design. To extend the battery lifetime for eMTC, 3GPP adopts two features, namely *Power Saving Mode* and *extended Discontinuous Reception*, that enable end devices to enter in a deep sleep mode for hours or days without losing their network registration. eMTC technology can be deployed within the regular LTE network up to 20 MHz of operation, coexisting with other LTE network services. eMTC is standardized to ensure that for Massive IoT deployment and coverage, it supports long battery life of about 10 years.

EC-GSM

Extended Coverage GSM is a compatible solution with a GSM/EDGE network. 3GPP standardization in its Release 13 specification introduced EC-GSM-IoT as a standard-based LPWAN emerging technology for the IoT, designed for high capacity, long range coverage, low energy and low complexity cellular system based on enhanced GPRS (eGPRS). Existing GSM networks can be upgraded using a software application in order to ensure extensive coverage. 3GPP aims to extend the GSM coverage using

Table 2.9. References about NB-IoT.

General description	[71, 72, 73, 74]
Security features	[55, 71, 75]

sub-GHz band for better signal penetration in indoor environments. EC-GSM exploits repetitive transmissions and signal processing techniques to improve coverage and capacity of legacy GPRS.

NB-IoT

Narrowband Internet of Things (NB-IoT), also known as LTE Cat-NB1, is standardized by 3GPP, and has been officially released in mid-2016. It uses cellular telecommunication band to connect and handle a massive number of connected devices and to prolong the battery operated nodes lifetime by using very aggressive sleep algorithms [71]. NB-IoT provides a way for connecting devices that require small amounts of data in area hardly accessible, offering low device cost, low battery consumption, and low response time. NB-IoT is expected to become part of the 5G mobile interconnection system [76].

NB-IoT is based on the LTE protocol and works on the 700, 800 and 900 MHz bandwidth. LTE functionalities are reduced to minimum and the functionalities required by IoT applications, such as power consumption, spectrum efficiency, and system capacity, are enhanced. Some of the removed features of LTE include handover, channel quality monitoring, carrier aggregation and dual connectivity. It is possible to introduce NB-IoT in an existing network in a small portion of available spectrum, easing the massive deployment of IoT. NB-IoT is designed for optimal co-existence performance with legacy GSM, GPRS and with LTE technologies, without compromising their performances. NB-IoT is not compatible with 3G but can be supported with only a software upgrade on top of existing LTE infrastructure. Reusing the existing cellular infrastructure can reduce the network installation and maintenance cost.

NB-IoT operates within a minimum system bandwidth of 180 kHz for both the downlink and uplink operations, respectively, so it is possible to replace one GSM carrier of 200 kHz or a LTE PRB of 180 kHz with an NB-IoT application. Three operation modes are possible: stand alone

operation, guard-band operation (usage of the unused frequency band of 180 kHz between the last PRB and the channelization edge), in-band operation (integrated as part of the resource regularly used for LTE). The stand alone deployment allows an effective reuse of the existing GSM carriers for IoT, while in-band and guard-band deployments use one PRB of LTE. NB-IoT can serve up to 50 thousands end devices per cell with the potential for scaling up the capacity by adding more carriers. Since it uses licensed bands, NB-IoT does not suffer from legal restrictions on its duty cycle or interference.

NB-IoT is designed to reuse the existing LTE design structure, which includes uplink Single-Carrier Frequency Division Multiple Access (SC-FDMA), downlink orthogonal Frequency Division Multiple Access (OFDMA) and interleaving. NB-IoT core network is based on the evolved packet system (EPS) and two optimizations for the Cellular Internet of Things (CIoT) were defined: the user plane CIoT EPS optimization and the control plane CIoT EPS optimization. Both planes choose the best path for control and user data packets, for uplink and downlink data.

An NB-IoT based radio can achieve a battery life of 10 years when transmitting 200 bytes of data per day on average, with a battery capacity of 5 Wh and uplink interval of 120 minutes. NB-IoT traffic is best effort and, during times of heavy voice/data traffic, NB-IoT application performance may be impacted. Furthermore only half of the messages are acknowledged due to limited downlink capacity, so it is necessary to implement some form of reliability mechanisms in the applications.

One of the goals of NB-IoT is to extend the coverage while reducing the UE transmit power. It was then designed to offer 20 dB coverage, which facilitates its penetration capability. Reducing system bandwidth elevates the UE transmitted PSD, but it degrades the data rate. Long battery life is achieved through Power Saving Mode (PSM) and extended discontinuous reception (eDRX). Moreover, since UL and DL can not work at the same time, NB-IoT keeps only one transceiver, shared by uplink and downlink, unlike LTE, lowering the power consumption.

NB-IoT architecture is divided in four sections: UE, which transmits the data to the base stations; network, consisting of gateway nodes and base stations (eNodeB); cloud, which receives and stores sensing data and performs data analysis; application server, which is the final aggregation point

and consists of various user applications, developed by various companies according to their requirements, that the user can exploit to interact with NB-IoT objects. The NB-IoT architecture constitutes 6 different protocol layers: the PHY layer, the MAC layer, the Radio Link Control (RLC) layer, the Packet data Convergence Protocol (PCDP) layer, the Radio Resource Control (RRC) layer and Non-Access Stratum (NAS) layer.
New 3GPP Releases introduce new features and mobility enhancements, extending its applicability to wearables and tracking services [74]. Release 14 added support for LTE features to increase functionality and the number of use cases covered. These features include support for new positioning methods, mobility and service continuity enhancements, new power class with a reduced output power. Release 15 introduced Time Division Duplexing (TDD) support, higher spectral efficiency, range enhancement, small cell support, and LTE Device to Device. In Release 15, coexistence of NB-IoT with 5G NR and eMTC was added, since NB-IoT devices are expected to live more than ten years and they are expected to be compatible with future 3GPP releases. The improvements introduced in Release 16 include improved UE power consumption, scheduling enhancement, network management enhancement and mobility enhancement.

Security

Chacko et al. [55] highlighted how the three layers that secure NB-IoT applications. The bottom is the perception layer, subject to passive attacks, in which the attacker monitors the network traffic, and to active attacks, in which the integrity of message is the target of the malicious entity. Cryptographic algorithms can be used to encrypt the data and provide authentication. The second layer is the transmission layer, which is complicated due to the use of high cost base stations. The third is the application layer, which collects a massive amount of data, that must be protected and must not be accessed by everyone.
NB-IoT inherits LTE's authentication and encryption. Data freshness is achieved using the sequence number and data availability is accomplished using the frequency hopping technique [71]. The support for mutual authentication and data integrity is provided through the ATR-128 CTR CMAC with 4 byte MIC, while data confidentiality is offered through ATR-128 CTR mode or SNOW 3G algorithms. The keys are managed by deriving a new HMAC-SHA256 session key for each individual session

key from each individual session between the UE and the network. ATR-128 is a proprietary algorithm which performs encryption, while SNOW 3G is a backup encryption algorithm used to provide data confidentiality in UMTS networks. CTR is the Counter Mode, that outputs a stream cipher from a block cipher. CMAC (Cipher-based Message Authentication Code) is a block cipher-based MAC scheme used to provide authenticity and integrity. MIC is a hash of the message which allows the receiving devices to ensure that the message has not been altered, ensuring the message integrity. MAC is a bit string that is transmitted together with a message and HMAC is an algorithm to convert hash functions into MACs. UE should always be protected since they are unattended and short of resources [77]. The passwords on the UEs are often weak and traditional software and firewall cannot provide efficient protection. To ensure the binding between SIM card and UE, the UE registers on the IoT platform. The platform stores the mapping relationship between the serial number of the terminal, the IMEI of the communication module, the IMSI of the SIM card and the MSISDN on the IoT platform. When the NB-IoT service is in operation, it checks whether the logical relationship is the same as the registered copy, otherwise the system refuses to offer service to the UE.
Mutual identity authentication between IoT platform and end nodes is required. Mutual authentication between the UE and the network is based on a pre-shared key and a cryptographic computation over a nonce generated by the Home Subscriber Server (HSS) [75]. The UE can verify the authenticity of the HSS and compute its authentication token that will be verified by the Mobility Management Entity. Once the UE and the HSS are authenticated, Ciphering and Integrity keys are derived from the combination of the pre-shared key and the nonce that will be used to protect further communications. The identification and secret keys are stored in a UICC (Universal integrated circuit card), resistant to physical attacks.
Another aspect to consider is privacy protection: NB-IoT is applicable in sensitive applications such as healthcare monitoring. The medical information about a patient should not be accessible by the network operator and the user information should not be leaked.

Table 2.10. LPWAN technical specifications [9, 10].

	Sigfox	**LoRaWAN**	**NB-IoT**
Modulation	D-BPSK (UL), GFSK (DL)	CSS	QPSK
Band	868 MHz (EU), 915 MHz (US)	868 MHz (EU), 915 MHz (US)	Licensed LTE bands 700,800 900 MHz
Bandwidth	100 Hz (UL), 600 Hz (DL)	125/500 kHz	180 kHz
Data rate (DL)	100 bps	0.25-50 kbps	0.5-200 kbps
Data rate (UL)	100 bps	0.25-50 kbps	0.2-180 kbps
Range (urban)	10 km	5 km	1 km
Range (rural)	40 km	20 km	10 km
Bidirectional	Limited	Yes	Yes
Adaptive Data Rate	No	Yes	No
Maximum payload length	12 B (UL), 8 B (DL)	243 B	1600 B
Maximum messages/day	140 (UL), 4 (DL)	Limited duty cycle	Unlimited

2.1.8 Comparison of LPWAN

The first evident difference between the three LPWAN technologies considered is the bands they work on. Both Sigfox and LoRaWAN work on the same ISM band (868 MHz in Europe and 915 MHz in US), while NB-IoT works on licensed LTE bands. The use of unlicensed bands impose some constraints on the duty cycle and the number of messages that it is possible to send daily. NB-IoT, which does not utilize free bands, does not suffer from this problem, but the amount of exchanged data could depend on the kind of subscription with the network operator. NB-IoT reuses cellular resources and infrastructure since it is compatible with GSM and LTE, Sigfox also follows an operator model, while for LoRaWAN it is necessary to deploy gateways where there is no coverage. The bitrate for the three technologies varies: Sigfox provides the smallest, only 100 bps, while NB-IoT provides the highest values, up to 200 kbps. LoRaWAN provides a variety of bitrates, according the Spreading Factor used: the SF allows to adapt the data rate and range trade-off, using higher bitrates for the devices closest to the gateway and lower bitrates for the furthest ones. LoRaWAN also introduces a feature not present in the other two technologies: classes of devices with different capabilities to receive downlink frames. On the other hand, a feature that is common to all three technologies is that not all the traffic is acknowledged, in order to save resources on the devices and not to occupy the wireless channels. All the three technologies considered provide coverage for several kilometers, with Sigfox providing the largest range and NB-IoT providing the smallest, both in urban and rural scenarios. This feature is undoubtedly one of the factors that aroused the interest in industrial operators: the possibility of deploying smart compact devices without the need of installing gateways to provide network connectivity is a promising catalyst for new applications.
There are several aspects that have been taken into account in the security principles of LPWAN technologies. Confidentiality is provided in LoRaWAN by using a session key (NwkSKey) for encryption between the end node and the application server, while Sigfox does not uses encryption by default.
The end devices authenticate to the network according to different approaches. Sigfox devices are provisioned with a unique Network Authentication Key, that remains the same for the whole lifetime of the device.

Table 2.11. References about 5G.

5G	
General description	[78, 79, 80, 81]
Security features	[82, 79, 83]

LoRaWAN uses a different strategy: there are actually two different approaches used for the generation of the session keys. The first approach puts the session keys on the device during its fabrication, the second approach generates new keys at the beginning of every new session and it is considered the most secure. NB-IoT follows a a scheme similar to the second approach used in LoRaWAN, since a new key is derived for every session.
Integrity is provided in Sigfox using HMAC and all messages are sent using a Message Authentication Code. LoRaWAN calculates a MIC over the whole packet to protect the communication between the end node and the network server while NB-IoT uses a Cipher-based Message Authentication Code.
A common feature for all the three technologies is the use of a sequence number or frame number for providing data freshness and avoiding replay attacks.
In literature, a longer list of attacks on LoRaWAN has been studied, while the study of attacks on Sigfox has been very limited. Sigfox proved to be vulnerable to replay attacks because of the small size of the sequence number. LoRaWAN was subject to the same attack as well, in particular for ABP-activated end nodes, which do not update their session keys. A couple of attacks are common between LoRaWAN and NB-IoT. In particular the battery of a device using one of these two technologies could be drained exploiting different features of the different network technology. LoRaWAN and NB-IoT are also vulnerable to MitM attacks. NB-IoT does not suffer from interference because it uses licensed bands, while Sigfox sends the same message on three different frequencies in order to increase the possibility of successful reception.

2.1.9 5G networks for the Internet of Things

Since 2016, the 3GPP group has been standardizing the next generation of mobile communications with the aim of increasing network throughput and offering an ambitious infrastructure encompassing new use cases. 5G is an integration of new disruptive technologies to meet the ever growing demands of user traffic and emerging services. 5G will connect a high number of end-user devices that will require more security, with the goal of providing security in critical infrastructures [78]. The goals of 5G are to provide a high data rate (1-10 Gbps) with low latency (<10 ms) to low cost devices and sensors with reduced energy consumption.
For reaching a data rate of 1-10 Gbps, 5G must employ MIMO antennas and mm-wave technologies.
The radio architecture will need extreme spectrum efficiency, cost-effective dense deployment, and effective coordination to support the demands that 5G wants to address [79]. For dense deployment, relaying technologies could improve the scalability of network access operations to ensure the required coverage extension. In cells with a very high density of IoT devices it is possible to let them associate to different Relay Stations (RSs) so that the burden of network access is distributed among many nodes. The adoption of RSs can also improve the fault tolerance of the communication infrastructure, with the removal of single points of failure.
Recent studies suggest that mm-wave frequencies could be used to augment the currently saturated 700 MHz to 2.6 GHz radio spectrum bands for wireless communications. mm-wave carrier frequencies allow for larger bandwidth allocations, hence higher data transfer rates. The most promising bands in the mm-wave technology are sub 6, 28-30 and 38-40 GHz, unlicensed band of 60 GHz, and 71-76 and 81-86 GHz. The main concerns that need to be tackled in the mm-wave technology are high propagation losses, which call for high density of antennas, particularly in difficult environments like city centers, where reflections and fading pose a serious technical obstacle.
The 5G wireless technology uses MIMO antennas in the form of smart antennas which has the capability of beam tracking, tracing and spatial multiplexing. Transmission of large information without any interference, better efficiency and secured communication are the major requirements in 5G IoT and they can be achieved by increasing the number of an-

tenna arrays in the MIMO configuration. Both transmitter and receiver are equipped with MIMO antennas because they have the tendency of interference cancellation and better spectral efficiency.
The waveform introduced in 5G is based on the OFDM technology with some updates to that of LTE. Scalable and multiplexing numerology is considered as the best suitable waveform candidate for 5G NR. The major difference between 4G and 5G is that the value of the carrier spacing is fixed in 4G while it changes with the IoT service requirements in 5G.
In order to enable the connectivity required for many of the IoT applications, many features and functionalities will need to be added. This leads to a strong heterogeneous networking (NetHet) paradigm with multiple types of wireless access nodes with different MAC/PHY, coverage, backhaul connectivity, and QoS parameters. HetNets will offer the required seamless connectivity for the emerging IoT through a complex set of mechanisms for coordination and management.
The current traditional networks cannot enable the growing networking technologies demand for future next generation networks. To overcome these problems, emerging technologies including Software-Defined Networks (SDN), Network Function Virtualization (NFV) and Cognitive Radio (CR) are among the network enablers proposed [80].
SDN separates the data plane from the control plane, centralizes network control into a logical entity and, enables programmability of the network. SDN will be able to address flexibility and interoperability challenges of future multi-vendor, multi-tenant 5G scenarios and will simplify network design, management, and maintenance in heterogeneous networked environments with different QoS requirements. NFV is a complementary technology of SDN which aims to virtualize a set of network functions. By moving network functions from dedicated hardware into general purpose computing/storage platforms, NFV technologies will allow the management of many heterogeneous devices [81]. CR is a key technology to utilize the limited and scarce spectrum resources. CR supports the capability of sharing the licensed spectrum in an opportunistic manner and operating in the best available channel.

Security

5G security requirements need advanced network authentication and key management procedures [82]. The security solutions used in previous gen-

erations will not suffice for 5G, because of the new conflicting services and technologies. The three basic principles for secure 5G systems are flexibility, automation, and built-in security. Security by design will help to handle all the very diverse technologies employed in 5G, while automation will come in need in all the situations in which threats and controls vary. The 5G security architecture has been defined by release 15 of the 3GPP technical specification and it divides security features into separate architectural components [79]. The SDN architecture supports highly reactive and proactive security monitoring, traffic analysis and response systems to facilitate network forensics, the alteration of security policies, and security service insertion. Security systems such as firewalls and Intrusion Detection Systems can be used for specific traffic by updating the flow tables of SDN switches. 5G needs proper communication channels security to prevent security threats and to maintain the additional advantages of SDN: IPSec is the most used security protocol to secure the communications channels.
The mutual authentication between the UEs and the network is performed using AKA, which was first used for authentication in GSM and it is still considered as the most viable mechanisms for authentication and authorization in 5G networks [79]. AKA is based on symmetric keys and runs in SIM. Extensible Authentication Protocol (EAP)-AKA method for 3G networks was developed by 3GPP to support identity privacy and fast re-authentication. For communication through trusted non-3GPP access network, the UE is authenticated through the AAA server using EAP-AKA. The 5G AKA Protocol consists of two main phases: a challenge-resposnse and an optional re-synchronization procedure. The EAP-AKA' protocol is very similar to 5G AKA: it relies on a challenge-response with a shared secret and the SQN for replay protection as well, but some key derivation functions are slightly changed. Authentication and key management procedures in 5G are differentiated in primary authentication and secondary authentication. Primary authentication defines the mechanisms that permit UEs accessing to the Serving Network domain. Secondary authentication defines how to access Data Networks (DNs) outside of the cellular infrastructure. The authentication in 5G Release 15 is based on new versions of the AKA protocols, notably the new 5G AKA protocol. The specification 3GPP TS 33.501 [83] describes how authentication and key establishment

are achieved in the 5G ecosystem. The three components involved in the cellular network architecture are subscribers, made of the combination of UEs and Universal Subscriber Identity Module (USIM), Home Networks (HNs), containing a database of their subscribers and responsible for their authentication, and Serving Networks (SNs), to which UEs may attach to when they are in a different location from the their corresponding HN. The USIM has cryptographic capabilities and stores a unique and permanent subscriber identity, the public asymmetric key of its corresponding HN, a long-term symmetric key, and a counter called Sequence Number (SQN).

2.2 Attacks to IoT network technologies

2.2.1 DoS and battery draining

BLE

Scapy [100] is a Python suite used for manually crafting packets and to perform attacks on Bluetooth. A DoS attack can be easily implemented by sending large quantities of pairing request packets, that devices are not able to handle. A simple Python program can be written to generate source addresses and send spoofed pairing requests. Scapy is used by the authors of [86] to perform a DoS and a Replay attack. BLE devices are able to send small amounts of data at a time. Sending large quantities of pairing request packet, it is possible to drain the battery of a device, prevent legitimate users from pairing and ultimately crashing the device.

802.15.4

It is possible to perform a DoS attack on a 802.15.4 network when it uses the AES-CTR suite [29]. The 802.15.4 specification does not include any integrity or confidentiality protection for acknowledgement packets. An adversary can forge an acknowledgment for any packet with the appropriate sequence number, which is sent in the clear so it is not difficult to retrieve. This weakness can be combined with targeted jamming to prevent delivery of selected packets. The attacker can transmit a short burst of interference while the packet is being sent, causing the CRC to be invalid at the recipient and the packet to be dropped. The attacker

can then forge a valid-looking acknowledgment, fooling the sender. This vulnerability renders acknowledgments untrustworthy and should not be relied upon for secure applications.
The replay-protection attack is another kind of DoS attack [26]. In the 802.15.4 specification, replayed messages are prevented by the replay protection mechanism. A receiver checks the recent counter and rejects the frame which has the counter equal or less than the previous obtained counter. An adversary can easily launch a replay-protection attack by sending many frames containing different large frame counters to a receiver which performs replay protection and raises the replay counter up as the largest frame counter. When a normal station sends a frame with a reasonable size of frame counter that is smaller than the replay counter maintained at the receiver, the frame will be discarded and the service will be denied.

Z-Wave

The vulnerabilities found by the authors of [33] allow to perform a DoS attack not detectable through jamming detection. This attack was possible by blocking the central node without using a jammer. This attack can be repeated to block the gateway constantly and is traceable only if the victim has a Z-Wave sniffer active. However the source of the attack can not be determined since the messages look like they have been sent from the gateway. At the end of the attack, the gateway returns to normal operations. This attack has been acknowledge by the proprietaries of the protocol and a new Z-Wave implementation and specification have been made available.

ZigBee

A ZigBee network is vulnerable to DoS attack if the message integrity is not verified, even if the communicated messages are encrypted. In the DoS attack, the attacker composes a message that includes a random content as the encrypted payload and sets the frame counter to the maximum value, the attacker sends the message to the victim device, which decrypts the payload to a random meaningless plaintext. Since the attacker has set the frame counter to the maximum value, any legitimate frame arriving

after the attack will be automatically rejected by the victim.
Another way to perform a DoS attack is by repeatedly jamming the medium during Contention Access Period (CAP) and Contention Free Period (CFP). In this way, a victim can be put on endless retransmission loop, draining its battery [43].
Cao et al. [87] investigate a flaw related to sending security headers in clear text. Their work analyzed an attack called *ghost-in-Zigbee*, in which an attacker constructs bogus messages to lure a node to do superfluous security-related computations to deplete its energy. The consequences of the attack will facilitate the execution of other attacks such as DoS, replay and loss of confidentiality.
Vidgren et al. [88] propose an End-Device Sabotage attack, in which a special signal makes a ZigBee end device wake up constantly until the battery runs out.

LoRaWAN

Class B devices periodically wake up to wait for any incoming messages during extra receive windows, specified by beacons broadcast by the gateway. Specific class B vulnerabilities allow a malicious actor to drain the battery of the field devices [84]. The beacons broadcast by a gateway to provide a time reference to the end devices are not encrypted nor protected against malicious modification. As the content of the beacon payload is public knowledge, an attacker can send out a beacon with malicious parameters that would be received and processed by the end devices. If the attacker can create his own beacons, it is possible to define random wake up time values to disturb the downlink operation of the device, that would wake up at different times than expected by legitimate gateways, or very frequently, increasing the power consumption of the sensor.
The main cause of attacks in class B devices is the fact that beacon frames are not properly protected and lack an integrity check value. Transitioning from a PHY CRC to a MIC, would solve the problem of any malicious beacon modification. Another solution would be the introduction of a cryptographic signature in place of a PHY MIC. This would allow a single beacon to be cryptographically verified by all end devices.

NB-IoT

Some NB-IoT applications usually run on an unmonitored network, so attacks could easily go unobserved. In such scenarios, if the NB-IoT base stations do not control the amount of size of resources requested by the end nodes, the attacker can trigger and monitor the resource allocation and consume all the available resources to perform a DoS attack. The Tracking Area Update reject messages are not protected, so an adversary can spoof such a message to block the access to the network to a target device provoking a DoS attack and temporarily blocking mobile devices [85]. If a rogue base station replies to an incoming connection with an Attack Reject message, it can fool the mobile device to believe that it is not allowed to connect to the network. The rogue base station can prevent any mobile device in its range from connecting to the network, resulting in a DoS. This is just a temporary DoS, solvable by rebooting the device in order to connect to the network again. LTE mobile devices then implement a timer which is started when an Attach Reject message blocks the device from connecting: this timer is configured to a value between 24 and 48 hours. However, the loss of connectivity for 24 hours could be critical.
A rogue base station could also indicate a victim end node that it is not allowed to access 3G and LTE services and trigger a downgrade of the connection to GSM, which is known for being more insecure. The target device will then try to connect to the GSM network. An attacker could combine this with a rogue GSM base station, eavesdropping all mobile network traffic. The attack is not possible with devices not supporting GSM.
Another possible attack in NB-IoT is battery draining: an attacker might force a UE to receive and send data (e.g. pinging) the victim, leading to a permanent DoS. A LTE smartphone provided with a NB-IoT SIM card can be misused and can connect to a NB-IoT APN and then used to create a Wi-Fi hotspot in order to allow a PC, source of the attack, to join a private APN network. Coman et al. [65] do not perform any malicious activity since this setup included a real network, but they showed how a ping scan is possible. Using the Zenmap tool, it is possible to scan the subnet to which the LTE UE is connected and then scan for open UDP and TCP ports on other devices on the network. Another way to drain the battery the device is by decreasing the sender's SNR. If a malicious device sends data when another node is transmitting, the SNR decreases and the

legitimate sender is then forced to increase its transmission power and it then finishes its power.

2.2.2 Eavesdropping

BLE

BLE devices stay on a particular channel only long enough to transmit a single packet. The time spent on each channel and the hopping sequence varies from connection to connection. To sniff a connection an attacker must know the hop interval or dwell time, the hop increment, the access address and the CRC init. The authors of [89] built a sniffer on the Ubertooth platform. The Ubertooth radio chip can monitor BLE channels and converts RF in a bitstream. The approach used by the authors is able to tune to a single BLE channel: this approach has tighter timing requirements, but it is deployed on a cheap hardware platform. The RF energy generated by a BLE device when transmitting a packet must be sniffed and then converted in bits. The start of a transmission is identified by searching for the fixed value of the 32 bit access address. The AA used on a data channel is exchanged during the connection setup. The start of transmission, identified by access address, defines the byte boundary of the message and the bits are then converted into a sequence of bytes. In order to follow the BLE connection it is necessary to hop along the same channels as the master and slave. The channel hopping sequence is very straight-forward and for every connection a hop increment is defined. The next channel is calculated by adding the hop increment to the current channel. The master and the slave hop to the same channel at the same time and the sniffer hops to the same sequence of channels. The four parameters needed to know to follow a connection (hop interval, hop increment, access address and CRC init) can be extracted either from the connection initialization packet or they are recovered by exploiting properties of BLE packets. The hop interval is measured by measuring the time between two consecutive packets. The hop increment is recover by measuring the interarrival time of packets on two data channels. After all the four parameters have been identified, it is possible to use following mode and follow a connection.
As reported in [86], Ubertooth One is a powerful card designed to sniff BLE traffic. It attempts to listen to traffic broadcast across the entire

frequency spectrum allotted for BLE communications, as opposed to other Bluetooth chips that can only pay attention to one channel at a time. Ubertooth is able to capture Bluetooth classic packets as well as BLE packets from all the devices in range. Other alternatives to sniff BLE communications are Bluefruit, another USB dongle that is able to capture only BLE communications, or other dongles provided by Texas Instrument and Nordic Semiconductor. The scan performed by Ubertooth works by first listening to advertisement packets from visible devices and then sniffing for powered on but invisible devices. Ubertooth supports different options for scanning and capturing. It scans the area to find any BLE device: at first it listens to advertisement packets from visible devices and then to invisible devices. Raw data capture form Ubertooth can not be opened but must be decrypted. There are two ways to decrypt a capture. First, if the Ubertooth capture contains a pairing event it is easy to brute force the TK. Once the TK is discovered, it can be applied to the same file or to additional captures of the same communication stream to uncover the LTK. It is also possible to brute force the LTK without these data points, but it would take a long time. Wireshark supports decryption of Bluetooth encryption if a LTK is given and can be used to analyse the data.
In [18] the authors used a Nordic Thingy 52 with BLE 4.0 to perform attacks. The Nordic Thingy 52 is a compact multi-sensor device that collects environmental data of various types, such as temperature and humidity, and sends it to the Nordic Thingy mobile application. Since it does not have I/O capabilities, it uses Just works as pairing mechanism. Furthermore, it does not have any authentication for user verification, so anyone with the mobile application can connect to the device. The sensor values can be read directly in the app or by using a packet analyzer tool. The attacks performed by the authors were implemented using an open source tool named BtleJuice. This tool clones the original device and advertises very frequently so that the user can first see the cloned device with the same name and characteristics of the original device. It is also possible to perform active interception, in which the attacker can alter the data transmission between the user device and its application. The ability of an attacker to modify data can lead to many MitM attacks, depending on the type of data and to the actions that can be made. BtleJuice allows to receive the original values and send them altered or unaltered.

Z-Wave

Fouladi and Ghanoun [90] exploited an implementation vulnerability in the key exchange protocol to unlock certain door locks. The authors demonstrated an implementation vulnerability in Z-Wave key exchange protocol that could be exploited to take full control of a target Z-Wave door lock by only knowing the Home and node IDs of the target device, which can be identified by eavesdropping the Z-Wave network traffic. This vulnerability was due to an implementation error in disabling the use of temporary key after initial network key exchange during inclusion of a node to the network.

ZigBee

Vidgren et al. [44] address the problem of Network Key Sniffing. When the network key is transported using the Standard Security level, the TC sends the current network key unencrypted over-the-air to the devices that want to join the network. An attacker can intercept the network key and use it for eavesdropping and attacking purposes. This vulnerability can be solved by removing the Standard Security level form the ZigBee specification or by installing the Network Key. However the better option is to use the High Security level in safety-critical systems.
Another attack that works against ZigBee is the same-nonce attack [44]. When a nonce is used as a part of the AES-CCM*, encrypting the same plaintext twice will result in two different ciphertexts, because the nonce will be different. If the ACL provides, for any reason, the same nonce, an eavesdropper can recover partial information regarding the plaintexts. The eavesdropper makes sure that the nonce and the security key are the same, in order that the XOR of these two ciphertexts will be the same as the XOR of their corresponding plaintexts. The Same-Nonce attack can be successfully performed by causing a power failure, resulting in a clear ACL. If the last nonces are unknown, the system resets the nonce status to the default value: the chance of reusing the same nonce increases and the system becomes vulnerable to the Same-Nonce attack. A simple countermeasure would be to store the nonce states in a Non-Volatile Memory, and recovery.
Olawumi et al. [43] propose two additional practical attacks against Zigbee,

conducted in a laboratory environment. The first attack is network discovery and device identification, which is based on the fact that the ZigBee networks within range and the configuration of legitimate devices can be discovered, during the phases of network discovery. Network discovery is a major weak point in ZigBee because if an attacker can perform a network discovery attack and obtain important information, all the other attacks are made easier. By mimicking the ZigBee discovery process, it is possible to collect this vital information and to perform interception of packets, which is the second attack described by the authors. This second attack is made possible since most of the ZigBee networks do not use encryption and it is possible to eavesdrop sensitive information. The authors also propose countermeasures against these attacks. One solution could be using some kind of intrusion detection for continuously monitoring. The most immediate countermeasure against interception of packets is to pre-install the network key when using the Standard Security level. Another option is to use High Security level in safety-critical ZigBee-enabled systems, because in this case the network key is not transmitted unencrypted.

2.2.3 Packet forging and manipulation

802.15.4

Message manipulation attacks are used to inject false data into the network by modifying a legitimate data frame with information chosen by the adversary. This kind of manipulation can be achieved using one of two different techniques. The first is symbol flipping, in which the attacker emits waves synchronized with the original signal that could be combined to form a new signal containing the false information. The second is signal overshadowing, which has tight timing and phase synchronization requirements and in which only the stronger of two signal is received [25].
Another possibility is forging the ACK frames, which do not have integrity protection. The eavesdropper can forge the ACK frame by using the unencrypted sequence number from the data frame. The adversary can send a forged ACK frame fooling the sender that the receiver successfully received the frame. A sender can not be sure if the received frame is coming from the receiver or another node [26].

LoRaWAN

The requirements of this kind of attack are either an unauthenticated and unencrypted connection between the gateway and the NS, or for the attacker to own a malicious gateway and connect it to the victim's NS. The attacker can forge any packet and force the NS to accept it and forward it to the Application Server. The decrypted payload of the modified packets will be meaningless since the attacker does not know the AppSKey, but this attack could DoS the application. The attack implemented by Coman et al. [65] aimed to test whether a LoRaWAN packet with the same FCntUp field but different Frame Counters (and thus different MICs) would be accepted by the network server. The authors showed that packets can be forged in some situations, forcing the application server to receive a packet with garbage payload and possibly causing a DoS. The application and network servers should then disallow the communication from the devices sending invalid packets to prevent DoS.
A bit-flipping attack is a method of an attacker which can change specific fields on ciphertext without decryption. If an attacker wants to modify specific fields, it is necessary to modulate bits in the same positions of the targeted ciphertext. A bit-flipping attack is possible because the integrity between the network server and the application server is not protected. If the attacker captures traffic, the application server can not detect if the message is from the attacker or the network server. If an attacker gains access to a network server, he can eavesdrop on the communication between the network and the application server [91]. The solution to avoid a malicious change of the payload content is to run the integrity check value at the application server and not at the network server. One strategy to secure the integrity between the network server and the application server is to check the MIC again when the message arrives at the application server [92].
The CTR mode used in LoRaWAN simply performs the XOR operation to encrypt a plaintext and does not shuffle the order of the bits in the plaintext [93]. LoRaWAN messages are encrypted and equipped with a MIC which can counteract a message-modifying attack. The payload encryption using the AppSKey is undone by the application provider, while the cryptographic MIC on the payload data and header information is checked by the infrastructure provider. Between the network server and the application

server, the content can not be checked for integrity and authenticity. The link between a LPWAN operator and a third party application provider is typically run over a public network such as the Internet, and the messages between the two parties could be altered in content or rerouted. The attack is possible if the attacker can insert himself between the LoRaWAN operator and the IoT solution provider, using one of the possible solutions such as routing-based approaches or physical and link-layer based attacks. The obvious solution to avoid a bit-flipping attack is to run the integrity check at the application server and not the network server. A more radical but better approach would be to repurpose protocol fields which would require some changes to the standard and firmware updates. The method proposed against bit flipping attack by Lee et al. [93] prevent attackers from recognizing positions of original fields. The method is performed by end devices and performs two phases. The first is the shift phase (the end device performs circular shift of frame payload to the left), the second is the swap phase (swap the positions of the shifted octets in the shift phase). The end device marks the location of the swapped octet into a swap table. The end device checks swap table and performs swap except for already swapped octets. In shift phase, the constant can be varied, so it is difficult for the attacker of the outside to predict the moving distance.

2.2.4 Replay

ZigBee

It is possible to practically conduct these attacks in ZigBee using KillerBee tools [43]. In order to defend against Replay attack, the timestamping mechanisms should be integrated into the encryption process of ZigBee, in order to discard retransmitted frames. Replay attacks can also be prevented by using a 32-bit frame counter: the frame will be discarded if a frame counter value is lower than the current value in the memory of the recipient. This is an efficient way to prevent attacks in theory, but in practice the ZigBee specification does not define when the TC has to update the network key. If the network key is not updated in time, it will lead to a deadlock in the network when the frame counter reaches its maximum [44].

Sigfox

Sigfox is vulnerable to replay attacks due to the small size of its SN, allowing the attacker to inject previously sent messages into the system. The 12-bit Sequence Number allows for $2^{12} = 4096$ messages. After 4096 messages, it overflows back to 0, and an attacker can replay any of the previous 4096 packets indefinitely. There is a maximum allowed gap between the SNs of consecutive Sigfox frames before packets will be dropped, so end-nodes can be DoS-ed, even without the intervention of an attacker, if they run out of coverage for a long period of time.
Coman et al. [65] observed the effect of message replay using a controlled setup, in which the SN was set to 0 and the end device was in a Faraday cage and periodically taken out of it, in order that only some packets could reach the Sigfox gateway. After the SN was reset to 0, the attacker was able to send one of the previously captured frames, which successfully reached the backend.

LoRaWAN

The attacker receives and transmits data exchange between two trusted parties. The malicious actor can capture and store a duplicate genuine request to a service. Services that can be accessed only by authenticated users can be tricked by utilizing handshake message or old data [92]. The replay attack depends on the fact that the NwkSKey and AppSKey are used as used as long-term key material and are not restricted to a single session. End devices should then physically secured to prevent a malicious system reset.
The ABP-activated end devices use static keys which are preprogrammed into the device. After resetting, an ABP-activated end device will reuse the frame counter value from 0 with the same keys. In this case, an attacker can grab messages in the last session with larger counter values and reuse it in the current session [84]. Another method to restart the counter is a counter overflow: the counter reaches its maximum and resets to 0, thus an attacker can replay messages with counter value from the previous session and the same session keys and cut off the communication between the end device and the server. This is a vulnerability for both ABP and OTAA, however attacking an ABP-activated end device will take less time. A mes-

sage replay is very easy to implement since an attacker can store the uplink messages of a node and wait until the device resets the counter value FCnt. The adversary must then send a message whose frame counter is accepted in the range (current frame counter; current frame counter + gap), where gap is the maximum accepted counter gap. Once the attacker gets the largest possible counter value for one end-device, it can periodically replay the message and block the end device permanently or until the session keys are changed, which requires a separate channel or a physical access.
In order to prevent a replay attack, the ABP method should not be used, or new keys should be downloaded periodically. End devices should also be physically protected in order to prevent a malicious party to initiate a system reset and to reduce the attack surface. In fact if an attacker can not reset the counter, the only way to perform this attack is to wait for a counter overflow. The end device may also be required to go through the OTAA activation procedure again everytime the counter reaches its maximum value in order to obtain new session keys. To prevent the malicious replay of an ACK, it is possible to add a cryptographic checksum with the returned acknowledgment. As the ACK is bound to one specific message, it is not possible to cache and resend ACKs.

2.2.5 Man in the Middle

BLE

A MitM attack happens when an adversarial node sneaks in between the Personal Area Network (PAN) nodes to read or insert fake messages. For example, this attack occurs in BLE when a user wants to connect two devices but, instead of connecting with each other, they connect to a third device which relays the information between the two legitimate devices. All the information between the two devices is then compromised. This attack is effective when the devices use unauthenticated connections, enabling the attacker to intercept, modify or inject information or commands. Several tools have been developed for implementing MitM attacks (GATTacker, BTLEjuice) and two attacks that leverage this concept are rogue device attack and relay attack [20]. In a rogue device attack, an attacker impersonates a target device, so that the device that wants to communicate with the target device is actually communicating with the malicious

device. The majority of applications do not properly authenticate with devices before sending commands so it is possible to clone the target device and send advertisements. The user application initiates a connection after it receives the cloned device advertisement. The user application then sends the cloned device commands, passwords and nonces, which can be used by the attacker to gain access to the target device. The relay attack is similar to the rogue device attack but it is designed for scenarios where the nonces are truly random. In this attack, an attacker impersonates a target device and forces the user to communicate via a bridge to the attacker's device. This enables the attacker's device to impersonate the target device and trick the user into communicating with the attacker. The danger of a relay attack is that a user can be anywhere as long as a rogue device is nearby to impersonate the target device. Secure simple pairing methods offer protection against MitM attacks [101].
Geofencing and BLE Guardian protect against MitM attacks [20]. Geofencing protects against unauthorized access by requiring a user to be within a specific distance of designated GPS coordinates in order to request credentials from a web server. A virtual fence is set around a device, a user must be within a set distance to gain access. Geofencing prevents cloned devices from tricking users into providing their credentials. BLE Guardian protects user privacy using an administrative program to control which entities can discover, scan and connect to a device. BLE Guardian controls advertisements packets: these protections are required because true advertisement packets are shielded from passive eavesdropping. The implementation of BLE Guardian requires an Ubertooth One in addition to the device to be protected. No additional implementation is required by the device manufacturer and this approach could be used in conjunction with other techniques to enhance security.

Z-Wave

A MitM attack can be executed in different forms, for example by deleting all the traffic received or by forwarding only a selection of the captured messages [102]. A black hole attack is a frame dropping attack where a node under the influence of a Black Hole Node (BHN) attacker silently drops applications frames when it is expected to forward them. Active Black Hole attacks exacerbate frame loss by manipulating topology

state information in the network to increase the number of routes that flow through the BHN [37].

LoRaWAN

Thomas et al. [94] presented a MitM attack in LoRaWAN. The attacker can use a SDR module to capture the raw frequencies travelling in the wireless medium. The testbed employed by the authors included two Raspberry Pi modules on which the LoRa modules were configured for the peer to peer communication, a HackRF One SDR module used to capture the data transmitted in the particular LoRa frequency range, two Lora SX 1272 modules by Semtech and LoRa communication packages. During the peer to peer communication between the LoRa modules connected to the Raspberry Pis, the attacker will setup the SDR, analyse the LoRa frequency range and capture the transmitted packets. The raw LoRa packets are converted in order to get the packet format of the data. Wireshark is then used to decode the packet structure and get the transmitted data between the two modules.
The defence proposed by Thomas et al. [94] for MitM attack in a peer to peer communication is a modified cryptographic counter mode algorithm. The Counter mode just performs the XOR function with the plaintext to give the ciphertext using AES. In this situation, it is very easy for the attacker to get the plaintext. The authors devised a method to protect the encrypted communication between the nodes from the attack by implementing a cryptographic Galois Counter Mode of encryption and decryption, in order to protect the payload from getting modified. The Galois Counter Mode is constructed from an approved symmetric 128 bit key block cipher. GCM offers personal data authentication security using an all-in-one hash function specified across the Galois finite binary field. The operations in GCM rely on the underlying block cipher symmetric key. The block cipher operation mode consists of two functions for each key: Authenticated Encryption Function that encrypts the confidential data and Authenticated Decryption Function that takes the initialisation vector and the ciphertext as input and carries out the decryption process by verifying the input given to the function and checks whether the authentication tag has changed or not. When the attacker performs the MitM attack, he captures the payload transmitted and tries to modify it. By implementing

GCM of encryption and decryption in the LoRa communication, the value of authentication tag gets modified when the data is changed and the decryption goes wrong. Thus, the receiver gets notified with the incorrect decryption.

NB-IoT

The lack of computing power on NB-IoT devices limits the use of cryptographic algorithms. If the Diffie Hellman exchange update key is used, the overall exchange process cannot be authenticated and is vulnerable to MitM attacks [95]. A possibility to perform a MitM attack in NB-IoT comes during the initial attach procedure, when the user identity transference is non-encrypted. This known weakness could be exploited to launch a MitM attack by user IMSI (International Mobile Subscriber Identity) impersonification.

2.2.6 Wormhole

LoRaWAN

A wormhole attack is an out-of-band connection between two IoT devices that can be used to forward packets faster than via typical paths. Using a low SF in LoRaWAN decreases the airtime of a message, so that the time the jammer has to reached is decreased. On the other hand, lower SFs have lower range and require less power output from the jammer to be disrupted [92]. In a wormhole attack, packets can be captured and never reach the gateway or can be replayed any time [63]. Important alarm messages can be captured in order not to reach the gateway or could be sent to the the gateway in situations where there is no alarm. Since there is no time-related information in LoRaWAN messages, it is hard to detect this attack.

NB-IoT

In an attack also known in literature as "shared node attack", a malicious node claims to provide better transmission path to send data to the destination: the attacker can then use this node to capture packets [73].

2.2.7 Impersonation and rogue controllers

Z-Wave

The goal of Rouch et al. [36] was to create a universal controller from an off-the-shelf controller. The operations carried out were: associate the adversary controller with all the possible nodes, capture the Home ID of the targeted network to be used by the adversary controller and restart the adversary controller. After the controller is restarted, it scans the target network. The problems that the authors had to overcome were the impossibility to assign a Home ID to the adversary controller and to send messages to the nodes not paired with the controller. A countermeasure to this attack would be to enable the secure mode with encryption to prevent the malicious controller from taking over a node paired in secure mode. However manufacturers implementations of the secure mode may introduce flaws.
Z-Wave gateways can also be vulnerable if consistently connected to the Internet [96]. Managing the network through a globally connected gateway provides the user with the ability to control the network with a mobile device from anywhere. New vulnerabilities arise because of this accessibility, and the WLAN defense can be compromised in order to gain access to the Home Automation Network (HAN). The attacker can gain access to the WLAN by associating with an unsecured network or using one of the available tools to penetrate its defense. Z-Wave gateways are typically susceptible to attacks once an attacker compromises the WLAN through physical access, lack of network authentication or network key compromise. When a Z-Wave gateway connected to the Internet is compromised, point-to-point message encryption becomes irrelevant.
A vulnerability found by the authors allows the injection of a rogue controller into the network that maintains communication with the devices. An attacker can gain access to the WLAN and copy the Z-Wave HAN configuration. This will allow persistent access and control of devices even if the user decides to remove the gateway from the Internet. It is possible to copy all the information and configuration from a gateway in inclusion mode. In this way, a rogue controller has been added to the network. Several mitigation strategies have been proposed by the authors. Hiding the SSID prevents attackers from using passive scans to locate the network.

MAC address filtering allows the WLAN administartor to select specific devices that are trusted on the WLAN. If MAC filtering is enabled, an attacker without knowledge of spoofing MAC addresses will not be able to authenticate to the WLAN. A Reverse Proxy Server (RPS) is an intermediary application between the user and the gateway and it provides additional authentication before allowing the user to access the gateway. Z-Wave devices trust the source and destination fields of the MDPU frame [37]. It is the easy to impersonate frames originating from the controller or another device. Devices using the Z-Wave security layer have some protection against outsider impersonation. Devices specify which command messages must use the secure frame. Secure frames are signed and encrypted using keys exchanged during network inclusion. An outsider who is not in possession of the authentication and encryption keys is unable to transmit a valid secure frame. The outsider is then unable to impersonate the origin of a command message if the destination requires that it is sent in a secure frame. However the outsider could perform impersonation attacks on the device using commands from a supported command class that does not require the use of the security layer.

ZigBee

ZEDs rely on a coordinator to remain awake and receive data packets. When a ZED wakes up from a sleep mode, it sends a poll request for any available data to its coordinator. In the ZigBee End-Device Sabotage attack proposed by Vidgren et al. [44], an attacker impersonates the ZigBee router or Coordinator in order to abuse the poll requests of legitimate ZEDs. The attacking device sends broadcast or multicast replies to all poll requests of legitimate ZEDs, thus keeping them awake all the time. Based on the default polling rate of 100ms and the power consumption, the attacker can cause power failures to ZigBee sensors and actuators. A practical countermeasure would be to use a remote alerting system for warning about power failures of devices. Another countermeasure would be to configure the legitimate ZEDs in a cyclic sleep mode that allows modules to wake up periodically for checking data.

2.2.8 Jamming

802.15.4

Jamming is a physical layer attack that is launched with the intent of creating a DoS against the network. The attacker emits radio signals in order to decrease the SNR of ongoing radio communications, disrupting the reception of messages. Radio jamming could be performed following different approaches, according to the band occupied and to the number of channels targeted. The mission of disturbing signal could be constant and continuous or more infrequent [25].

LoRaWAN

LoRa devices suffer from coexistence issues and simultaneous LoRa transmissions can meddle with each other. This weakness permits attackers to utilize Commercial-off-the-Shelf LoRa devices to jam LoRa networks [59].
An attacker with malicious intentions can flood LoRa messages at a certain frequency to clean out all the transmissions in that frequency [92]. Jamming attacks could be pointed to different layers of the OSI model: Physical layer, where the malicious actor assign assign any wideband signal with a higher SNR than the user; MAC layer jamming, where the malicious actor jams explicit pieces of the message [65]. The jamming of an entire network can be detected since all the devices that communicate in that frequency would abruptly start to drop out. Some techniques that should be used are creating dense LoRa networks since jamming is more complex with the presence of various gateways, maximize the utilization of channel hopping and moving to a higher spreading factor.
Jamming is a serious threat to LoRaWAN: malicious entities can transmit a powerful radio signal in proximity of application devices and disrupt radio transmissions [63]. Anyone with malicious intent can flood LoRa messages at a certain frequency to wipe out all the transmissions in that frequency. It is possible to jam LoRa messages using well timed malicious transmissions. LoRa in fact suffers from coexistence issues and devices which send data simultaneously using certain frequencies can corrupt each other's signal. Jamming a whole network can be easily detected, because all the devices should perform a joining procedure again, but it is also

possible to use a selective jamming approach, classifying messages as they are on air [97]. There are three different techniques to perform jamming on the gateway and prevent communications coming from an end device to reach a gateway. A selective jammer can systematically jam a particular type of message or all the messages coming from a particular device.

1. Triggered jamming: LoRa modules have the capability to scan a certain channel to detect whether there is an ongoing transmission. This capability is not required by the protocol and can be abused by attackers to detect activity on the channel. When a LoRa transmission is detected, the malicious device can start transmitting in order to jam this transmission. This vulnerability allows malicious entities to use off-the-shelf devices to increase packet loss in a network. Triggered jamming relies on detecting preamble symbols and then jamming the device without demodulating or decoding any part of the signal.

2. Selective jamming: a more sophisticated and efficient jamming technique. Triggered and continuous jamming affect all the devices on a certain frequency, so they are easy to detect and it is possible to take action, for example by changing the communication frequency or enabling frequency hopping. Selective jamming only jams selected devices or selected messages, and since other devices or messages are not affected, it is difficult to detect the jamming. This attack is suitable for preventing a specific event from being communicated to the gateway. Selective jamming requires to read some parts of the message before deciding whether to jam or not. This technique has strict timing requirements and the time remaining to successfully jam a packet is shorter.

3. Combination of selective jamming and a wormhole attack: this attack needs two devices, a sniffer and a jammer. The sniffer receives messages and decides whether to jam or not. If it decides to jam, it informs the jammer and keeps listening to the original transmission and stores it for later use in a replay attack. The two devices have to be kept far enough apart so the jammer does not jam the sniffer. Ideally the sniffer is close to the end device while the jammer is close

to the gateway. This approach is powerful when high SF communications are involved. By using a triggered or selective jammer close to a device to block join attempts with that gateway, it is possible to force a device to join a more distant gateway, forcing it to move to higher SFs and enabling this selective and wormhole attack to operate.

Aras et al [97] also implemented two of these techniques.

- Selective jamming: a jammer needs to detect a LoRaWAN packet and to start receiving it. It then aborts receiving if the received content triggers the jamming policy and immediately jams the channel. During the attack, the radio module starts in receiver mode, and once a message is detected, it starts to write data to its FIFO buffer starting with the message type and device address. The FIFO buffer is then read byte by byte and once enough bytes are read, the jamming policy is applied. If the message has to be jammed, the radio module switches to jammer mode.

- Combination of selective jamming and wormhole attack: this attack requires two devices, a sniffer and a jammer. Between the two devices it is necessary to have a low latency link, and the authors used UDP communications over Ethernet to connect these two devices, using Arduino shields or a Raspberry Pi.

In the tests performed by the authors for the second technique, the messages with SF7 were not jammable, while the jamming was successful for SF11 and SF12. The jamming attacks presented rely on two characteristics of LoRa messages: the long air-time of these messages and the possibility of drowning out legitimate messages with jamming messages broadcast with more power.
It is possible to reduce the effectiveness of jamming techniques using the capabilities of LoRa [97]. To beat a jamming attack, it is recommended to create dense LoRa networks with overlapping coverage regions, since the jamming is more complex in the presence of multiple gateways. Other possible mitigations are maximising the use of channel hopping, so that the jammer must listen to multiple channels, and use low SF and small packet size to beat jammer reaction time. It is also possible to perform traffic

analysis and identify variations in the pattern of incoming messages and to trigger alarms or adaptations to the network. It is possible in fact to detect changes in traffic patterns if the traffic analyser knows the sending rate of the end devices.

5G

5G will accomodate a huge number of user devices and smart things, that will send and receive data simultaneously, practically jamming the radio interfaces [79]. Malicious nodes sending excessive traffic to cause DoS attacks could worse the situation.

2.2.9 Privacy leaks

BLE

There are possible privacy leakages when using BLE devices. Fitness trackers for example only periodically connect to exchange data, while for most of the time they are in advertising mode, broadcasting messages announcing their presence. The BLE standard also outlines the use of randomized addresses for prevent tracking, however this is not followed by major manufacturers [98]: fitness trackers for example use unchanged BLE addresses that can be tracked. Randomizing the advertised address is a solution to prevent user tracking based on advertising packets. However, usage of randomized addresses can lead to the smartphone not being able to identify the BLE fitness tracker to which has already been paired and the user experience could be damaged. BLE tracking can provide information on the location of a device due to its small range and also on the user's activity, since it is easy to detect whether an user is moving or sitting according to the traffic exchanged. It is then possible to identify a user from some BLE traffic attributes.

5G

The major privacy concerns could arise from data, location and identity. At the physical layer level, location privacy can be leaked by access point selection algorithms in 5G mobile networks. Such attacks can also

be caused by setting up a fake base station which is considered as a preferred base station by the UE. Furthermore, 5G networks have different actors such as Virtual MNOs, Communication Service Providers, and network infrastructure providers. Mobile operators thus do not have direct access and control of all the system components and will rely on these new actors. User and data privacy are seriously challenged in shared environments where the same infrastructure is shared among various actors [78].
Another threat comes from the paging protocol, which allows devices to periodically poll for pending services during their idle, low-power state. The 4G/5G protocol fixes the paging occasions, the moments when a device can poll for services. The fixed nature of paging occasions can be exploited to associate the soft-identity of a victim (e.g. its phone number) with its paging occasion, through an attack named ToRPEDO. An adversary can retrieve the personal identity of a device with a brute-force IMSI-Cracking attack while using ToRPEDO as an attack sub-step. Once the attacker knows the paging occasion of the victim, the attacker can mount a DoS attack by injecting empty paging messages, thus blocking the victim from receiving any pending services. For defending against ToRPEDO, the authors of [99] design and evaluate a countermeasure that adds noise in the form of fake paging messages for perturbing the underlying paging message distribution.
IMSI catching is also a possible attack, in which a fake LTE base station obtains the identity of a 5G UE. Active IMSI catchers attack by impersonating a serving network (SN). 3GPP Release 15 includes protection against IMSI catchers, using a concealed identity called subscription concealed identifier (SUCI). This protection works only when the SN is also a 5G entity, so an active IMSI catcher can mount a downgrade attack against a 5G UE so that impersonates an LTE SN, in order to exploit the weakness of LTE and steal the IMSI of the 5G UE. The authors propose to use a pseudonym-based solution to protect user identity privacy of 5G UEs and to include a mechanism for updating LTE pseudonyms in the public key encryption based 5G identity privacy procedure. The proposed solution uses pseudonyms that have the same format as IMSI for LTE communication to defeat the downgrade attack in order to confound IMSI catchers. Using this mechanism, pseudonyms in the UE and home network are automatically synchronized when the UE connects to 5G. The

proposed mechanism utilizes existing LTE and 3GPP Release 15 messages and require modifications only in the UE and home network in order to provide identity privacy.

2.3 Aspects for practical IoT security

The analysis of literature shows there is lacking of knowledge in certain aspects of security communications for IoT. These aspects are fundamental for employing IoT in security-critical use cases.

2.3.1 Performance evaluation of IoT networks

Several studies about IoT devices and long-range network technologies have been reported in literature. Several groups of researchers have shown interest in finding a solution for their IoT applications, testing which of the several available devices and communication technologies satisfied their needs. Examples of temperature and environment monitoring stations using different boards (e.g. Arduino, Pycom) and different network technologies (e.g. NB-IoT, LoRaWAN) are reported in [103], [104] and [105]. Ugwuanyi et al. [106] instead present a practical deployment of a typical NB-IoT network in an industrial environment. These works do not present experimental results to assess IoT network performance. Other works focus on the evaluation of the performance of communication technologies on different hardware, measuring several metrics. The authors of [107] use a multichannel LoRa gateway and several LoRa end-nodes to evaluate communication range, packet losses, and data rates in an indoor scenario. Oliviera et al. [108] compare the performance of their LoRa and Sigfox prototypes in terms of signal strength and packet delivery ratio. Zhou et al. [109] evaluate the positioning services provided by Sigfox Geolocation and GPS in terms of precision and power consumption, using Pycom boards. The same kind of boards is also used in [110] for a comparison between LoRaWAN and Sigfox in terms of coverage and energy consumption.
The performance of real devices in particular scenarios are evaluated in additional works. Wang et al. [111] study LPWAN performance when using drones moving at high speeds in the air. They evaluate latency and

loss rate of NB-IoT and Sigfox, which are obviously impacted by the high speed of the end-devices and the gateways. Their analysis on the latency is based on the measurement of the arrival of a message to a database server on the Internet in the case of NB-IoT, and to the Sigfox gateway in the other case. Perkovic et al. [112] compare Sigfox, NB-IoT and LoRaWAN using three boards based on Arduino and ultrasound and infrared sensors, which are the most popular sensor devices used in a parking context. They classify the sensors according to their power consumption and detection accuracy.
We have listed a number of works that described examples of use of IoT systems based on LPWAN, with some of them also focusing on the performance of the networks employed. One of the important metrics that has not been analysed in depth by the works mentioned is the latency of a message to its final destination. Perkovic et al. [112] have conducted experiments with the same goal, but LoRaWAN is not included and the authors do not consider the arrival of a Sigfox message to a final destination. Moreover their work evaluate performance in the case of devices moving at high speed, which is a factor that greatly impacts these results. The goal of our research is to evaluate an overall assessment of the performance an LPWAN system, comparing the performance of three network technologies. We aim to evaluate metrics that were not considered in other works, in the condition of devices not moving, recreating the situation of sensors employed for monitoring in a fixed position.

2.3.2 Performance monitoring

Among the several use cases of CPS, distributed robotics has been the subject of research about the introduction of monitoring for optimization. In particular, the successful coexistence of different computational paradigms (such as Cloud or Edge computing) can be achieved only by monitoring computational and network resources of robots. *Distributed Robot Monitoring System (Drums)* [113] is a proposed a tool for monitoring distributed robot systems and their underlying resources. One of its goals is to find bugs in the underlying systems abstracted by middleware systems such as ROS. A middleware adapter has the function of monitoring the state of the middleware to maintain a computation graph, initiating or removing the monitoring jobs, and acting as a bridge between the middle-

ware and the Drums infrastructure. The monitoring infrastructure gathers information from various sources to describe the state of the model. The collector works a daemon application that collects statistics from elements of the computation graph and pushes the collected data over a publisher/-subscriber channel. Data is collected using four monitoring modules which collect information about operating system processes, resource utilization (CPU, RAM, I/O, network), packet length, and latency. Rivera et al. [114] propose another monitoring framework for robots based on ROS, developing a system for network monitoring using the extended Berkely Packet Filters (eBPF) and eXpress Data Path (XDP). Their proposed framework includes a security policy enforcement tool and distributed data visualization tool for both ROS1 and ROS2. Another aspect investigated in literature is execution monitoring and failure detection. Robots do not have complete knowledge of the environment in which they operate, so the state of a robot and the surrounding environment should be continuously monitored in order to detect anomalies or failures during task execution. Failure detection in industrial robots can be classified in knowledge-based, relying on the experience of expert users, model-based, relying on a mathematical representation of the system, and data-driven, which uses machine learning and measurement data to build a representation of the robot operation process. Using the different communication behaviors between the modules of the control software in order to monitor the status of the system and to detect and localize faults is the idea at the basis of [115]. Observers continuously monitor communication between the modules, and if abnormal communication is observed, the diagnosis engine is triggered for determining the reason of failure. Steinbauer et al. [116] present a solution for detection, localization and repair of faults in the control software of autonomous robots. The proposed diagnosis system uses model-based diagnosis for fault detection and localization. The problem of fault detection and identification could also be addressed by using linear approximations of rover dynamics or approximating the solution using Monte Carlo methods [117]. Machine learning approaches have also been proposed in failure monitoring systems. A technique for curating data before building a classification model is proposed in [118]: the results of this work show that pre-processing techniques improve the failure detection performance. Inceoglu et al. [119] on the other hand present a system to detect manip-

ulation failures using past observations acquired from different sensors as a training set to create models of success and failure. A framework for anomaly detection based on offline training and online inference is proposed in [120]. The offline phase trains its model on historical data, the the prediction model obtains the prediction results for anomaly detection. New anomalies in real-time sensor data are detected using incremental learning to update the online model.

2.3.3 Secure task distribution

Resource-constrained devices belonging to a CPS need the support of resource-rich machines to execute computational and memory intensive tasks. Computationally expensive tasks will need to be sent to powerful devices in a process called computational offloading. There are different possible solutions to this problem: reaching Cloud services via internet or relying on instances closer to the edge of the network. These instances could be Fog or Edge nodes. Fog computing is a distributed computing paradigm that provides data and services closer to end users. On the other hand, Edge nodes are typically in the same local network of the end-devices and are preferable when latency requirements are stringent. Multiple edge nodes should be present in the network to satisfy the demand and to support application diversity.
Offloading has been a very popular research topic in recent years. One of the main goals for the research in this topic has been the optimal resource allocation for the tasks of a collaborative system made of multiple peripheral machines and Cloud instances. Afrin et al. [121] address simultaneous optimization of energy consumption, makespan, and cost while allocating resources for the tasks of a robotic system. The authors design an Edge-based multi-robot system: the Edge layer assists robotic systems by executing the latency-sensitive services. The resource allocation for robotic workflow is modelled as a constrained multi-objective optimization problem, in which energy consumption of resources, overall makespan and monetary cost are targeted to be simultaneously minimized. Zhao et al. [122] try to solve a computation offloading problem when Cloud and Mobile Edge Computing (MEC) are collaborating. The authors propose a collaborative computation offloading and resource allocation optimization (CCORAO) scheme. This scheme achieves the optimal solution, under the

constraints of task processing delay and cost of computation resource. The allocation of Fog computing resources for IoT users is studied in [123]. The proposed model envisages IoT users competing against each other in order to maximize its own quality of experience, in terms of energy consumption and delay. The authors also propose a near-optimal resource allocation mechanism that reduces the computation delay and enables low-latency fog computing services for delay-sensitive IoT applications. Another approach for adaptive task offloading in Fog is the one proposed for FRATO, a framework for collaborative task offloading in IoT systems [124]. FRATO is based on an adaptive task offloading mechanism to select flexibly the optimal offloading policy in order to offer the minimal service provisioning delay. An appraoch based on Fog nodes ranking is proposed in [125]: Fog nodes are ranked according to the available resources that can fulfill the requirements of the incoming requests. The resources of Fog nodes are dynamically allocated, and a high priority Fog node for processing is chosen. The load of individual Fog node is then calculated and sent to the broker for the next decision. Meng et al. [126] focus on minimizing the energy consumption for both communication and computation in the case of Cloud computing servers and Fog computing servers. To solve the computational offloading problem, the problem is divided into four subproblems according to the computation energy efficiency. The execution time and energy consumption of mobile devices are also analyzed by Xu et al. [127], which propose a computational offloading method for solving the multi-objective optimization problem for IoT-enabled Cloud-Edge computing. Du et al. [128] address the computation offloading problem in a mixed Fog/Cloud system by jointly optimizing the allocation of computation resource, transmit power, and radio bandwidth, while guaranteeing user fairness. The optimization problem is formulated to minimize the maximal weighted cost of delay and energy consumption among all UEs. The authors propose a low-complexity suboptimal algorithm to solve the mixed-integer non-linear programming problem. Sarker et al. [129] present a system architecture for offloading computationally expensive tasks to Edge gateways which rely on Fog services. Their work shows that the battery life of a prototype robot can be improved by moving computation to the Edge layer. Offloading can also be addressed using a game theoretic approach [130], in which a Nash equilibrium is sought, or by

finding an optimal allocation of tasks via linear programming [131]. Pu et al. [132] propose a mobile task offloading framework based on network-assisted device-to-device collaboration. The computation and communication resources can be shared via the control assistance by the network operators (e.g. base stations). The goal is to achieve energy efficient task executions, in particular to minimize the time-average energy consumption for task executions of all users.

The interested in performing network informed task offloading has been limited so far. Sacco et al. [133] propose a distributed and adaptive task offloading algorithm when the network conditions fluctuate. Their algorithm predicts the length of future task queues in order to anticipate the node overloading and to avoid the exhaustion of batteries and computational resources. The Autoregressive Integrated Moving Average (ARIMA) algorithm is used for determining the future load of an agent. The agents are able to communicate directly with each other, with each node capable of monitoring the channel between itself and the rest of the network. One of the most common monitored indicators is RSSI: when it is lower than a threshold, the signal is too weak and it not recommended not to migrate tasks to that node.

When multiple Edge nodes are present, a resource-constrained IoT node needs to decide which edge node it should offload a task to. Edge nodes, just like sensor nodes, can often become targets of several attacks. Many of them concern network level, and also well-known attacks like eavesdropping, jamming, and others, are applicable to these resource-constrained systems. It is then important to make the right choice when deciding the edge node to which offload tasks to and exclude the edge node under attack from the rest of our system. Previous works [134, 135] address this problem by assessing a measure of behavioural trust that a resource-constrained node has in a resource-rich node to correctly execute a task. Trust has been used in several areas of interest, such as routing of messages in wireless sensor networks [136], or intrusion into the network [137]. Trust has been used to solve several problems in IoT applications, such as authenticity of the sender of a message. Trust has also been evaluated using several machine learning approaches. Support vector machines are trained offline for facilitating behavioural trust classification in [138], while a model-free reinforcement learning technique is used to learn an offloading

policy in [139]. These approaches however are unsuitable for use in systems in which powerful memory and processing power are lacking. This means that lightweight approaches such as the Beta Reputation systems are needed. When trust information is required, a node needs to keep track of a history of interactions with the different edge nodes, hence this kind of approach is not suitable for the resource-constrained IoT networks. Existing trust-based evaluation approaches typically use a history of interactions to assess a trust value. The Beta Reputation System [140] classifies interactions as good or bad, and calculates the expected likelihood of a future event being good or bad based on the history of interactions. This trust model maintains two counters: one for the good events observed and one for the bad events observed. The trust value is then assessed via the expected value of this distribution, calculated as the the ratio of good events to the total number of events. Another simple method that can be used for computing trust is using a hidden Markov model (HMM) to evaluate trust in systems exhibiting dynamic behaviour [141]. Bradbury et al. [142] overcome the problems of resource constraints and recency by proposing a proactive approach for trust evaluation, based on periodic challenging a resource-rich node with a non-trivial task. The nodes that can be trusted or not are recorded by a trust tracker.
The choice of the most adequate task offloading model should also take into consideration the resources of the involved devices. For example, expensive Edge evaluation techniques are used in Multi-access Edge Computing (MEC), the task offloading problem in vehicular and cellular networks. These techniques are not practical in IoT networks because of the resource constraints. There are a very few proposal for task offloading focused on resource-constrained devices. Hasan et al. [143] propose a model for offloading in direction from resource-rich mobile clients to resource-constrained IoT devices. A task offloading solution for TinyOS is presented in [144], but it does describe how another device would be selected for offloading. The authors of [145] propose a scheme for sharing tasks among peer sensor nodes, considering their computational and networking conditions, but not the presence of a resource-rich device. Bradbury et al. present a trust-based system architecture for computation offloading, based on behavioural evidence [146, 147]. This system architecture provides confidentiality, authentication and non-repudiation of messages, and

is able to operate within the resource constraints of IoT. This architecture is designed for arbitrary trust models and multiple applications running on both edge nodes and IoT devices. The feasibility of this architecture with an example deployment of the Beta Reputation System. The goal of the authors is to realize decentralised behavioural trust assessment and decentralised edge node selection for task offloading.
The capability of discriminating a malicious edge node from the rest of the system allows users to always communicate with trusted entities. The approach we introduce bases its decision on the monitoring of the network performance, which is a novelty in literature. The isolation of the corrupted nodes in an IoT systems enables of safe use of critical applications.

Table 2.12. Well-known attacks on IoT networks.

	SFX	LRW	NBI	BLE	802.15.4	Z-Wave	ZigBee	5G
DoS and battery draining		[84]	[85, 65]	[86]	[29],[26]	[33]	[43],[87], [88]	
Eavesdropping				[89],[86], [18]		[90]	[44],[43]	
Packet forging and manipulation		[65],[91], [92],[93]			[25],[26]			
Replay	[65]	[92],[84]					[43],[44]	
Man in the Middle		[94]	[95]	[20]		[37]		
Wormhole		[63]	[73]					
Impersonation and rogue controllers						[36],[96], [37]	[44]	
Jamming		[59],[92], [65],[63], [97]			[25]			[79]
Privacy leaks				[98]				[78],[99]

Chapter 3

Methodology and tools

In this chapter we describe the various devices and testbeds employed in our research. The network connections employed are different, but all the experiments share a same basic idea: IoT sensor nodes must reach a more powerful entity for processing the collected data. The information generated by the Things is usually unstructured data that require Big Data techniques to be processed in order to extract something useful. Moving the data to the Cloud could be a possibility, but new challenges appear in this scenario. Edge computing is then a solution for extending powerful computing closer to the Things. An edge computing node can be any network device with the capabilities of storage and computing. These edge nodes form a support network for the resource-constrained devices needing a support network for computationally expensive tasks. Edge nodes collect the tasks offloaded by the resource-constrained devices and calculate the result. Edge computing is preferred for computational offloading in cases when the network has no access to Cloud services or when latency is important, so edge nodes should exist in the same local network as the IoT devices.

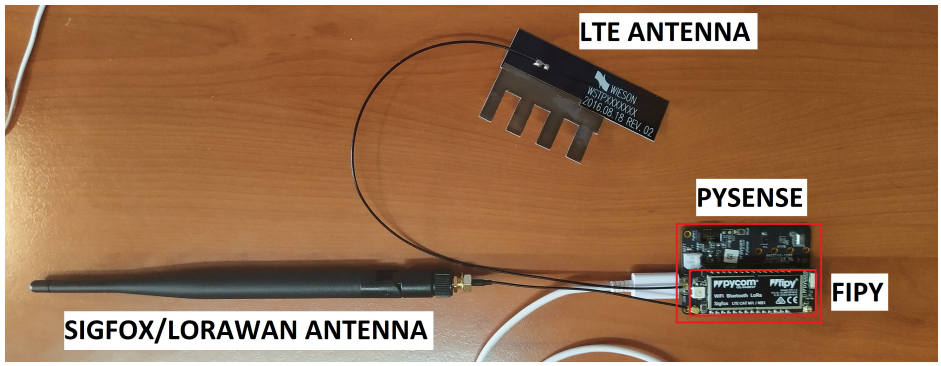


Figure 3.1. Our FiPy with the Pysense shield and the antennas.

3.1 Performance evaluation of IoT networks

3.1.1 Development boards

In the experiments of performance assessment of LPWAN networks, the testbed used is made of IoT development boards equipped with sensors. The board used in our experiments is Pycom FiPy, a programmable board that features WiFi 802.11 b/g/n, Bluetooth (LE and classic), LoRa, Sigfox and dual LTE-M (CAT-M1 and NB-IoT) [148]. It runs on an Espressif ESP32 chipset and is provided with 4 MB of RAM and 8 MB of flash memory. The FiPy board is equipped with a firmware based on Micropython, an optimised implementation of the Python 3 language [149]. The Micropython firmware is based on a RTOS kernel, which can be seen like a BIOS or library and not as a full OS. Pycom boards support standard Python libraries, Micropython-specific libraries and modules specific to the Pycom devices [150].

Figure 3.1 shows the device at our disposal: we use a FiPy device mounted on a Pysense board, which is a sensor shield containing ambient light, pressure, humidity, and other sensors. When powered up normally or upon pressing the reset button, the FiPy module will boot into standard mode, with the execution of the boot.py and the main.py files.
For interfacing with the module, the needed software is Pymakr, a plugin for both Visual Studio Code and Atom IDE code editors. The easiest way to connect to the FiPy is via an expansion board or a Pysense shield. By

Figure 3.2. Sigfox network overview [3].

default, the FiPy module runs an interactive Python REPL: code can be run via this interactive REPL or via the PyMakr plugin, which is also useful to upload code to the board. The FiPy can also act as a Wi–Fi access point: once connected to the FiPy's Wi–Fi network it is possible to access it in two ways: via Telnet or via FTP. The FiPy also runs a FTP server that allows us to copy files to and from the device, including an SD card if one is connected. The Telnet and FTP server do not work only when the FiPy acts as an Access Point, but also when the FiPy is connected to our Wi-Fi network and we know its IP address.

3.1.2 Sigfox

The Sigfox network architecture is shown in Figure 3.2 and its various components are described in the following.
Every Sigfox Pycom device comes with a Sigfox DevKit contract included, that provides one year of Sigfox connectivity. Every device needs to be registered on the Sigfox backend using its Sigfox ID and Sigfox Porting Authorization Code (PAC), a couple of unique numbers mandatory for device registration. Sigfox impose a limitation that sets a maximum number of 140 daily uplink messages and 4 daily downlink messages.
It is possible to manage all the registered devices on the Sigfox backend, known as *Sigfox Cloud*. A scheme of the Sigfox Cloud is shown in Figure 3.3. On this backend, a company is represented by a *group*, which contains at least one *device type*. A device type regroups all units of the same product, to allow them to behave in the same way when the Sigfox network receives a message. The Sigfox Cloud also receives the data sent from the devices, and automatically forward them using callback integra-

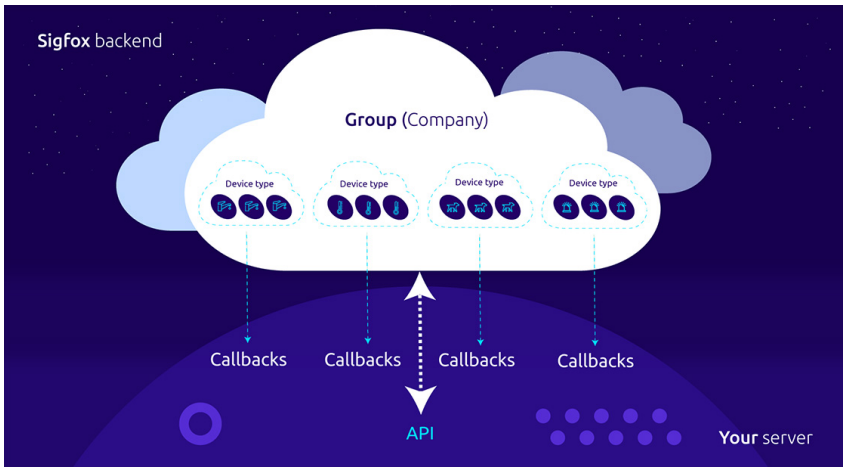


Figure 3.3. Sigfox Cloud [4].

tions defined on a per-device type basis [4]. Callback queries are HTTP requests working as notification messages tied to a device type, that our server can use for obtaining data but not for managing devices. When the Sigfox Cloud receives a message from an emitting device, it instantly generates a callback message and sends it to our endpoint. Callback services are triggered every time any device sends a message and send custom request containing data to a given server or platform. The callback used in our tests is a Data Callback, triggered by the reception of an uplink device message. In the definition of a callback we need to set the URL to which forward the message, the HTTP method to be used, the type and the body of the message forwarded. In the body of the message we can use some the defined variable provided by Sigfox: common variables available for uplink data callback service are time, deviceTypeId, device, data and seqNumber. The callback defined for our tests, as shown in Figure 3.4 uses the POST method, its body is formatted in JSON and the fields in the message are the data, the device id and the event timestamp.

3.1.3 LoRaWAN

To communicate using LoRaWAN, we used a FiPy as a node and another FiPy as a *nano-gateway*. The nano-gateway is connected to a Wi-Fi

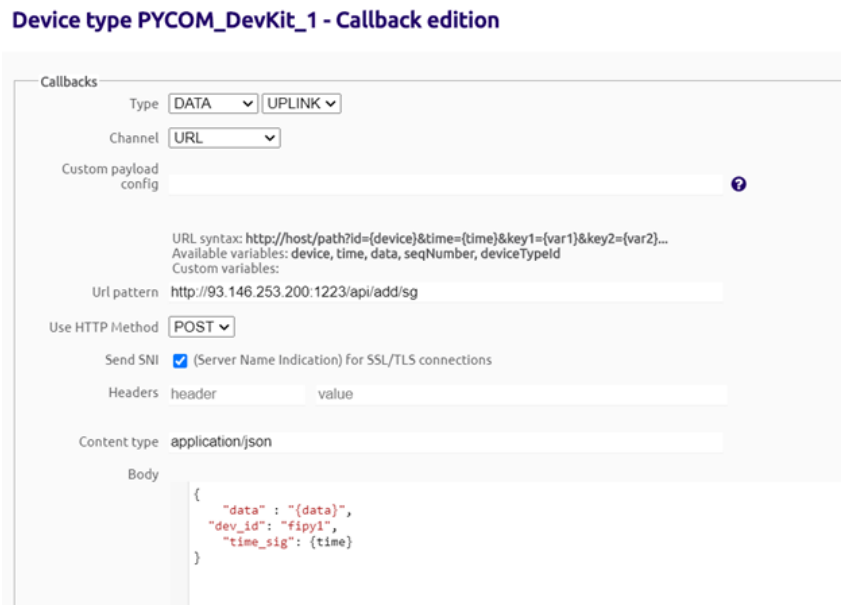


Figure 3.4. Sigfox callback.

hotspot to reach *The Things Network* (TTN), a decentralized infrastructure for the Internet of Things. TTN supports LoRaWAN for long range (5 to 15 km), low power and low bandwidth communication [151]. TTN provide a set of open tools and a global, open, free of charge network to build an IoT application at low cost. TTN backend systems route IoT data between devices and applications. TTN has its own fair access policy [152]: 30 seconds per day of uplink time per device and 10 downlink messages per day. It is also advised to send payloads as small as possible, to have a large interval between two consecutive messages and to use a high data rate to minimize our airtime [153].
The main components of a LoRaWAN network, shown in Figure 3.5, are described in the following. Nodes broadcast LoRaWAN messages that are received by one or more gateways, which work as bridges between radio protocols and the Internet and forward data to the backend [154]. Gateways are connected to a router, a microservice that receives messages from the gateway and that is responsible for managing the status and the scheduling

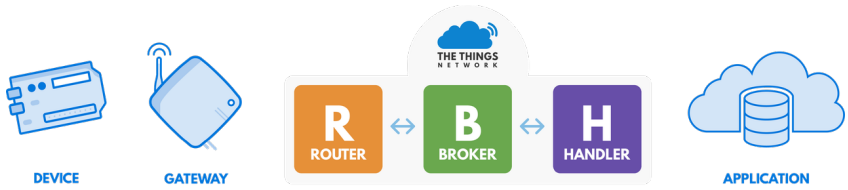


Figure 3.5. Main components of the LoRaWAN network [5].

of the transmissions of the gateway. Each router is connected to one or more brokers, which map devices to applications, forward uplink messages to the correct application and downlink messages to the correct router. A handler is then responsible for handling the data forwarded by the broker to one or more applications. The handler is also the point where data is encrypted or decrypted. It is also possible to deploy private network and run all these components in a private environment.

In order to use TTN, we must register to its website and register our devices either as a node or as a nano-gateway. To register our device as a node and to connect to TTN, we must create an application which our devices will belong to. By applications, TTN developers mean whatever it is the devices communicate with on the Internet. TTN allows us integrate applications in order to communicate with the registered devices. The HTTP integration allows to send uplink data to an endpoint and receive downlink data over HTTP. It is possible to configure the URL, the HTTP method and optionally the HTTP Authorization header [155].
Gateways can run even on minimal firmware, so it is possible to setup a Pycom module as a nano-gateway in order to form the bridge between devices and TTN [156]. Devices use LoRaWAN to connect to a gateway, which then uses high bandwidth networks like Wi-Fi or Ethernet for connecting to TTN. All gateways within reach of a device will receive the messages from a device and forward them to TTN.

To analyze the content of the message forwarded by TTN, we set up an integration to Requestbin, where it is possible to collect HTTP or webhook requests and inspect data. TTN sends a lot of information and metadata together with our payload. The value of interest is in the payload_fileds field, as shown in Figure 3.6. As we can see from the data received by Requestbin, TTN specifies the URL that can be used to send downlink

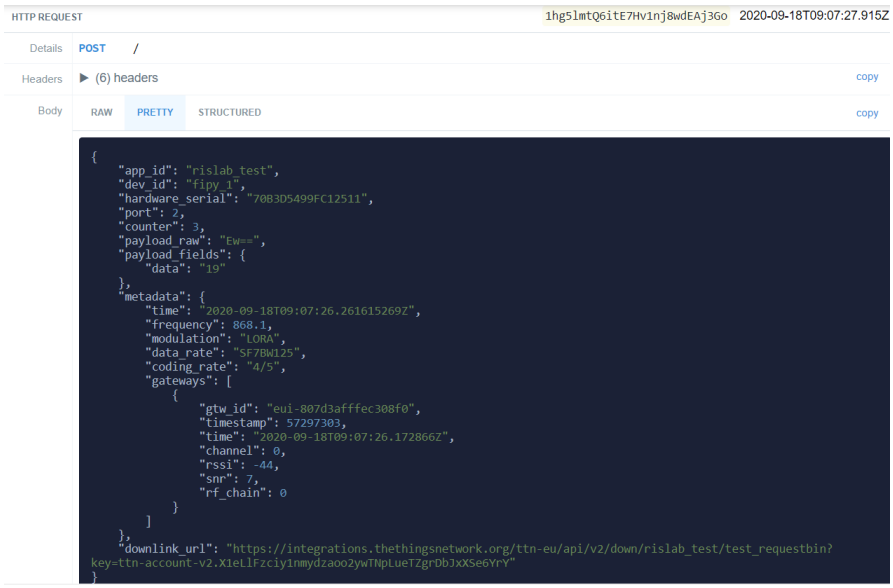


Figure 3.6. Message sent from TTN and received by Requestbin.

messages to our device.

3.1.4 NB-IoT

The FiPy connects to the NB-IoT network using a Vodafone SIM card provided by Pycom. These Vodafone SIM cards can only be used with Pycom cellular devices, and Pycom NB-IoT connectivity provision has been configured to work only with Pycom's *Pybytes* platform. The devices are supported by an airtime service with an encrypted end-to-end data connection from each endpoint to the back-end systems in Pybytes. By default the devices connect to Pybytes over a shared APN using a dynamic address, but it is also possible to opt for a dedicated private APN so that only SIMs assigned to our organisation can transfer data and unauthorised access is prevented [157]. The code employed to connect to NB-IoT needs the Pycom module *LTE* to enable radio functionality, attach to the LTE network and start a data session. The APN employed is *pycom.io* while the band employed is Band 20 (uplink band: 832 - 862 MHz, downlink band: 791 - 821 MHz, bandwidth: 30 MHz). The device connects to the

Figure 3.7. Data received in Pybytes using NB-IoT connection.

broker *mqtt.pybytes.pycom.io*, which is hosted on an Amazon Web Server in Germany. The code needs other Pycom libraries to collect data from the Pysense sensor shield. The data are sent to Pybytes using the function *pybytes.send_signal* which takes as input the number of the signal (where data is saved on Pybytes) and the data, sent as a string or as a integer. The data received at Pybeytes in one of our preliminary tests appear in Figure 3.7.

Pybytes, the free cloud-based device management platform available for all Pycom development boards, offers a way to interact with external IoT platforms or custom services [158]. To interact with a remote destination, it is possible to use a Web Hooks integration, that allows to define HTTP callbacks [159]. Whenever one of the integrated devices sends a signal to Pybytes, the platform performs an HTTP request defined by the user. The user can specify the remote URL to which the data will be sent, the HTTP method (POST, PUT, etc.) and the request format (JSON, custom, etc.). It is also possible to define the body of the message forwarded to the remote destination. An example of callback is reported in Figure 3.8: in this case, the remote destination is our backend and the data exchanged is the message payload received by Pybytes and sent in a JSON format.

3.1.5 Description of the experiments

A correct assessment of performance parameters for the three LPWAN technologies is crucial for deciding if it is possible use these communica-

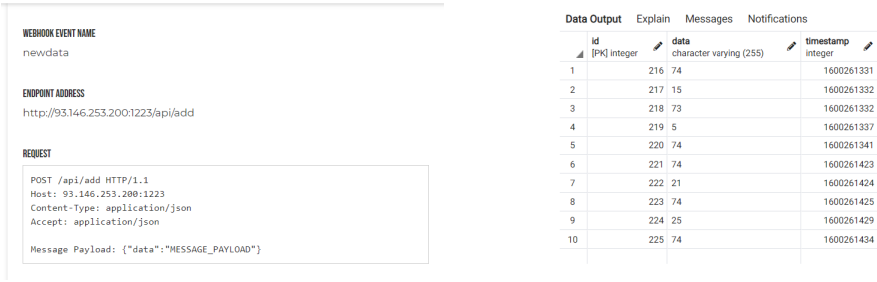


Figure 3.8. Webhook application and data received in our database.

Table 3.1. Performance metrics.

Quantity	Methodology
Clock accuracy	Difference of the timestamp at our backend server and the non-updated timestamp sent by the device
Energy efficiency	Number of messages sent with a fully charged battery
Message losses	Comparison of number of sent and received messages
Latency	Difference of the timestamp at our backend server and the updated timestamp sent by the device

tion in safe and secure IoT applications [160]. Some of the main metrics that describe the qualities of an IoT system are the energy efficiency (low consumption also means low cost), the percentage of message losses, and the latency of a message (aiming at the shortest time to deliver a message to its final destination). Another parameter evaluated in our tests is the accuracy of the clock on the device. All the metrics considered are in Table 3.1.

Our testbed is made of FiPys connected to a sensing board and communicating with the appropriate gateway or base station. The testbed is represented in Fig. 3.9. The performance analysis of the devices and the network technologies is carried out by sending a message from the device containing the same information: a timestamp, information on the battery, and data from the temperature, the humidity and the light sensors. The timestamp and the information on the battery are sent because they are fundamental to our analysis, the other data are chosen among the data available from the Pysense sensor board which allow to maximize the in-

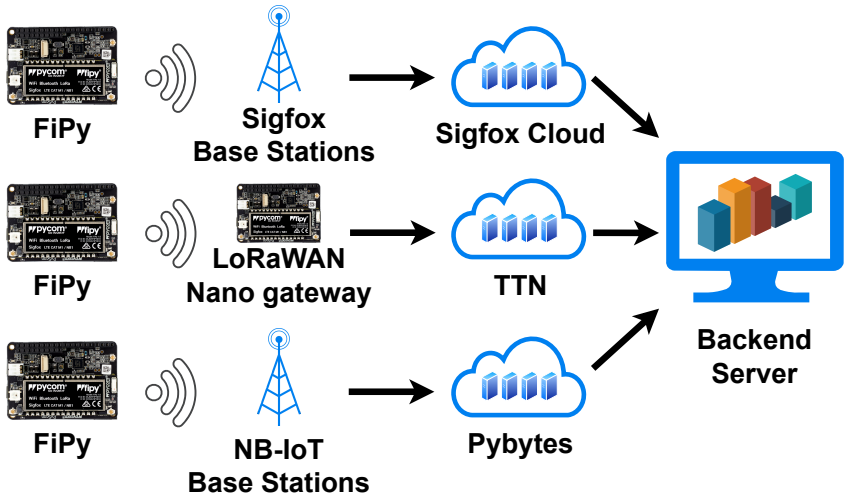


Figure 3.9. Testbed employed in our tests.

formation carried by a Sigfox payload, whose maximum size is limited to 12 bytes. The payload sent over LoRaWAN is identical, while the payload sent over NB-IoT has a different format. The sent messages go through the different IoT networks and platforms and are finally delivered to our backend server, where their payload is saved along with the timestamp of its arrival. The data useful for the performance analysis are retrieved from the device log files and the messages received from our backend. The energy efficiency is evaluated considering the number, reported in the log files, of the messages sent with a fully charged battery. The message losses are calculated by comparing the number of sent messages reported in the log files and the number of received messages at our backend. The latency is evaluated by subtracting the sent timestamp to the timestamp of the arrival at our backend, while we evaluate the accuracy of the clock on the device by considering the mentioned timestamps.

The scripts employed in our tests use functions provided by the Pycom documentation. The executed operations are: awakening of the device, connection to a Wi-Fi network to synchronize to an NTP server, retrieval of the data from the sensors, creation of a socket/connection and sending of the message, deinitialization of the LTE modem and entrance in sleep

mode in order to save power.

3.2 Performance monitoring

As mentioned earlier in this work, the use case we consider as an example of CPS for introducing a monitoring platform is distributed robotics. The basis for the design and implementation of a platform for monitoirng in robotics applications is Robot Operating System (ROS), one of the most relevant software platforms available for robotics. ROS is an open-source, meta-operating system for robots [161]. It provides the typical services offered by an operating system (e.g. hardware abstraction, message-passing between processes, etc.) and tools and libraries for software development and execution. ROS is an excellent solution for the robotics applications that need distributed computation [162] and that rely on software that runs across several different machines. One of the basic goals of ROS is to design software as a collection of small independent programs called *nodes*, processes that perform computation and communicate with one another. Communication between nodes is possible thanks to the different paradigms of communication provided by ROS:

- synchronous, for request/reply communications;
- asynchronous streaming of data over topics, buses over which nodes exchange messages, for unidirectional communication, supporting publish/subscribe semantics;
- Parameter Server, a shared dictionary used to store and retrieve parameters at runtime.

A new version of ROS, called ROS 2, is now available and is intended to support real time programming, a wider variety of computing environments, and utilizes more recent technologies [163]. ROS 2 uses Data Distribution Service (DDS) as the underlying communication standard [164]. ROS 2 provides a ROS 1-like interface on top of DDS which hides much of the complexity of DDS for the majority of ROS users, but provides access to the underlying DDS implementation for users that have particular use cases. DDS provides a publish-subscribe transport which is very similar to ROS. The main difference between the communication standard

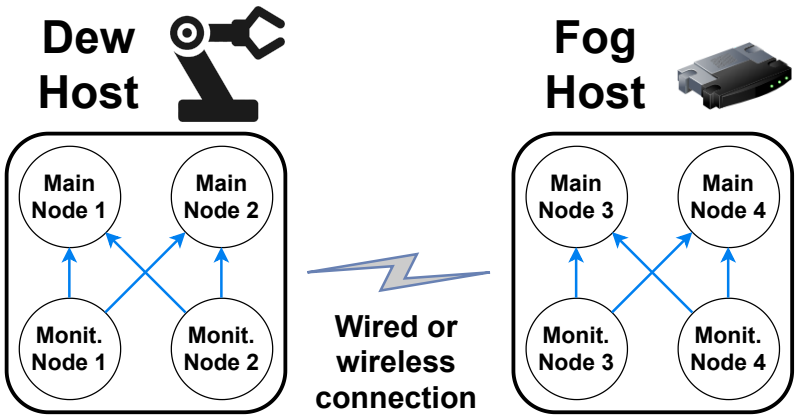


Figure 3.10. DewROS2 architecture.

of ROS and ROS 2 is the discovery mechanisms: the default discovery system provided by DDS is in fact distributed. ROS instead has a master which works like a DNS server that returns information to queries. ROS 2 does not have a master and all its nodes are connected in a peer-to-peer architecture, making the system more fault tolerant and flexible.

The platform that we design for the monitoring in distributed robotics CPSs is called DewROS2. The architecture of DewROS2 is composed of several different nodes interacting with each other and distributed over different network hosts, as shown in Fig. 3.10. These nodes can communicate with each other through wired or wireless network technologies, depending on the scenario. In DewROS2 architecture there are two main groups of nodes. The first group is composed of the nodes that actually execute the needed operations to fulfill the final goal and they are deployed on the machine that best satisfies their needs in terms of resources. These nodes, called main nodes in the following, carry out different tasks, which may or may not need powerful resources and that can be more or less critical. Typically, the dew hosts have simple hardware and limited power supply, so they host the less computation-demanding operations, while the fog hosts have more computational, storage and power resources, appropriate for the more demanding tasks. The tasks executed by the main nodes depend on the use case of the desired application. The second group is composed of several ROS nodes that monitor performance parameters regarding the

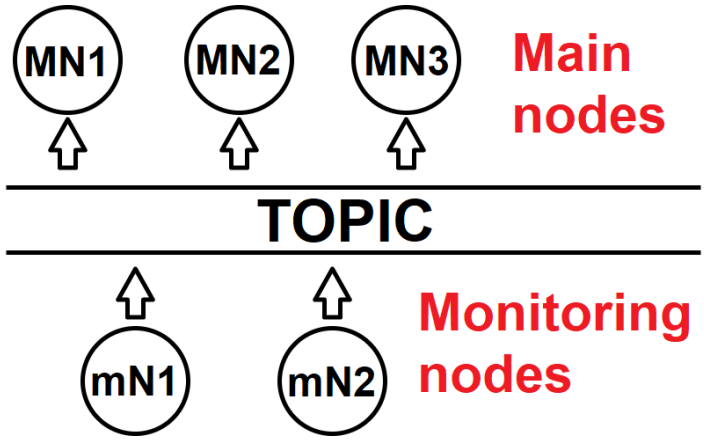


Figure 3.11. Main nodes (MN) and monitoring nodes (mN) in DewROS2.

hosts they run on and interact with the nodes belonging to both groups. We refer to these nodes as monitoring nodes in the following. The main nodes can exploit the information collected and shared by the monitoring nodes in order to change their operating conditions and to optimise their operations. The communication among the nodes is in line with the ROS communications paradigm.

DewROS2 works as follows: the main nodes start their operations on the Dew or Fog machines. In the meantime, the monitoring nodes begin their work and periodically inform the other nodes, sending messages via ROS topics containing the values measured. Thanks to the information received, the main nodes can make changes about their operating conditions and adapt to the status of the system. Fig. 3.11 gives a schematic representation of how the different nodes interact with each other and communicate over a topic.

We believe that DewROS2 will be more and more important in the future as it provides the necessary support for several operations in a multi-layer Cloud Robotics scenario. Having computational power at Edge or Dew level is a solution for overcoming the main issues of Cloud Robotics, which are control of the network and absence of local context information, and to satisfy the requirements of latency and safety of some applications. In order to know which tasks to offload remotely and which execute locally,

robots need to have a comprehensive view of the scenario they are working in: network conditions, resource loading, etc. It is then clear how important it is to gain information: this is the motivation of the introduction of our DewROS2 solution. DewROS2 is suitable for all the distributed robotic applications where devices with limited resources are employed and where the monitoring of performance parameters can bring advantages in taking informed decisions when resource optimization becomes an important goal.

3.2.1 A use case: DewROS2 in SHERPA

A possible critical field in which DewROS2 could be applied is Search and Rescue activities. We specifically consider the context of SHERPA [6], an European project whose research activities started in 2013. The real world scenario inspiring the SHERPA activities is the surveillance and rescuing in unfriendly and hostile environments like mountains or forests. The project focuses on the alpine rescuing scenario but its results can also be applied to other rescuing and surveillance fields such as natural disasters or building crashes. The main goal of SHERPA is to develop a robotic platform supporting the rescuers in their activity in order to improve their capabilities while decreasing the costs and the risks. The adverse environmental conditions in which the platform operates ask for robust control and communication. The activities of SHERPA are focused on a combined aerial and ground robotic platform to support human operators in surveillance and rescuing tasks in hostile environments, like the alpine scenario targeted by the project (Fig. 3.12). The "SHERPA team" is made of four components:

- A human rescuer expert of the specific rescuing mission which transmits wirelessly his position to the robotic platform using a device that does not distract him from his rescuing actions and that does not flood him with irrelevant information. The human rescuer is able to provide high-value inputs to the robotic platform in order to achieve the team goal thanks to his experience in the field;
- Small scale rotary-wing Uncrewed Aerial Vehicles (UAVs) equipped with small cameras and other sensors, used to support the rescuing mission by enlarging the patrolled area with respect to the area potentially covered by the human rescuer alone. These vehicles are

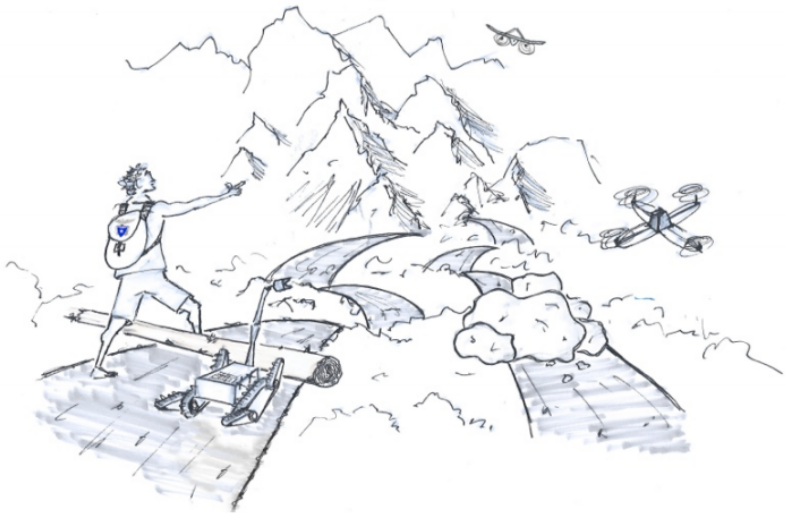


Figure 3.12. A sketch of the SHERPA team [6].

designed to operate in autonomy as if they were "flying eyes" of the rescuer. They have limited autonomy and limited operative radius, but on the other hand they have the capability of capturing data in accessible areas with high manoeuvrability;

- A ground rover that serves as a transportation module for the rescuer equipment, as a hardware station with computation and communication capabilities, and as a recharging module for the UAVs. It is conceived to operate with a high degree of autonomy and long endurance. In order to improve the capabilities of the robotic platform, a robotic arm is also installed on the rover. It has intrinsic limits in terms of reaching wild areas and overtaking big natural obstacles;

- Long-endurance and high-altitude aerial vehicles with complementary features with respect to the small-scale UAVs introduced before. These are used to construct a 3D map of the rescuing area and to patrol large areas. They fly at a height of around 50-100m above ground. The high altitude information captured by these vehicles enables coordination of the local activities of the team.

Table 3.2. Monitoring nodes employed in this work.

Passive nodes			
Label	**Measure**	**Command**	**Description**
CU	CPU utilization	psutil	CPU usage percentage
BC	Battery charge	upower	Battery percentage
PB	Passive bitrate	ifconfig	Average bitrate over a time period
SQ	Socket queue size	netstat	Size of data waiting to be transmitted
Active nodes			
Label	**Measure**	**Command**	**Description**
AT	Achievable throughput	iperf3 and D-ITG	Maximum achievable throughput
AL	Application latency	custom script	Evaluation of the RTT between two ROS nodes

3.2.2 Description of the implementation

The monitoring nodes implemented and used in this work are reported in Table 3.2. These nodes can be classified in two groups: passive and active. A passive node is a single entity that monitors a specific quantity on the machine it is deployed on using utilities and libraries provided by the operating system. The active nodes on the other hand rely on the interaction of multiple nodes deployed on more than one machine to monitor the quantity of interest.
The measurements and the publication of their results are periodic, and it is possible to choose the monitoring and the publication periods. The measured values are published on a ROS topic and can also be stored in a CSV log file. The monitoring nodes can log the results of their activity using two different levels of detail: log level 1, in which the monitoring nodes store only their final output in a single file, and log level 2, in which the monitoring nodes also store the output generated by the command used (ifconfig, netstat, etc.) in different files used for debugging purposes. The user has the possibility of deciding which monitoring nodes to activate using a bitmask, and whether to launch them sequentially or in parallel

using separate threads.
The quantities that we decided to monitor in this work are the CPU utilization, the battery charge, and different parameters regarding network performance. These parameters are of fundamental interest in optimizing the operating conditions of a distributed robotics system. For example, knowing the state of the CPU usage could be useful to decide whether a machine has sufficient resources to host another task or if the current task has enough resources to be correctly executed. Knowing if the battery charge is decreasing too quickly could be a index that too many tasks are being executed on a machine. A main node can use this information to reduce the number of running processes when the CPU is overloaded or to decide where to perform a task according to the energy requirements and availability on the robot. Network performance metrics is essential in a variety of cases. Knowing the achievable throughput between the hosts is useful to understand whether the network is able to sustain our application or if it is necessary to reduce its load. Another example could be when a Dew host needs to communicate with a Fog host in a safety critical application: the latency between the two hosts should not be high. In case of a high latency, the Dew host should rely only on its capabilities and not offload critical tasks to the Fog.
The CPU utilization (CU) node returns a float representing the current CPU utilization as a percentage, while the battery charge (BC) node retrieves the residual percentage of charge. The passive bitrate (PB) node monitors the numbers of bytes sent or received on a network interface at two consecutive moments and calculates the average bitrate sent or received over a time period. The socket queue size (SQ) node checks the size of the queues of the data that are waiting to be transmitted on a socket. These size values give hints about the bitrate availability and are used by the main nodes in order to decide whether the bitrate provided by the network is high enough to transmit the data of the application. The achievable throughput (AT) nodes measure the achievable throughput over a network link. Two types of this node were implemented, using two different commands: *iperf3* and *D-ITG*. *iperf3* is a network monitoring tool which uses the client-server paradigm to monitor the amount of data transferable. The client node sends data to the server, which then sends them back to the client. D-ITG (Distributed Internet Traffic Generator) [165] is a tool

able to measure the most common performance metrics (e.g. throughput, delay, jitter, packet loss) at packet level. D-ITG uses three different entities (sender, receiver, decoder) deployed on different machines to evaluate the achievable throughput. The application latency (AL) node evaluates the round trip time (RTT) between the Dew host and the Fog host. The RTT is measured by sending a timestamp to the Fog host over a topic. The Fog host receives this timestamp and sends it back to the Dew host on another topic. The Dew host can then calculate the difference between the timestamp sent and the current time to evaluate the RTT.
Various machines and devices are involved in the creation and testing of this monitoring platform. One of the goals of the experiments carried out is to prove that this platform can be employed on resource-constrained devices: here is why different machines are involved in these experiments. The platform has been firstly tested on two Ubuntu virtual machines, one acting as the peripheral node and the other as the edge node. The two machines communicate through their Ethernet interface and carry out a video acquisition application. This series of experiments allows us to evaluate the impact that monitoring nodes have on the resources of a machine. Our goal is then to compare the resource usage when only the monitoring nodes are working with the resource usage when both main and monitoring nodes are enabled. The overhead generated by the active and the passive nodes is also evaluated separately. The resource usage is monitored through external Linux tools. The platform has been later deployed on a drone performing SLAM, the activity of constructing and updating the map of an unknown environment while also keeping track of the location occupied in such map. The Fog host is once again an Ubuntu virtual machine running on a Windows computer. The SLAM operations are executed on the hardware of the drone, while the Fog host is responsible for running the entities of the AL and AT nodes that need to be executed on the Fog side. The information that we use for studying the outcome of the experiments is retrieved from the log files stored by the monitoring nodes on the drone. Another scenario in which the effectiveness of the monitoring is tested considers the same deployment of the monitoring nodes between the Dew and the Fog hosts, but on different hardware. In this case, the Dew host is a Raspberry Pi able to move inside our laboratory, while the Fog host is a Virtual Machine running on a commercial router. The two

hosts communicate via Wi-Fi. The Dew host runs a process for moving itself, while the monitoring nodes run on both Dew and Fog hosts, according to their implementation. As in the previous case, the information that can be retrieved from the log files generated by the monitoring nodes on the Dew host is used for the considerations reported in Section 4.1.2. The last setup in which the platform has been tested involves a virtual machine and a Raspberry Pi. The Raspberry Pi has the role of the resource-constrained device and it can connect to the VM via Ethernet if it occupies a fixed position or via Wi-Fi if it moves. In this scenario, the two hosts exchange not only data for the monitoring nodes, but also image data. The video acquisition is enabled on both hosts: the Dew captures video frames and sends them to the Fog host. The tests performed using an Ethernet connection allows us to test the platform in a controlled scenario in which is possible to fine tune the variables of interest. The tests using Wi-Fi are on the other hand in uncontrolled conditions. The information used for the results chapter of this work is retrieved from the log files generated by both main and monitoring nodes. In this case we are not interested only in the resource usage, but also in the operating conditions of the main nodes, here is why we need other information such as the width of the captured video frames. Further details about these experiments are provided in Section 4.1.2.

DewROS2 in SHERPA

One of the purposes of rescuing missions is finding survivors after a natural disaster in an alpine environment. The UAVs employed by SHERPA are therefore sent in the location of the calamity to capture videos and images in order to find dispersed people. The huge amount of information captured cannot be easily analyzed by a human operator, who will easily become the bottleneck of the system performance. Exploiting the computational resources of a Cloud service can surely help. Cloud services for video analysis have now reached a high level of maturity and can identify several objects in a video, providing so called labels referring such identified objects. The request to obtain such labels for a video is called annotation request. Capturing videos at high definition is necessary for good video analysis results, but such videos involve a large volume of bytes, which cannot be easily transferred from the drone (the Dew host) to

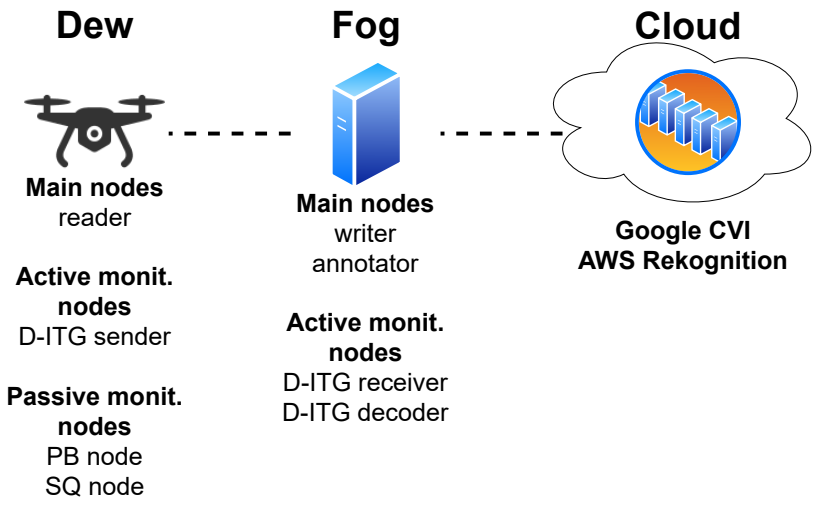


Figure 3.13. Nodes for the DewROS2 platform in SHERPA.

the ground (a Fog host), especially in alpine scenarios, where the network conditions are not optimal and largely variable while the drone is moving around. DewROS2 can provide several benefits in this scenario. Our monitoring nodes can continuously check the network conditions and its performance parameters in order to maximize the video quality according to such conditions and to capture videos at the highest resolution possible. The information about the network performance metrics is shared from the monitoring nodes to the main nodes involved in the video capture, which can then decide the best quality of the video captured to be transferred towards the Cloud for the analysis.

For this use case, we devise a solution made of three main nodes that execute video acquisition and processing, and five monitoring nodes [166]. The nodes are described in the following and a scheme of the machines which they are deployed on is in Fig. 3.13. The three main nodes that execute the video analysis are: a *reader* node that captures video frames from a camera, a *writer* node that receives the frames and writes them in a video file, and an *annotator* node that receives the video to analyze and sends an annotation request to a cloud service. The reader node is deployed on the Dew host and the other two nodes run on the Fog host,

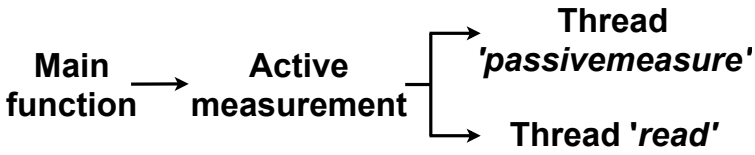


Figure 3.14. Reader node diagram.

while the most resource demanding task is performed by the cloud services. The monitoring nodes are launched by the reader node and influence its behaviour providing it with useful information. Active and passive monitoring nodes with different goals are used in this use case. The active monitoring nodes evaluate the initial value of the achievable throughput between the Dew and the Fog hosts before the frames capture begins. The passive monitoring nodes are the PB and the SQ nodes already mentioned in Table 3.2. This second group of nodes works at the same time of the frames capture and provides the information that the reader node will use to change its operating conditions. In the following we report a more detailed description of the main and active monitoring nodes.
The reader node is responsible for starting the active monitoring, the passive monitoring and the frames capture. The main function of the script initializes the node, starts the active measurement and then, when the active measurement finishes, launches two threads: one for the passive measurement and one for the frames capture (Fig. 3.14). The active monitoring is done by three AT nodes that use D-ITG and that work as follows:

- **receiver node**: on the Fog host, it calls the D-ITG Receiver entity and starts listening;
- **sender node**: on the Dew host, it calls the D-ITG Sender entity which sends packets to the receiver node;
- **decoder node**: on the Fog host, it calls the D-ITG Decoder entity to analyze the log file stored during the experiment, retrieves the measured throughput value and saves it on the ROSparameter server to make it accessible to the other nodes.

The initial values of the video resolution (i.e. the frame size) are decided on the basis of the measured achievable throughput. Once the active moni-

toring is completed, the reader node instantiates two threads. One of them launches the two passive nodes: PB and SQ. The size value measured by the SQ node will be sent over a ROS topic and used by the reader node to increase or decrease the video resolution. The second thread launched by the reader node is responsible for frame capture. For the frame capture we used OpenCV, an open source computer vision and machine learning software library. The reader node creates an OpenCV *VideoCapture* object, sets width and height according to the throughput value determined by the active nodes and starts capturing. Every frame captured using *VideoCapture* is converted in a CompressedImage message using the JPEG compression and sent over a topic to the writer node. While the node is capturing video frames, it periodically checks if a new message from the second passive node is available. If a new message is not available the node will keep on capturing, otherwise it will check the received queue size value and decide if it is necessary to change the video resolution using a simple adaptive algorithm. The algorithm keeps track of the resolution currently used: if the received queue size value is zero, the resolution will be increased to the next one available; if the received queue size value is larger than 1 MB, the resolution will be decreased. The resolution remains the same if the received queue size value is included between 0 and 1 MB. The set of possible resolutions that the camera can employ is chosen according to the available hardware, and it is comprised between 160x120 and 640x480 in our experiments.
The writer node runs on the Fog host. After its initialization, the node subscribes to the topic over which the reader node sends the frames in a *CompressedImage* message. Every time a new frame is received, the callback function converts the CompressedImage in an OpenCV image and accesses its width and its height: we need to check the dimensions of every frame received to decide whether it is possible to write the new frame on the current video chunk or if it necessary to open a new one. The first frame received is saved in a video file, for all the others frames the script compares their dimensions to the previous frame received: if the dimensions are the same, they are saved in the same file (up to the maximum number of frames we want to have in a video chunk), otherwise a new file is opened. The video files are created using the OpenCV *VideoWriter* object. Every time a new *VideoWriter* object is created, the previous one

is closed, its corresponding video file is saved and it is possible to send its name to the annotator node.
The annotator node runs on the Fog host and works with videos saved locally. We implement two different annotator nodes that use two different Cloud services: one for Google Cloud Video Intelligence and one for Amazon Web Services Rekognition. The common part of these two nodes is that they subscribe to a ROS topic over which the writer node will transmit the name of the video file to upload and analyze. The Google annotator can communicate with the Google servers thanks to the *videointelligence* library provided by Google, while in the Amazon annotator we need to import the *boto3* library to exploit the AWS functionalities. The result of the annotation process is a text file containing: the labels (objects identified in the video), the confidence level of the labels, and the time the object is detected in the video.

3.3 Secure task distribution

The experiments for the classification of malicious entities are conducted on a testbed comprising eight Nordic nRF52840 DK. The nRF52840 DK is a versatile single-board development kit for BLE, Bluetooth Mesh, Zigbee, and 802.15.4 [167, 168]. Each nRF52840 DK has a 64 MHz CPU, 256 KiB of RAM and 1 MiB of programmable flash. The several nRF52840 DKs in our testbed have different roles, according to the system architecture presented in [147]: one as root node, two as resource-rich Edge nodes, and five as resource-constrained sensor nodes. All the boards are in each other's communication range and are connected to their own Raspberry Pi to log output. The root node provides network services such as edge node discovery, while the edge nodes run the applications which require additional computational capability. The operating system running on the boards is Contiki-NG, an open-source, cross-platform operating system for IoT devices [169]. It focuses on dependable low-power communication and standard protocols, such as IPv6/6LoWPAN, RPL, and CoAP. The implementation of RPL employed in Contiki-NG allows the use of only one border router acting as a root node: we then assume that the single root node is secure and trustworthy. The system frequently publishes capabilities, generates a monitoring task, and generates a routing task. The

tasks generated by the sensor devices will be submitted to resource-rich devices that have the capabilities to calculate their results. The tasks are independent, hence a task submitted by one IoT device does not depend on a different task submitted by another IoT device.
As IoT nodes can become subject to a wide variety of attacks, the behaviour of a node can suddenly change and it may not work as expected. This way, it will not accomplish the tasks it is supposed to perform. This may represent an important problem since it compromises the integrity and the security of our system. The variety of possible attacks on IoT systems is very wide, hence the variety of unexpected behaviours is wide as well. Obviously, one of the possible indicators that show whether an edge node is not behaving as expected is the change in the received throughput. If the throughput decreases, we can assume that the edge node has been hacked, hence we can not consider it as a reliable component of our system.
The approach we propose for the classification of malicious nodes is to collect the n throughput observations $T_{1:n}$. From these observed values, we maintain a sequence of running means $\{\mu_{\text{ave}}\}_{1:n}$ and variances $\{\sigma^2_{\text{ave}}\}_{1:n}$, and a sequence of running exponentially weighted moving averages (EWMA) $\{\mu_{\text{ewma}}\}_{1:n}$ and variances $\{\sigma^2_{\text{ewma}}\}_{1:n}$ weight by parameter α. There is no need to save the sequence of $n-1$ values in the memory of the IoT devices, it is important to maintain only the latest mean and variance. A sensor node has to keep the values of the running mean, the EWMA, and their variances:

- for every application running on it (there is only one application running in our case, but they can be multiple in general);
- for both the directions (incoming and outgoing);
- for every edge node available in the network (typically more than one).

For example, in our case with one application, two directions and two edge nodes, the sensor nodes will have to store the values of mean and variance for four different distributions of running mean and four different distributions of EWMA. We will have a couple of normal distributions for every application, for every direction, and for every edge node, modeled as

in the following:

$$\mathscr{X} \sim N(\{\mu_{\text{ave}}\}_n, \{\sigma^2_{\text{ave}}\}_n) \tag{3.1}$$

$$\mathscr{Y} \sim N(\{\mu_{\text{ewma}}\}_n, \{\sigma^2_{\text{ewma}}\}_n) \tag{3.2}$$

A sensor node has to collect also the value of running mean and variance for another distribution, that we call global, that does not discriminate between the different edge nodes, but that will be useful to get information about the performance of the whole system. In our case with one application, the sensor node will keep track of other two distribution, one for incoming and one for outgoing.
We use these distributions to calculate the indexes of quality we need. The first index is calculated by comparing a running mean for a specific application, a specific edge node, and a specific direction, with its corresponding EWMA. We calculate a function G defined as in the following:

$$G = \begin{cases} 0.5 & \text{if } n = 0 \\ 1 & \text{if } \{\sigma^2_{\text{ewma}}\}_n = 0 \wedge \{\mu_{\text{ave}}\}_n \geq \{\mu_{\text{ewma}}\}_n \\ 0 & \text{if } \{\sigma^2_{\text{ewma}}\}_n = 0 \wedge \{\mu_{\text{ave}}\}_n < \{\mu_{\text{ewma}}\}_n \\ Pr(\mathscr{Y} > \{\mu_{\text{ave}}\}_n) & \text{otherwise} \end{cases} \tag{3.3}$$

This equation handles two special cases: when $n = 0$, that is no measurement has been made before, and when the EWMA variance is 0. If we calculate the average of G for the outgoing direction and G for the incoming direction, we will obtain the first index of quality, that we called *goodness of throughput*. The second index of quality is calculated by comparing a running mean with its corresponding global distribution. In this case we calculate $1 - G$ for the incoming and for the outgoing direction, we calculate the average between these two values and obtain the *global goodness of throughput*.
The goal of our experiment is to run an application on sensor nodes that requires the support of edge nodes in order to be correctly executed. The exchange of data between the nodes will generate a throughput that is exploited for the classification of the edge nodes in good or malicious. The considered application running on our IoT sensor nodes in our tests asks for a fake request to the edge node and gets a reply from an edge node.

Every time this application is run, it records the values of incoming and outgoing throughput and updates the running mean, the running EWMA, the global mean, and their variances. After updating these distributions, the sensor node calculates the two goodnesses of throughput we have mentioned earlier. All the values of these distributions are also stored in log files that allow us to carry out the analysis of the outcome of the experiment. All the edge nodes in the system are considered good at the beginning of an experiment, but at every execution of an application these indexes are updated. The condition that must be verified in order to classify an edge node as malicious is $GOT <= 0.25 \wedge globalGOT < 0.5$. These conditions are verified when the incoming throughput from one edge node starts to be lower to the ones registered from the other edge nodes. In this way, it is possible to discriminate the malicious entities in our system and guarantee security to the communications of our applicaiton.

Chapter 4

Results

This section will show all the outcome of our experimentations. The tests here described respond to the questions stated at the beginning of the work, going from an overall assessment of an IoT system based on LPWAN to the isolation of corrupted nodes from the legitimate components of an IoT system.

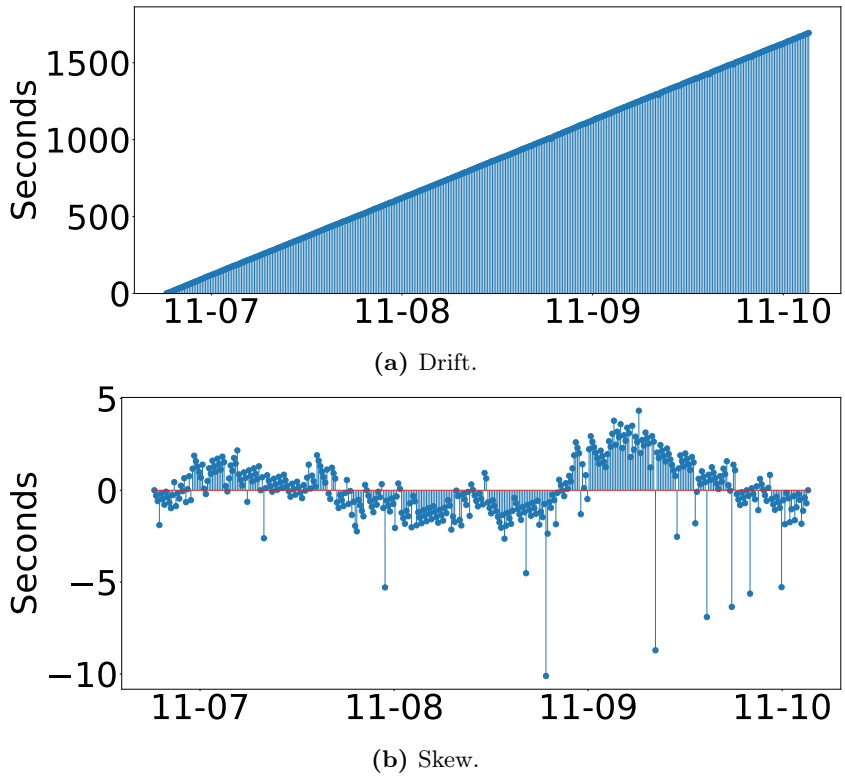
4.1 Performance evaluation and monitoring

4.1.1 Empirical performance evaluation of LPWANs

We report the results of our tests in terms of the metrics previously mentioned in Table 3.1.

Clock accuracy

The first consideration we can make from our results regards the accuracy of the internal clock of the device. In the first tests carried out, the device connects to a NTP server only the first time it wakes up and then sends messages using the Sigfox network. We notice how the three timestamps (timestamp sent, timestamp of the event at the Sigfox Cloud, and timestamp of the arrival of the message at our backend) of the first message are correctly ordered. From the second message onward however, the timestamp sent from the device is subsequent to the other two timestamps. This behaviour is observed with both the devices and it is valid



(a) Drift.

(b) Skew.

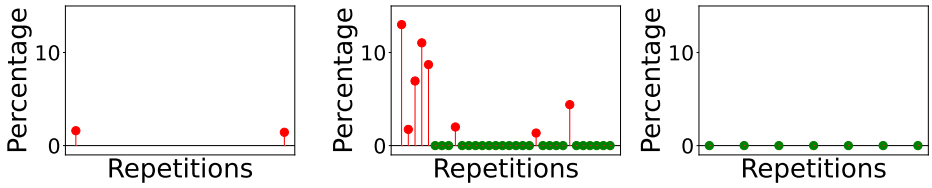
Figure 4.1. Drift and skew using Sigfox.

for all the communication technologies. The explanation for this effect is that in our system there are different clocks running at different speeds on different machines: the Pycom device, the Sigfox Cloud and our backend server. In the plots of Fig. 4.1a we can notice how the differences between the timestamps change during the tests. This graph is obtained from one of the first tests using Sigfox on the first device, but we observe the same behaviour with the other device and other network technologies as well. In this plot we report the difference between the timestamps sent and the ones at our backend. This difference is called drift. We can notice that these differences grow according to a linear relation. The slope of the line that interpolates the first and the last points of the plot in Fig. 4.1a is

the ratio of the difference between the drift at the last and at the first moments and the time elapsed between the two moments considered: the slope is then the average rate at which the two clocks diverge. The slope calculated is $5.816 * 10^{-3}$: this means that the difference between the two clocks varies by 5.8 ms every second. If we consider the whole duration of the experiment (around 3 days and 9 hours, that is 291600 seconds), we can calculate that the estimated difference between the two clocks is 1691 seconds, that is close to the value actually measured.
The plot of Fig. 4.1a shows the average difference of the two speeds, while the plot of Fig. 4.1b shows the instantaneous difference of the two clocks. This instantaneous difference is called skew and it is calculated by subtracting the value of the ideal line from the actual difference between the two timestamps. This instantaneous variation is caused by different transmission and elaboration times. The plot of Fig. 4.1b shows that the skew can assume positive or negative values. If we consider that the messages are sent every 12 minutes, we can say that the average misalignment between the two clocks every 12 minutes is 4.2 ± 10.1 seconds, where 4.2 is the average misalignment in 12 minutes ($12 * 60 * 5.816 * 10^{-}3 = 4.2$) and 10.1 is the absolute value of the maximum skew measured (as shown in Fig. 4.1b). **These results show that the two clocks distance themselves of** 5.8 ± 0.01 **milliseconds per second**, where 0.01 is the maximum skew in a second ($10.1/(12 * 60)$).
Drift and skew can be avoided by connecting to a NTP server every time the device wakes up: in this case we do not observe strange trends and the timestamps are correctly ordered.

Energy efficiency

Sigfox and LoRaWAN have similar performance, with Sigfox being the most efficient with around 395 sent messages in different repetitions and LoRaWAN the second most efficient with around 375 messages. Several setbacks are faced when carrying out the same experiment using NB-IoT. In this case, the Pycom device becomes unstable and is not able to exploit completely the battery because it gets stuck during its operations. After numerous repetitions, it becomes evident that the operation that causes the interruption of the normal operations is the connection to the NB-IoT network. The device uses some proprietary functions to connect to the



(a) LoRaWAN **(b)** NB-IoT **(c)** Sigfox

Figure 4.2. Percentage of lost messages.

network: one of these is *pybytes.connect()*, which is not always correctly executed. When this instruction is not correctly carried out, the device can behave in two different ways: 1) it hangs when trying to connect to the network, 2) it is restarted by an internal watchdog after 21 minutes of inactivity. However this second possibility does not guarantee that the device resumes its normal operation: it is actually very likely that the *pybytes.connect()* operation will remain unstable until it is manually powered down. Our analysis also highlights that when the device hangs during the connection to the network, the battery is quickly drained. The device has a stable behaviour in only one of the many repetitions performed, and the number of messages sent is 200, which is lower than the number of messages sent using the other two technologies. **The results show that Sigfox has the lowest consumption and that NB-IoT has the worst performance.**

Message losses

The message loss ratio is 0% in all of our tests with Sigfox, while we always record a ratio lower than 2% when using LoRaWAN. As stated in Subsection 4.1.1, the device does not show a stable behaviour when using NB-IoT. After several repeated experiments to understand how to make it work better, the performance shows an improvement. The message loss ratio is around 10% in the first tests, but it then decreases to 0% for the 88% of the repetitions. The results in Fig. 4.2 show that **Sigfox is the best communication option in terms of messages correctly delivered**.

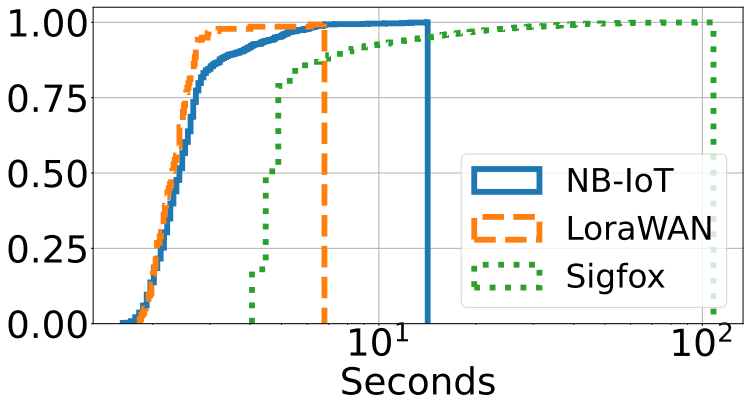


Figure 4.3. CDF of the message latencies.

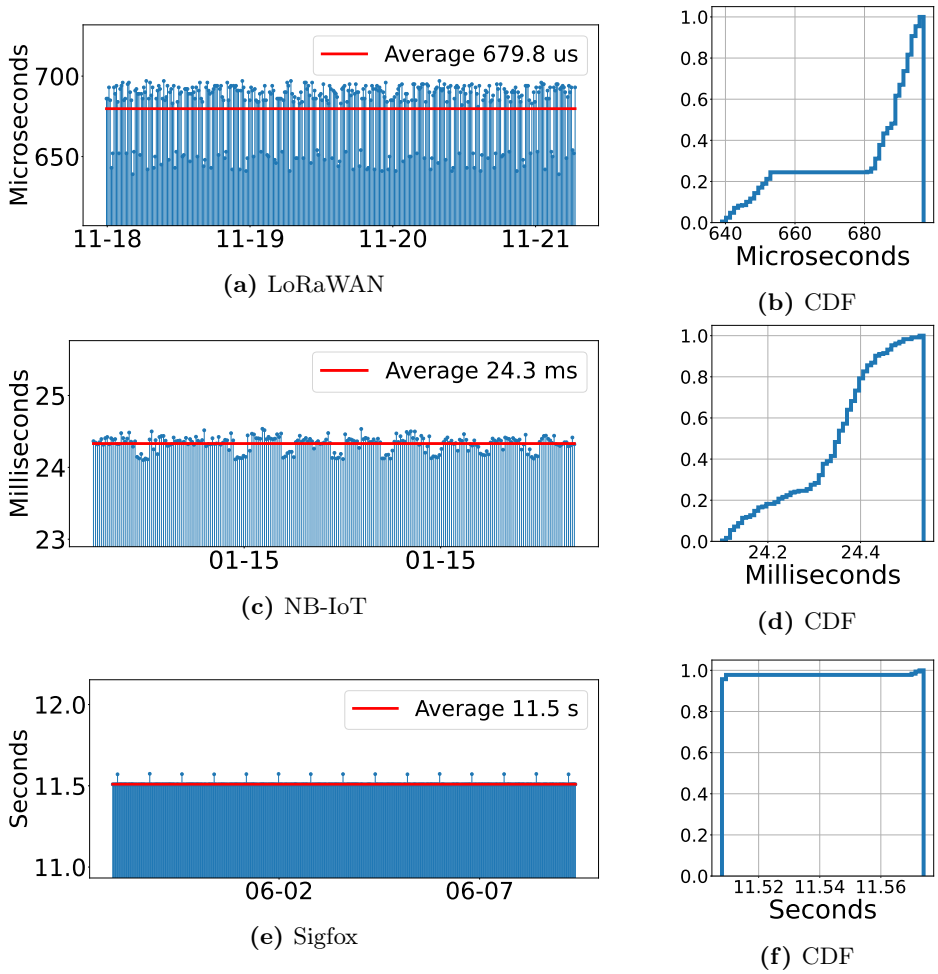
Table 4.1. Analysis of the latency for the three technologies.

	Min	Avg	90th perc.	Max
LoRaWAN	1.8 s	2.4 s	2.7 s	6.8 s
NB-IoT	1.6 s	2.7 s	3.7 s	14.1 s
Sigfox	4.1 s	6.3 s	8.0 s	108.3 s

Time analysis

We report the CDF of all the latencies measured in all the experiments in Fig. 4.3 while significant values are reported in Table 4.1. LoRaWAN presents the lowest average value, 90th percentile, and maximum value (respectively 2.4, 2.7, and 6.8 seconds). NB-IoT shows the lowest minimum latency (1.6 seconds), while the worst results are obtained when using Sigfox. In this latter case, the 90th percentile is around 8 seconds while the maximum value exceeds 100 seconds, a value that is not measured with other network technologies. If we consider the maximum values measured and the error caused by the clock inaccuracy, we have maximum latencies of $6.79 \pm 0.04s$ for LoRaWAN, $14.15 \pm 0.08s$ for NB-IoT and $108.26 \pm 0.62s$ for Sigfox. These results show that **the values measured using Sigfox are higher than the ones measured with the other two technologies, which have smaller and less variable latencies.**

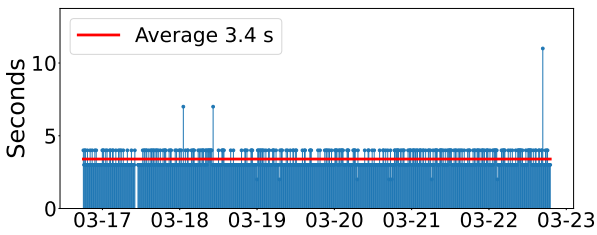
Another interesting parameter to evaluate is the **duration of the**



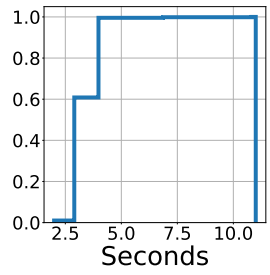
(a) LoRaWAN **(b)** CDF **(c)** NB-IoT **(d)** CDF **(e)** Sigfox **(f)** CDF

Figure 4.4. Duration of the transmission of a message.

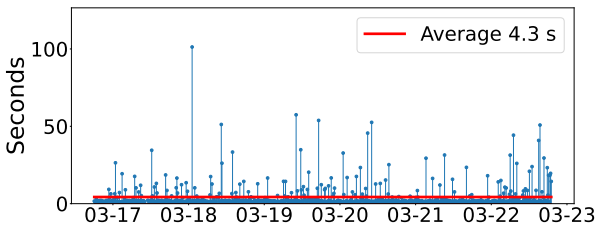
transmission of a message. The average transmission of a message with Sigfox shows a value of around 11.5 seconds for all the repetitions of our tests. Moreover a peak value is repeated every 47 messages in every repetition. The transmission of a message is way shorter for LoRaWAN and NB-IoT: with the former the average is around 680 microseconds while for the latter the average is around 24 milliseconds. The values retrieved



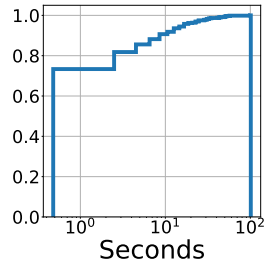
(a) Difference: timestamp Sigfox event - timestamp sent.



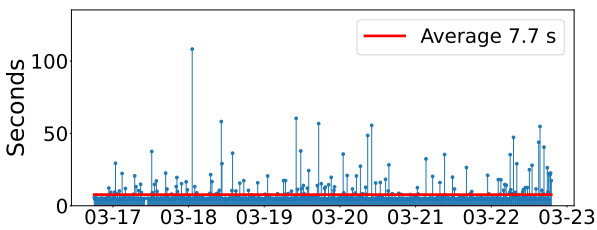
(b) CDF



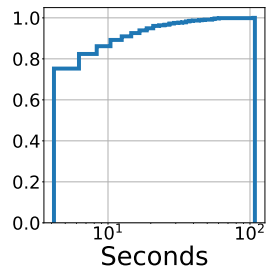
(c) Difference: timestamp server - timestamp Sigfox event.



(d) CDF



(e) Difference: timestamp server - timestamp sent.



(f) CDF

Figure 4.5. Latencies in an experiment using Sigfox.

from one of the tests for every technology are reported in Fig. 4.4.

Understanding Sigfox latency

It is possible to perform a more detailed analysis of the latency of a message when using Sigfox. In this case we can retrieve the timestamp of the arrival of a message on the Sigfox Cloud, so it is possible to know

the time elapsed between the sending of a message and its corresponding Sigfox event and the Sigfox event and the reception of a message on our backend. The results obtained with both the devices, in repeated experiments, in different conditions, are very similar and the outcome of one of our experiments is reported in Fig. 4.5. The wireless channel between the device and the gateway seems to have a stable behaviour: the majority of the ts_sigfox-ts_sent intervals are indeed included in the (2; 4) seconds intervals, and only in very rare cases there are outliers (Fig. 4.5a). The communication between the Sigfox Cloud and our backend introduces large latency and variability (Fig. 4.5c). For the test reported in Fig. 4.5, the minimum value of the difference between the timestamp at our backend and the timestamp at the Sigfox Cloud is 0.48 seconds, the maximum is 101.26 seconds, while the 90th percentile is 10 seconds. From multiple experiments, it is possible to notice how these outliers do not follow a particular trend and their distribution is random. **The variability introduced between the Sigfox Cloud and the final destination may severely impact communications in critical applications.**
In the Sigfox case, only the time measured between the device and the Sigfox Cloud is subject to the error due to the inaccuracy of the clock, while we can assume that the two clocks on the Sigfox Cloud and our backend server are more accurate. In this case, the maximum is 11 ± 0.06 seconds.

4.1.2 Performance monitoring

The proposed platform for monitoring of CPSs has been tested for different purposes. We are going to illustrate the impact and the effectiveness of this platform at first, showing that it is really feasible to deploy it on a CPS. We are then going to evaluate the benefits of using it and the improvement of performance that its usage can bring.

Impact of the monitoring nodes on the resources

In this section the tests carried out to evaluate the impact of the monitoring nodes on the resources of the machines employed are described. Evaluating their impact is important, since the goal is to deploy these monitoring entities on resource constrained devices, that can sustain only a limited amount of computation. The utilization of the resources when

Table 4.2. Description of the scenarios of the tests evaluating the impact of the monitoring nodes.

Scenario	Machine	Monitoring nodes	Log level	Nodes launch	Video capture
1	Only ROS master running				
2	Dew/Fog	Passive	1	Sequential	Disabled
3	Dew/Fog	Passive	2	Sequential	Disabled
4	Dew/Fog	Passive	1	Parallel	Disabled
5	Dew/Fog	Passive	2	Parallel	Disabled
6	Dew	Active	1	Sequential	Disabled
7	Fog	Active	1	Sequential	Disabled
8	Dew	Passive	1	Sequential	Enabled
9	Fog	Passive	1	Sequential	Enabled

only the monitoring nodes are running is compared with the utilization of the resources when both main and monitoring nodes are active.
In these tests two virtual machines are used, one acting as the Dew host and one as the Fog host. Each of these two VMs uses 2 GB of RAM and 1 core of the physical CPU. The two machines communicate through their Ethernet interface and are performing video acquisition: in particular one machine captures video frames and sends them to the other VM, which saves them in a video file.
These two machines are equipped with the monitoring nodes that are reported in Table 3.2. The various scenarios and configurations employed in our experiments are described in Table 4.2. The scenarios differ from each other in terms of which monitoring nodes are used, how they are launched, which log level is employed, and whether the main nodes are active. In this way, we have different scenarios with different resource usage to evaluate the impact of the monitoring nodes in different conditions on the resources employed. Four parameters of interest are monitored with external tools for evaluating the usage of the resources on our machines while doing the experiments:

- CPU usage, using the tool *top*;
- memory usage, using the tools *top* and *free*;

Table 4.3. Resource usage in the tests evaluating the impact of the monitoring nodes.

Scenario	CPU usage	Memory usage	Disk input	Network (incoming)	Network (outgoing)
1	1.2 %	5.5 %			
2	1.9 %	5.4 %	20.4 kBps	11.0 kbps	46.5 kbps
3	2.0 %	5.4 %	36.0 kBps	1.8 kbps	25.4 kbps
4	3.5 %	5.4 %	34.4 kBps	12.9 kbps	47.7 kbps
5	3.2 %	5.4 %	34.4 kBps	2.7 kbps	26.7 kbps
6	3.4 %	3.5 %	0	23.6 kbps	49.8 Mbps
7	3.4 %	3.4 %	0	49.0 Mbps	23.5 kbps
8	20.0 %	5.9 %	31.7 kBps	69.9 kbps	22.6 Mbps
9	90.0 %	8.5 %	53.4 kBps	7.7 Mbps	120.4 kbps

- disk I/O usage, using the tool *iotop -o*;
- network usage, using the tool *nload*.

The goal of these tests is to determine whether the monitoring nodes introduce an important overhead that can not be sustained by devices with constrained resources, and the results are reported in Table 4.3. In the first scenario the machines are not overloaded since their only task is to run the ROS master: this scenario is the baseline for our comparison. Results reported in Table 4.3 show that the values of CPU, memory, and disk usage are low when the video capture is disabled, or in other words when the main nodes are not working. The values for CPU and memory usage increase only when the main nodes are active (scenarios 8 and 9). On the other hand they do not increase when only the monitoring nodes are active (scenarios 2-7). In conclusion, the monitoring nodes do not have a significant impact on the resource consumption. Regarding the network, we notice an increase of the bitrate in the last four scenarios of Table 4.3. In particular, in scenarios 6 and 7 we use the active nodes employing iperf3 limited at 50 Mbps (the maximum bitrate we expect in our video capture application). In scenarios 8 and 9, the recorded throughput is high since the video frames captured by the Dew machine are transferred to the Fog. This explains the high usage of the network. In the other scenarios the

load on the network is very limited, meaning that the passive monitoring the nodes do not flood the network for exchanging their information. The minor overhead introduced by the monitoring nodes in terms of CPU, memory, and disk usage means that these nodes do not occupy a large slice of the available resources and can be deployed even on hosts with constrained hardware and computational power.

Effectiveness of the monitoring nodes

The goal of this section is assessing the effectiveness of our monitoring platform, comparing the expected behaviour of the measures of interest (e.g. CPU utilization) with the values reported by the monitoring nodes. This assessment is performed by deploying and testing DewROS2 on real machines.
In these experiments, we introduce a Watchdog script on the Dew host that allows to keep under control all the monitoring nodes. The controls concern if the nodes are running, if they are blocked and whether the Dew host has lost the Wi-Fi connection. The Watchdog can control if the nodes are running by checking the PIDs (process identifiers) sent on a dedicated topic. In particular there are two different possibilities: the node is inactive if the PID is missing, or it is blocked if the node is not properly working but the PID is still present. If a node is inactive, the Watchdog restarts it using a 'roslaunch' command. If a node is blocked, the size of the log files does not increase and the Watchdog then kills the respective process. To check if the Dew host has lost its Wi-Fi connection, the Watchdog retrieves the SSID using the *iwconfig* command and suspends its controls.

A **first series of experiments** involves a drone as a Dew host, while an Ubuntu virtual machine running on a PC is used as the Fog one. The PC also works as the access point which the drone is connected to. All the six monitoring nodes mentioned before are enabled.
In this experiment, we move the drone in our laboratory manually, without using its wings. The drone is moved in different positions, at different distances from the PC and stays in the same position for ten minutes, as shown in Fig. 4.6.
Two different computing boards are connected to the drone in this series of tests: a Px4 board, a popular general purpose flight controller; and an In-

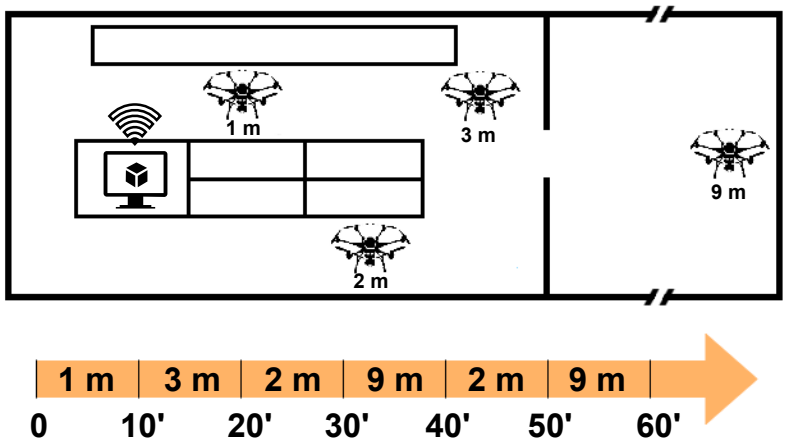
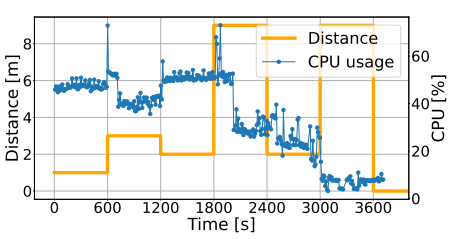


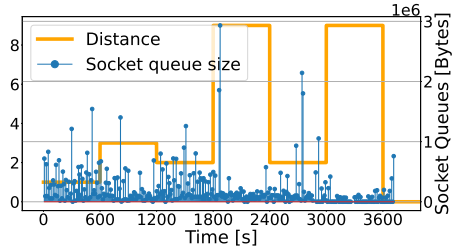
Figure 4.6. Position of the drone and timeline of its movements inside the laboratory.

tel Up Squared, used for running the monitoring nodes and for performing SLAM, the activity of constructing and updating the map of an unknown environment while also keeping track of the location occupied in such map. We also attach a laser sensor to develop 3D scans of the environment and build the map.

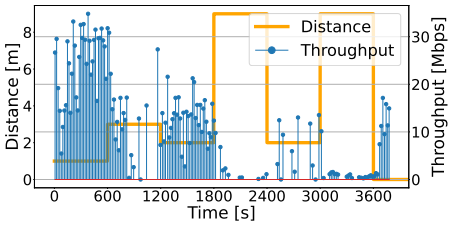
During the SLAM activity four significant processes are executed: octomap_server (map generator), node_let (process for reading the measures from the laser), mavros (process for reading data from the autopilot), and mapping (process for performing SLAM). The values of CPU utilization collected by the CU monitoring node are reported in Fig. 4.7a. In the case of SLAM, we expect that the CPU usage varies according to the environment analyzed: in particular it should be high when numerous objects are present in the surroundings and it should be lower in the case of a plain and simple environment. The CPU utilization should also vary according to the previous knowledge of the location, with lower values when the environment has already been analyzed. As shown in Fig. 4.7a, the expected behaviour is actually monitored by the CU node. In correspondence of 600, 1200 and 1800 seconds, we can notice peak values when the drone moves to an unknown environment. Another interesting aspect is that the first time the drone is 2 and 9 meters away from the access point,



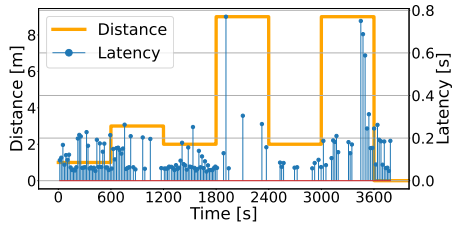
(a) CPU utilization.



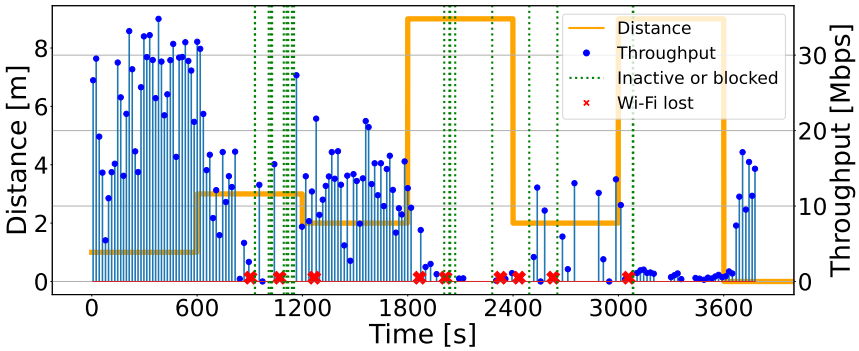
(b) Socket queue sizes.



(c) Achievable throughput.



(d) Latency.



(e) Achievable throughput, inactive and blocked nodes, Wi-Fi connection lost. The blue stem plot is the achievable throughput, the red crosses are the points in which the Wi-Fi connection is lost, the green dotted lines represent the moments when the nodes are inactive or blocked.

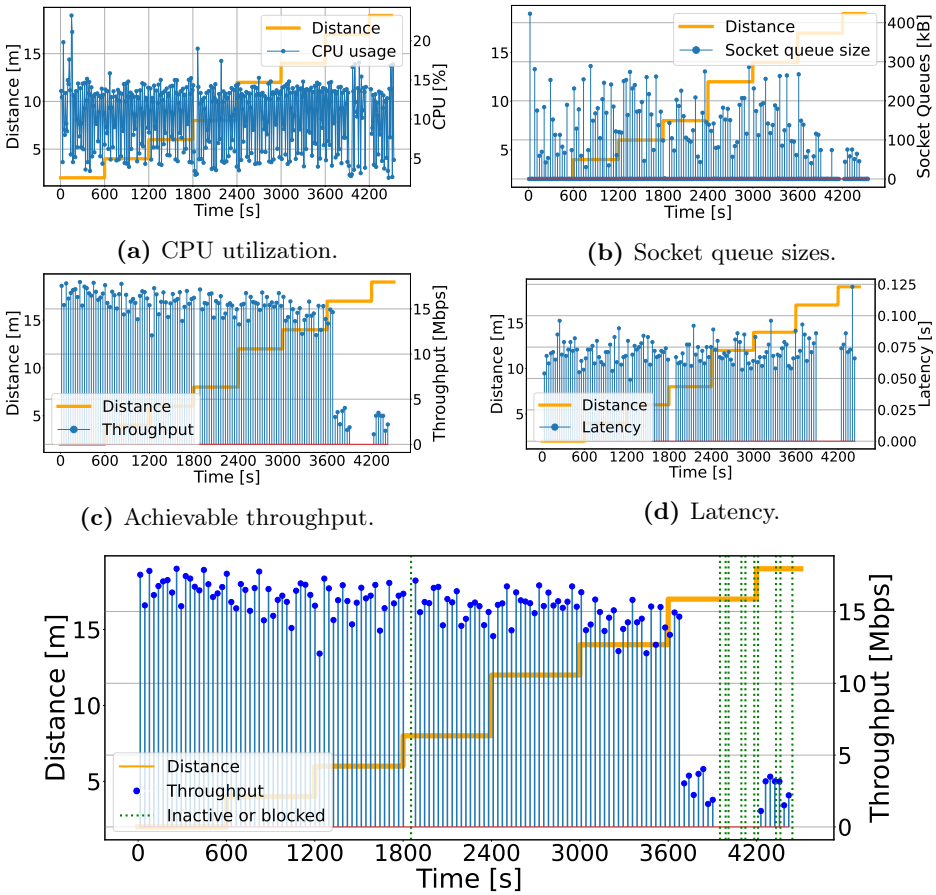
Figure 4.7. Results obtained in the experiments with the drone.

the CPU usage is higher than the utilization retrieved the second time the drone occupies the same positions. In particular, the values retrieved for the 2 meters positions are higher than the values at 9 meters because of

the complexity of the surroundings, as we expected.
The behaviour of the waiting queues arising on the drone is reported in Fig. 4.7b, in which the values collected by the SQ node are reported. These queues arise because the AT node sends packets on the network for measuring the achievable throughput. The measured sizes of the queues are larger when the SQ node is activated at the same time of the AT node, because in those occasions the SQ node detects the data sent by the AT node.
The achievable throughput is reported in Fig. 4.7c. We expect to measure higher values when the drone is closer to the access point and lower values the distance increases. The expected behaviour is actually observed in the figure and the highest values are recorded when the drone is only 1 meter away from the access point. The further the drone is from the access point, the more problems are experienced by the AT node, which is not able to obtain a measure. The values of achievable throughput are also affected by the interference present in our laboratory. A similar behaviour is expected for the latency between the two hosts, since distance plays an important role in determining this delay. This behaviour is confirmed by the plot of Fig. 4.7d: the peak values are indeed recorded when the drone is 9 meters away from the access point. Thanks to the watchdog, we are also able to check if the nodes are inactive or blocked and if the drone loses its Wi-Fi connection (Fig. 4.7e). The nodes that are often inactive or blocked are the active nodes. The Wi-Fi connection is lost several times in this experiment. When the Wi-Fi connection is interrupted, the watchdog script suspends all the monitoring nodes until the connection is restored. When the Wi-Fi connection is unavailable, we do not have any measurement value.
The results of this experiment prove the effectiveness of these monitoring nodes, since their outcome respects the expected behaviour of the hosts, in particular of the CPU usage and the achievable throughput.

In a **second series of experiments**, the hardware on which DewROS2 is tested is different. In this case, the Dew host is a Raspberry Pi mounted on a moving device, a Smart Video Car produced by the company Sunfounder [170], while the Fog host is an Ubuntu virtual machine running on a Cisco Connected Grid Router (CGR) 1120. The Dew host is connected

to the CGR through Wi-Fi. The Cisco CGR 1120 is a communication platform running the Cisco IOS operating system that allows to run third party applications and to host applications at the edge of the network. The IOx Connected Grid Module - System Server (CGM-SRV) is used in CGRs to execute applications or virtual machines [171]. The CGM-SRV module is actually a small server that contains a multi-core x86 CPU, memory and storage. This module can work as a Fog host at the edge of the network, bringing distributed intelligence to operational networks.
The task performed on the Raspberry Pi 4 connected to the car is called *picar_controller* and it is used to drive the car with the joystick. The CU, SQ, AT and AL nodes of Table 3.2 are active. The car moves across the laboratory occupying different positions at different distances from the router, as reported in Fig. 4.9. In this case, the Dew host does not perform a demanding activity such as video capture or an activity whose load varies over time like SLAM. We do not notice indeed a significant increase of the CPU utilization over time or at difference distances: the load of the operations executed on the Raspberry Pi remains stable (Fig. 4.8a). The results of the measurements of the SQ node are reported in Fig. 4.8b. The queues measured by the node are caused by the traffic sent by the AT node. Distance and physical obstacles between the two hosts lead to higher latency and lower throughput. In Fig. 4.8c and Fig. 4.8d we report respectively the result of the measurements of achievable throughput and latency. Up to a distance of 14 meters, the achievable throughput is always high. When the car is further than 14 meters away from the router and a wall obstacles the communication, the achievable throughput decreases drastically, while the latency shows a peak value.
In this experiment with the PiCar and the CISCO router, we face the same problem as in the previous experiment: the nodes that cause problems are the active ones. The nodes are inactive in several occasions when the PiCar is 17 and 19 meters away from the router (Fig. 4.8e). However the Wi-Fi connection is never lost: this is probably due to the fact that in the laboratory in which this experiment is carried out no other wireless devices are deployed, hence the PiCar does not suffer from much interference.
The outcome presented in this section shows how DewROS2 can be deployed on different hardware: from Ubuntu VMs to industrial routers, from boards for drones to Raspberry Pis. Moreover we have evidence that the



(a) CPU utilization.

(b) Socket queue sizes.

(c) Achievable throughput.

(d) Latency.

(e) Achievable throughput, inactive and blocked nodes. The blue stem plot is the achievable throughput, the green dotted lines represent the moments when the nodes are inactive or blocked. There are no red crosses because the Wi-Fi connection is never lost in this case.

Figure 4.8. Results obtained in the experiments with the commercial router.

measurements performed by the nodes reflect the expected results. For example, in the case of SLAM, the obtained results support the fact that the CPU utilization is indeed higher when the drone is in an unknown environment. Another example is the measurements of bandwidth and la-

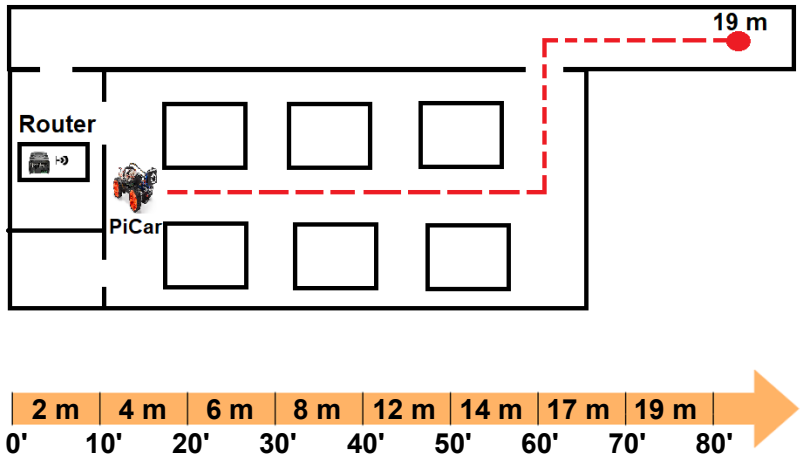


Figure 4.9. Position of the PiCar and timeline of its movements inside and outside the laboratory.

tency in the experiments with a commercial router. In this case we notice how the values get worse in the presence of an obstacle between the Dew and the Fog hosts. The effectiveness of these monitoring nodes brings advantages to the application in which they are deployed, since they allow to make informed decisions on the basis of the status of the system, ensuring optimization of the resources and of the performance.
The variety of hardware on which our solution can run and the accuracy of the measured values are undoubtedly a strength of our solution. The effectiveness of the monitoring nodes for making informed decisions will be shown in the following.

Benefits and increase of performance

To evaluate the benefits provided by our solution we test DewROS2 in controlled and uncontrolled conditions [166], using a wired connection in the former case and a wireless connection in the latter one. In both cases the Dew host is a Raspberry Pi, while the Fog one is an Ubuntu virtual machine running on a PC.

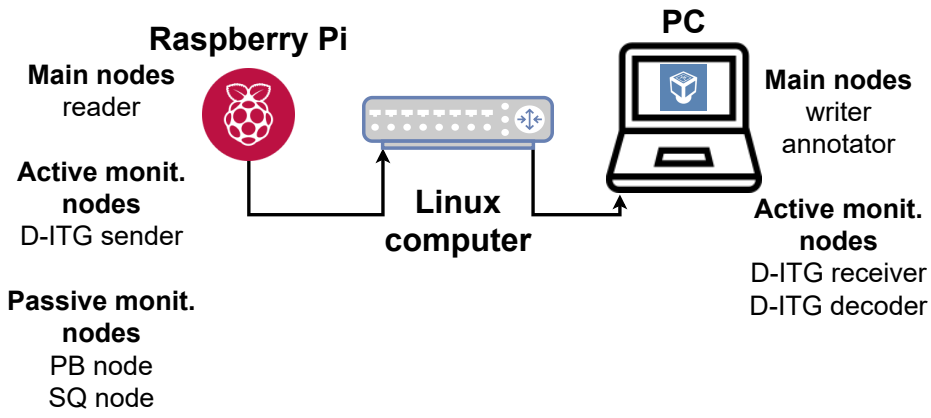


Figure 4.10. Testing in controlled conditions: employed devices.

For our **testing in controlled conditions**, we use an Ethernet connection and tools that limit the available bitrate on our network in order to test our adaptive algorithm. The employed tools are *tc* (traffic control) and *netem* (network emulator). *Tc* is used to configure traffic control settings in the Linux kernel and allows us to add a queuing discipline to our selected interface; *netem* is an enhancement of the Linux traffic control facilities that allows to add delay, packet loss and other characteristics to packets outgoing from a selected network interface. Specifically we use the *rate* option of *netem* to set the maximum bitrate outgoing from our selected interface. We carry out our experiments linking our Raspberry Pi (Dew host) to an Ethernet interface of a Linux computer and our PC (Fog host) to another (bridged) interface of the same computer as shown in Fig. 4.10. We use *tc/netem* on the Linux computer and periodically change the maximum outgoing bitrate, randomly choosing its value from a set of seven values ranging from 0.5 to 32 Mbps. We also test our system changing two parameters: the time period used in the script that sets a *tc/netem* constraint, and the monitoring period, used in the SQ monitoring node to communicate the aggregate value of the queues. Fig. 4.11 shows the results of a test lasting about 40 minutes, with a *tc/netem* period of 180 seconds and a SQ monitoring period of 5 seconds.

The blue line in Fig. 4.11 shows the values of bitrate imposed by *tc* and *netem*, the red dots show the bitrate measured by the PB monitoring

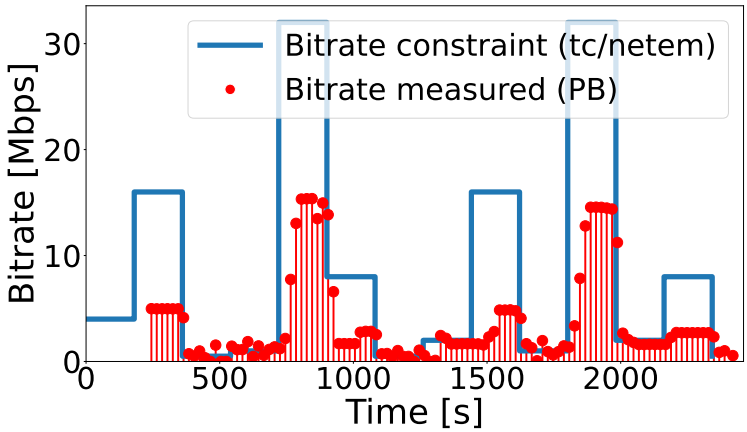


Figure 4.11. Testing in controlled conditions: bitrate values.

node on our Raspberry Pi. We can see how the outgoing bitrate follows the restrictions imposed by *tc/netem*. As shown in Fig. 4.11 the maximum outgoing bitrate we measure is around 15 Mbps: this is due to the fact that the maximum video resolution we use is 640x480 pixels. The constraint on the maximum resolution employed is imposed by the hardware resources of the Raspberry Pi camera employed in these tests, since it does not support the possibility to capture videos at high definitions. Using more powerful resources, it would be possible to capture video frames at higher resolution, produce a higher bitrate and get closer to the maximum bitrate available.

Fig. 4.12 shows how the video resolution changes according to the queue size. The blue line is the width of the frames, the red line shows the aggregate values of the queues: these are the values received by the reader node from the SQ node and are plotted using a logarithmic scale for the y-axis. The dashed black line is the threshold above which the video resolution is decreased. We can easily see that the width of the frame decreases every time there is a large queue while the width increases when the queue is absent. If we compare the two graphs we can notice a trend: when the available throughput imposed by *netem* is high, there are no queues, so the reader uses a higher video resolution; when the available throughput drops, the queue sizes increase, so the reader uses a lower resolution.
We carry out several experiments using three different values of *tc/netem*

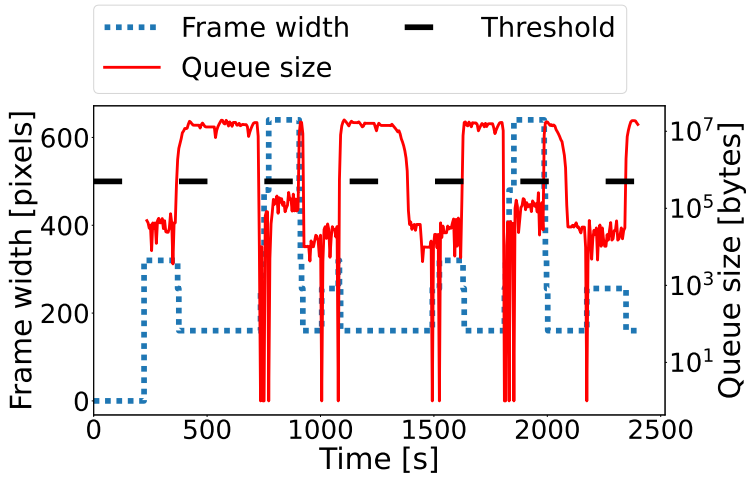


Figure 4.12. Testing in controlled conditions: frame width and queue sizes.

periods and three values of monitoring period for the SQ node. The traffic constraint changes every 30, 60 or 180 seconds while the three periods on the SQ node are 2, 5 or 10 seconds. For every combination of these values three experiments are performed and the difference between the area under the *tc/netem* curve and the area under the *PB* curve is calculated (see for example Fig. 4.11). We report the boxplots of these differences in Fig. 4.13. We are interested in the lowest possible value of these medians, so that we are exploiting the available throughput at its best. For the monitoring periods of 2 and 5 seconds we observe the expected trend: the medians of the differences between the curves are minimum when the channel changes slowly (that is when the traffic control constraint changes every 180 seconds), while they are maximum when the channel changes rapidly. In particular for the 2 seconds period we have the smallest differences since the passive node is more sensitive to channel variations. Therefore, our system can better follow the traffic control constraints when the monitoring period is smaller. The expected trend is not respected for the 10 seconds monitoring period: such interval is in fact too high and does not allow our application to follow the variabilities of the channel and to exploit the maximum available bitrate.

For every combination of the two time periods we also calculate the aver-

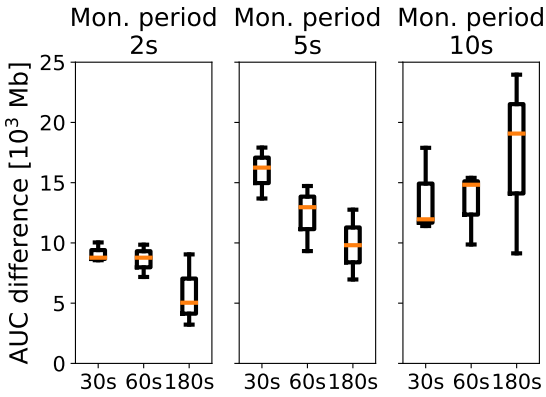


Figure 4.13. Testing in controlled conditions: boxplot of the average difference in thousands of Mbit between the area under the tc/netem curve and the area under the PB curve. The three channel variation periods are on the X-axis.

age error in Mbps between the available bitrate (*tc/netem*) and the produced bitrate (PB node). The boxplots of these values are reported in Fig. 4.14. From this figure it is possible to easily quantify the effectiveness of DewROS2 in using the available resources, that is, in this case, the amount of available that is actually used. To deepen this evaluation, we compare the results obtained with our monitoring infrastructure to the hypothetical result we would have obtained if we captured video frames at the lowest resolution possible on the Raspberry Pi camera, 160x120 pixels. We use this resolution since it allows us to have the lowest bitrate possible, and we can consider the transmission of this video as the worst case in our scenario. In particular, for every test performed, we calculate the average outgoing bitrate when we use the 160x120 resolution and then calculate the difference between the actual PB curve of our tests and the hypothetical curve we would have obtained if the Raspberry Pi had sent the captured video frames at the average bitrate calculated for the 160x120 resolution. The boxplots of these differences are in Fig. 4.15. We can see that the difference is smaller when the monitoring period is 10 seconds and it is larger when the monitoring period is 2 seconds: this means that we can exploit the available bandwidth and transfer more data when the

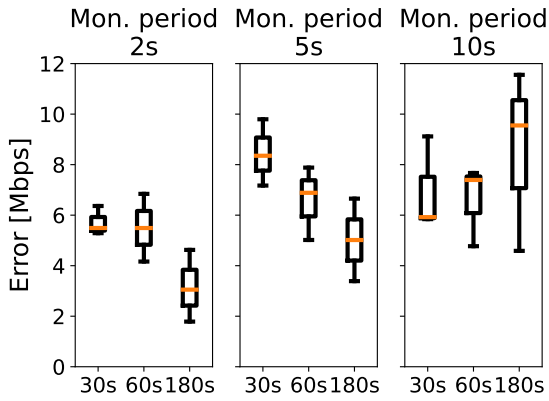


Figure 4.14. Testing in controlled conditions: boxplots of the error in Mbps between the available and the used bitrate. The three channel variation periods are on the X-axis.

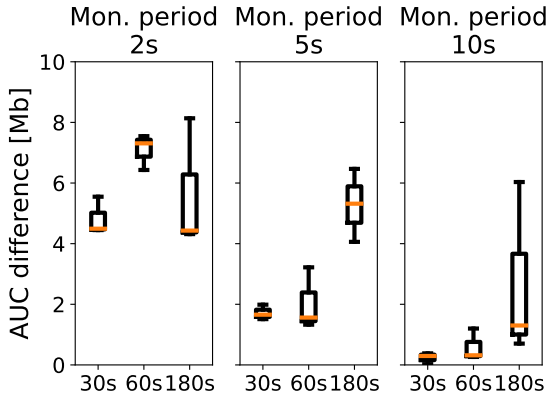


Figure 4.15. Testing in controlled conditions: error bar of the average difference in Mbit between the area under the PB curve and the area under the hypothetical ifconfig curve at 160x120. The three channel variation periods are on the X-axis.

queues are monitored more frequently. When the queues are monitored less frequently, the transmission is very similar to the one using the lowest possible resolution. This confirms that it is important to have a proper and properly tuned monitoring system.
We then calculate the relative error between the outgoing bitrate from the Raspberry and the limitation imposed by *tc/netem*. The difference to 1 of these errors can be interpreted as the resource utilization rate. The values are reported as percentages in Table 4.4a. We also calculate the relative error we would have obtained with constrained bitrate and capturing at the 160x120 resolution: the values are reported in Table 4.4b as percentages as well. Comparing the values reported in the two tables, we can see how the

Table 4.4. Testing in controlled conditions: percentage of usage.

	30 s	60 s	180 s
2 s	52.04	55.86	64.68
5 s	38.24	41.84	53.22
10 s	38.98	41.18	38.73

(a) Percentage of usage using DewROS2.

	30 s	60 s	180 s
2 s	26.41	36.68	44.49
5 s	30.75	35.31	34.51
10 s	33.93	34.18	33.35

(b) Percentage of usage without using DewROS2.

available bandwidth is better exploited when we introduce our monitoring platform. We notice this difference for every monitoring period, and the gaps between the usage percentages using a 2 seconds monitoring period are more substantial. The first row (monitoring period 2 s) of Table 4.4a and the first row of Table 4.4b show that the bandwidth usage is improved by more than 45% for every channel variation period using DewROS2. We can also see how the bandwidth usage doubles in the case (monitoring period 2 s, *tc/netem* period 30 s), increasing from around 26% in Table 4.4b to around 52% in Table 4.4a.

Our solution has also been **tested in uncontrolled conditions**, using Wi-Fi. We move our Raspberry Pi around an access point in order to have a variable channel. Our PC, on which the virtual machine is running, works as an access point in these tests, while the Raspberry Pi is mounted on a moving device. The employed device is once again the Smart Video

Car (Fig. 4.16).

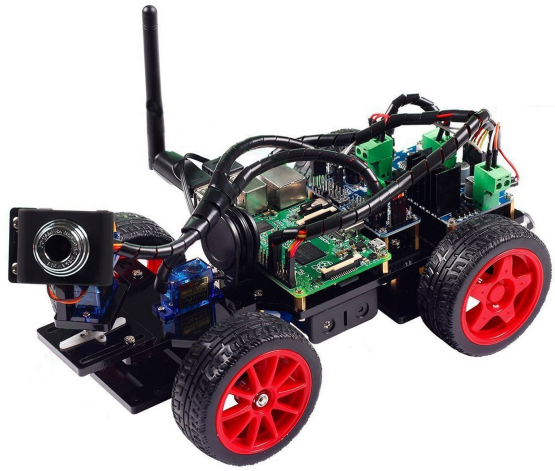


Figure 4.16. Sunfounder Smart Video Car Kit for Raspberry Pi [7].

For our tests we want the car to follow a predefined trajectory, so we develop a Python script that sends the same movement commands to the car.
The first experiments are carried out inside our laboratory. Fig. 4.17 shows a map of the laboratory and the trajectory followed by the Smart Video Car. This experiment is repeated three times but we report only the results of the first repetition, since the three results are similar.

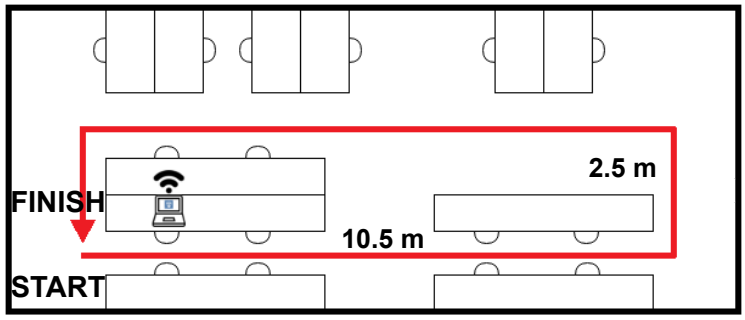
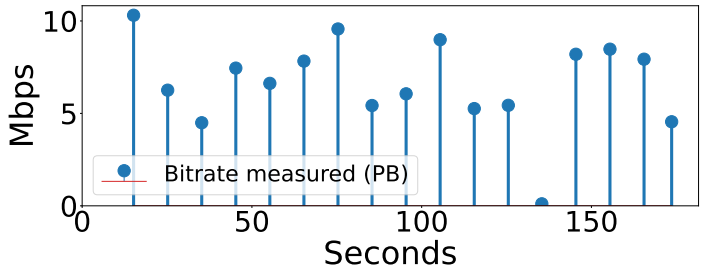
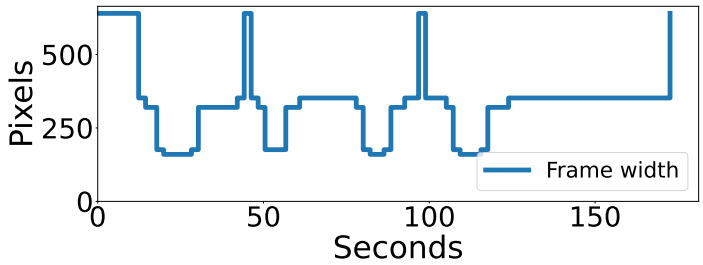


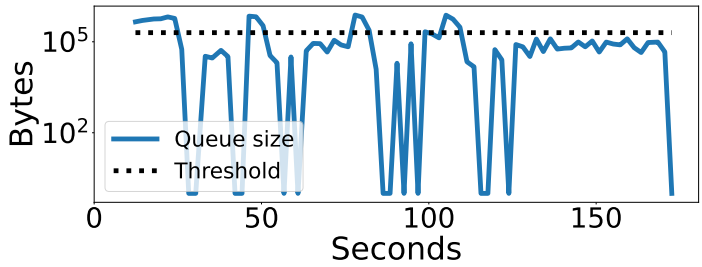
Figure 4.17. Testing in uncontrolled conditions: path inside our laboratory.



(a) Bitrate measured.



(b) Frame width.



(c) Queue sizes.

Figure 4.18. Testing in uncontrolled conditions: results of the experiment inside the laboratory.

The first plot of Fig. 4.18 shows the bitrate measured by the PB node every ten seconds. The frame widths are reported in the second plot. The outcome of the SQ node with a monitoring period of 2 seconds is reported in the third plot. As we can see, the reader decreases the video resolution when the queue size exceeds the threshold, while the video resolution is increased when the queue size is null.
We then carry out a second group of experiments, driving the Smart Video Car outside our laboratory, introducing a brick wall obstacle between the Raspberry Pi and the access point. The path followed by our car is reported in Fig. 4.19. In this case we use the same interval of two seconds

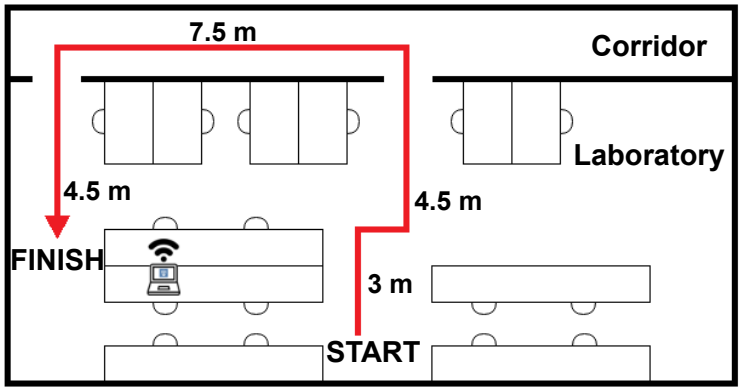


Figure 4.19. Testing in uncontrolled conditions: path in the laboratory and in the corridor.

for both of our passive nodes in order to have more information in our graphs. In the first plot of Fig. 4.20 we also report the bitrate values retrieved from *iwconfig*. The Wi-Fi NIC of our Raspberry supports multiple bit rates, so we can interpret the red line of the first plot as the nominal available bitrate. In Fig. 4.20 we can see a sudden drop of the bitrate reported by *iwconfig* before the 120th second of the experiment. These moments correspond to the exit of the car from the laboratory and the beginning of the straight path in the corridor. Even though this seems to be the section where the network conditions are the worst, our platform is still able to capture frames at the lowest resolution possible. We can see that on the 120th second the queue is empty, so it is possible to increase

the video resolution, even if the available bitrate is limited.

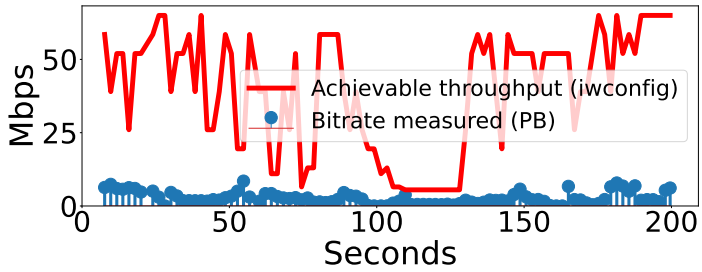
As we can notice from Fig. 4.18 and Fig. 4.20, the frames capture starts with the 640x480 resolution, which is decided after the active monitoring, that in our experiments always returns a bitrate of 30-35 Mbps. Despite having this available bitrate and using only around 5-10 Mbps to send the video frames, we observe large queues at the beginning of our experiment and the video resolution is quickly reduced. The large number of bytes not acknowledged by the remote host may be caused by the ACK mechanisms in IEEE 802.11. However, the queues in some moments of the experiments are zero, so it is possible for our system to increase the video resolution. Thanks to these experiments, we have shown that DewROS2 works properly even in uncontrolled conditions, and allows to exploit the available network. Thanks to the continuous monitoring, we can improve the quality of the video transmission when possible, and ultimately improve the results of the final application.

4.2 Secure task distribution

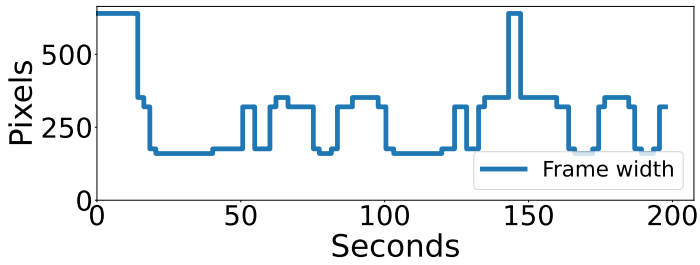
This section describes the results obtained when testing our approach for the classification of malicious edge nodes. The testbed on which our solution was tested is described in the previous chapter. It is made of one root node, two edge nodes, and five sensor nodes. We show the soundness of our approaches by testing two cases: in both cases one edge node has always a correct behaviour, while the second one does not behave correctly. In particular the malicious edge node has a constant bad behaviour in the first scenario, while it periodically switches between good and bad behaviour in the second scenario. The difference between the two edge nodes is however clear in both scenarios and the malicious one can always be excluded from our system.

4.2.1 Edge node always malicious

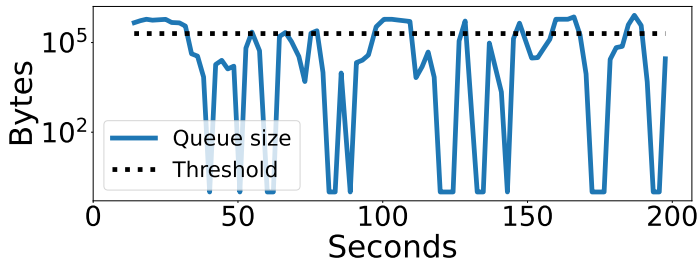
In this scenario, two edge nodes are available for supporting the application running on the sensor nodes: one edge node has a malicious behaviour that introduce a delay in the reply, lowering the throughput, while the other one always has a fair behaviour. We can notice the dif-



(a) Bitrate measured.



(b) Frame width.



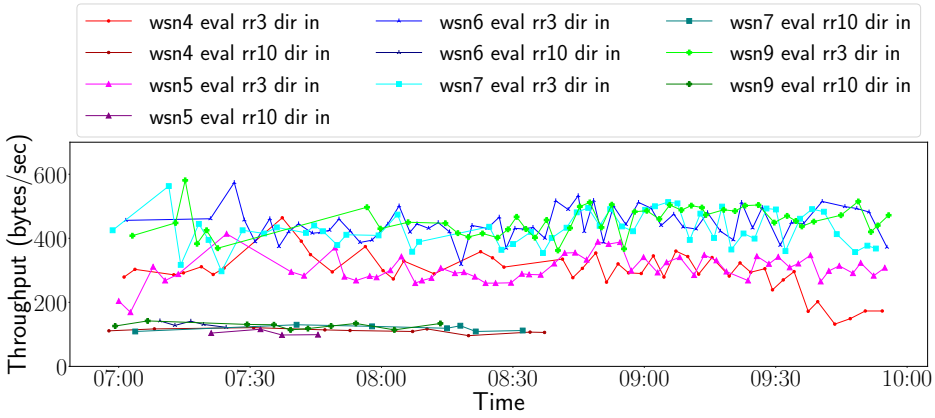
(c) Queue sizes.

Figure 4.20. Testing in uncontrolled conditions: results of the experiment in the laboratory and in the corridor.

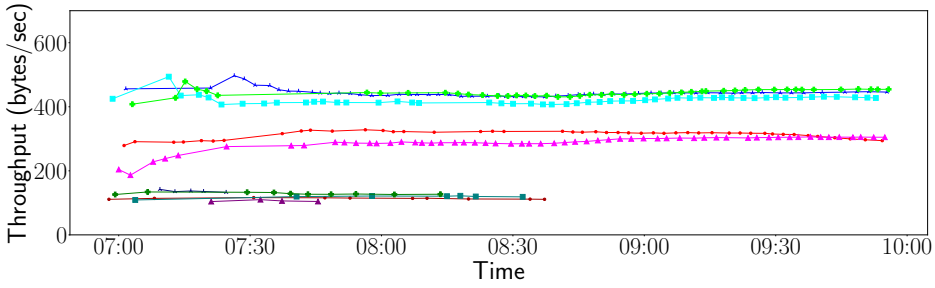
ference in the behaviour of the edge nodes by monitoring the incoming throughput for the considered application, which receives a response from the edge node. In Fig. 4.21 we report the trends of the throughput from the edge nodes to the sensor nodes and their averages. The difference between the good and the malicious edge node is evident: throughput coming from the good edge node is always higher than the the one coming form the malicious edge node. The plots already show how the malicious edge node is dropped by the sensor nodes after a certain amount of time. The reason why the bad edge node is dropped can be pinpointed from the the trends of the goodnesses of throughput (GOTs), reported in Fig 4.22 and Fig. 4.23. In Fig. 4.22c we notice that the "normal" GOT reaches 0 in some occasions even for the good edge node. The GOTs reported in Fig. 4.22 do not provide enough information for distinguishing the two different edge nodes. What really helps our system are the global GOTs, reported in Fig. 4.23. Fig. 4.23c shows that the global GOT for the good edge node never goes below the threshold of 0.5. Therefore, the good edge node is always correctly classified as good. On the other hand, the global GOT for the bad edge node often assumes values lower than 0.5. The bad nodes are therefore classified as bad when both the conditions mentioned in the previous chapter are met at the same time. The outcome of our approach can be also seen by considering the number of times the application running on the sensor nodes offloads its task to the two different edge nodes. Fig. 4.24 shows the number of tasks respectively offloaded to the two edge nodes, and it is clear that the good edge node (rr3 in the plot) gets the majority of the offloaded tasks.

4.2.2 Edge node periodically malicious

In the second scenario, one edge node never introduces delay, while the second one can assume two possible behaviours: one in which it introduces delay, and one in which it does not. This two behaviours alternate periodically. The trend of the incoming throughput is reported in Fig. 4.25. The sensor nodes 4 and 5 receive a lower throughput from edge node 3 than the one received by sensor nodes 6, 7, and 9. However we can still see how the classification works and the good edge node is never considered as malicious. The GOT assumes values included in the range [0;1] also in this scenario and there is no clear distinction between the curves of the



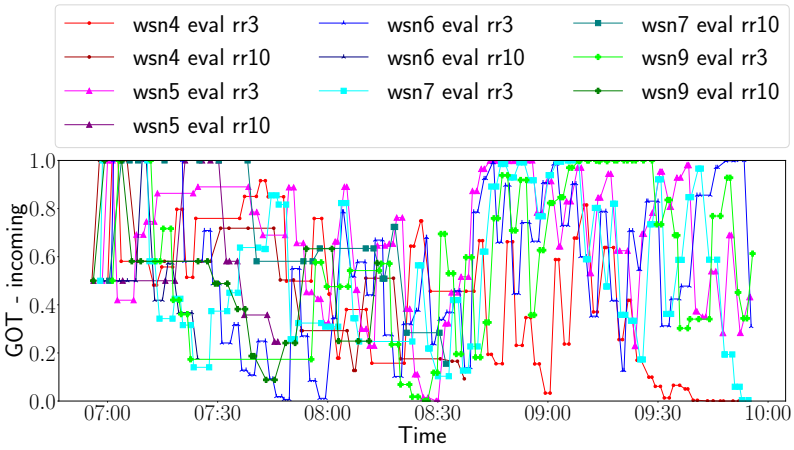
(a) Incoming throughput for routing application.



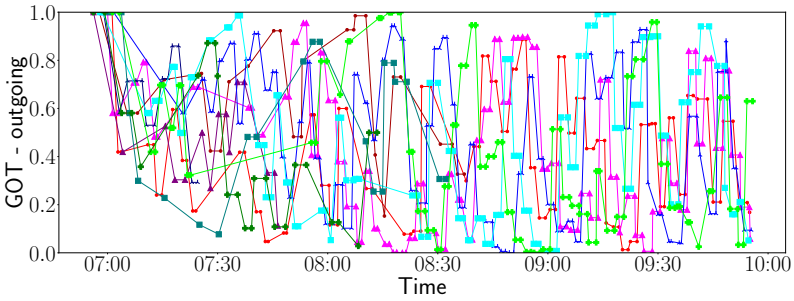
(b) Average incoming throughput for routing application.

Figure 4.21. Scenario with one edge node (rr10) always malicious, for routing application.

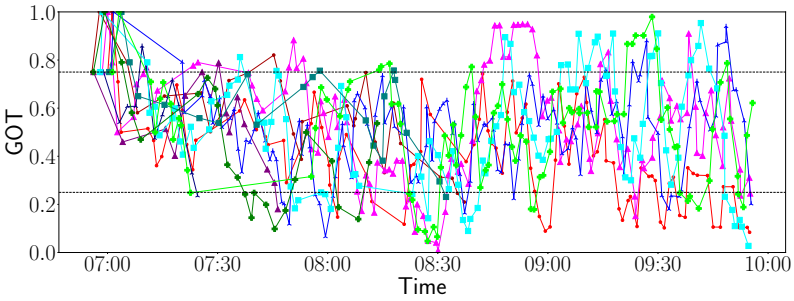
good edge node and the bad one (Fig. 4.26). In this case, the global GOT is once again the index that allows to discriminate the edge node with the bad behaviour. The difference of the values reported in Fig. 4.27a and in the average reported in Fig. 4.27a make it clear that the two edge nodes are behaving in different ways. When both the conditions on the thresholds are true, the malicious edge node is classified as bad. The trend of the number of tasks sent to the two edge nodes is very interesting in this case. The plot of Fig. 4.28 shows the number of offloaded tasks for every period in which the bad edge nodes switches behaviour. At the beginning of the experiment the distribution of tasks is more or less balanced between



(a) Goodness of incoming throughput for routing

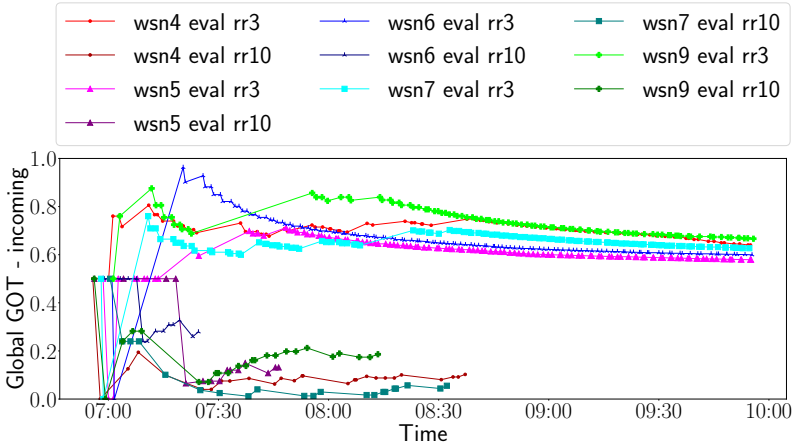


(b) Goodness of outgoing throughput for routing

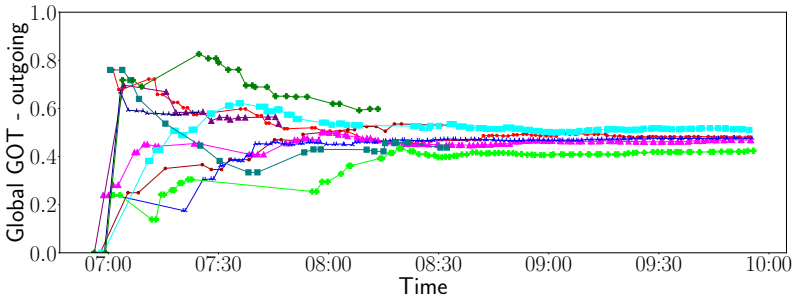


(c) Goodness of throughput for routing

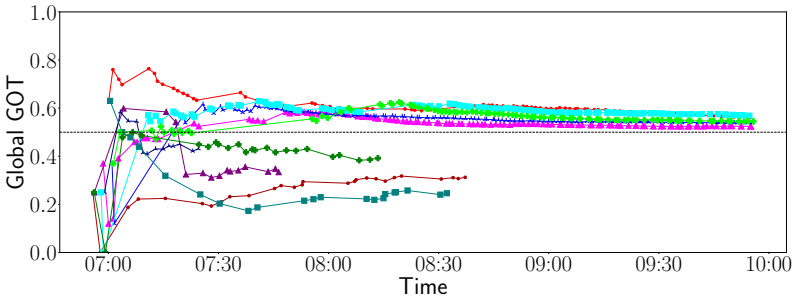
Figure 4.22. Goodnesses of throughput for routing application. Edge node rr3 is always good while rr10 is always malicious in this scenario.



(a) Global goodness of incoming throughput for routing



(b) Global goodness of outgoing throughput for routing



(c) Global goodness of throughput for routing

Figure 4.23. Global goodnesses of throughput for routing application. Edge node rr3 is always good while rr10 is always malicious in this scenario.

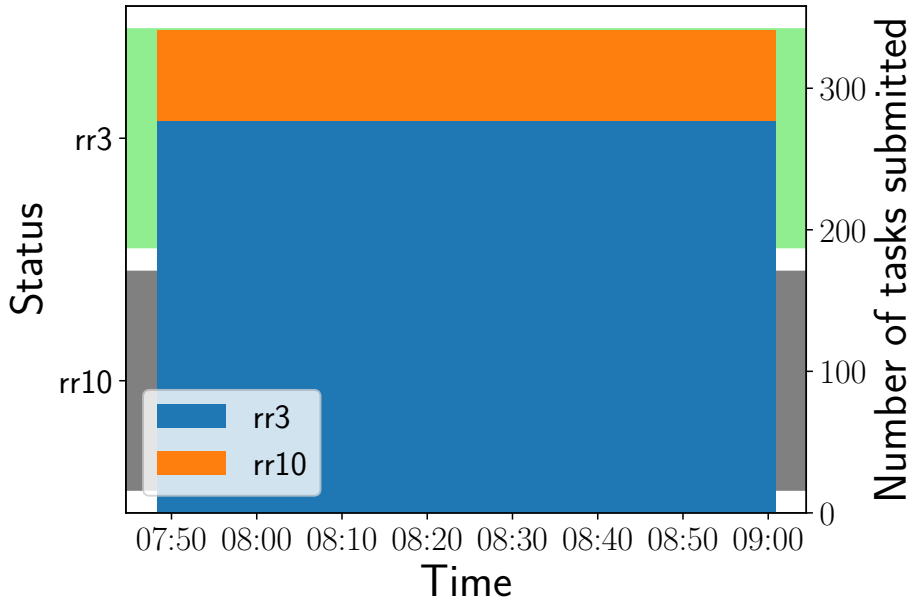
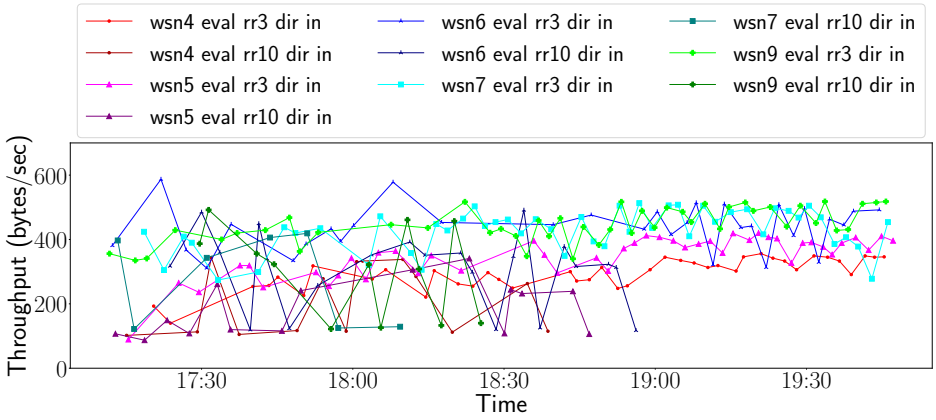
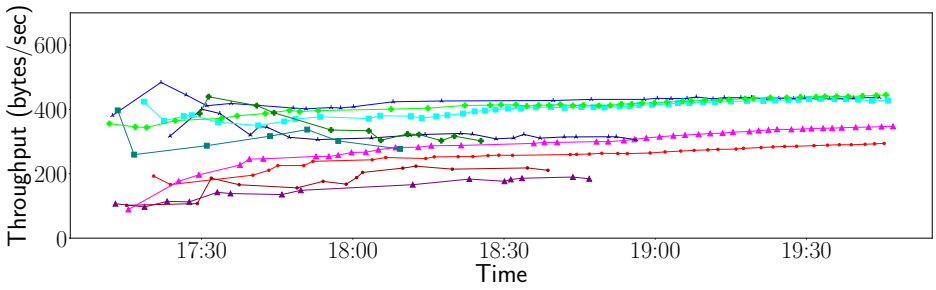


Figure 4.24. Number of tasks submitted to the edge nodes. Edge node rr3 is always good while rr10 is always malicious in this scenario.



(a) Incoming throughput for routing application.

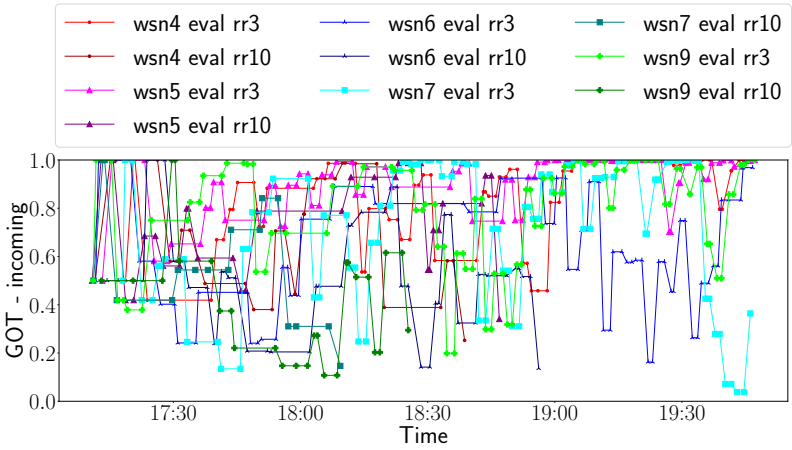


(b) Average incoming throughput for routing application.

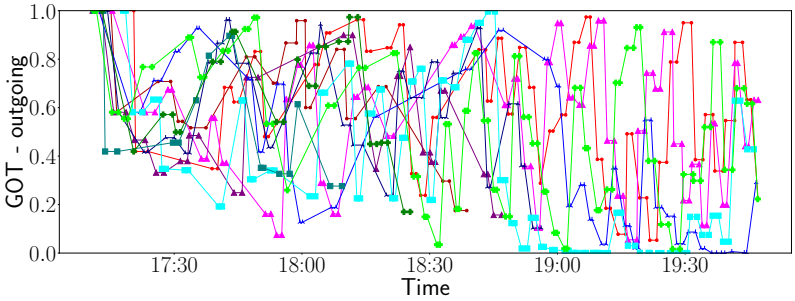
Figure 4.25. Scenario with one edge node (rr10) periodically malicious, incoming throughput for routing application.

the two edge nodes. As the experiment progresses, the number of tasks offloaded to the malicious edge node (rr10 in the figure) decreases, since the sensor nodes, one after another, classify it as bad and choose only the good one. This is the second proof of the soundness of our approach.

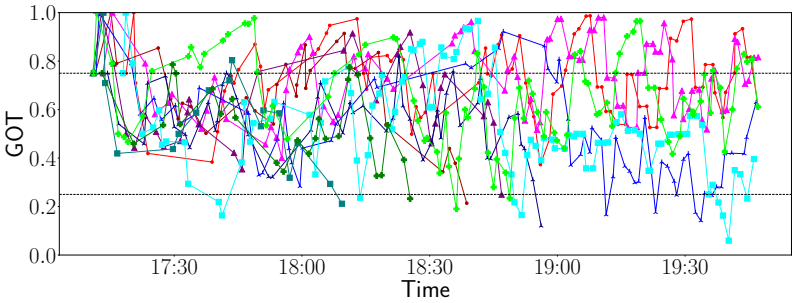
These experiments show that is possible to monitor the status of the nodes of an IoT system considering its communication performance. By removing the corrupted entities from our system, we can guarantee that the communication is only between sound components, granting the security of our IoT application.



(a) Goodness of incoming throughput for routing

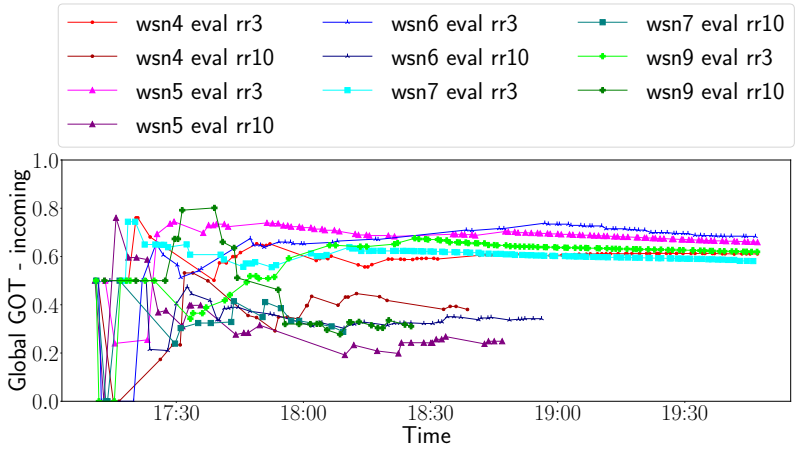


(b) Goodness of outgoing throughput for routing

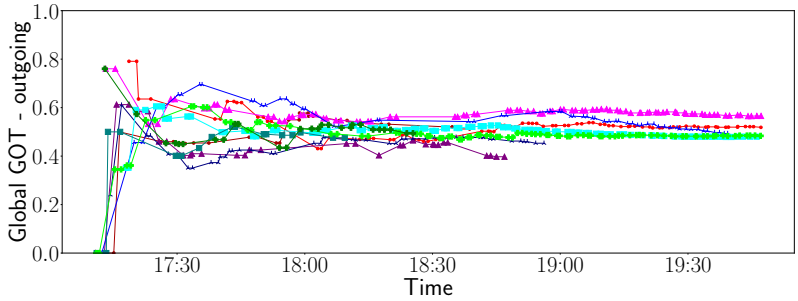


(c) Goodness of throughput for routing

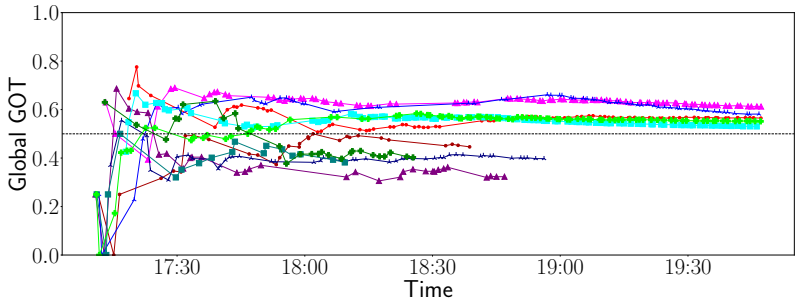
Figure 4.26. Goodnesses of throughput for routing application. Edge node rr3 is always good while rr10 is periodically malicious in this scenario.



(a) Global goodness of incoming throughput for routing



(b) Global goodness of outgoing throughput for routing



(c) Global goodness of throughput for routing

Figure 4.27. Global goodnesses of throughput for routing application. Edge node rr3 is always good while rr10 is periodically malicious in this scenario.

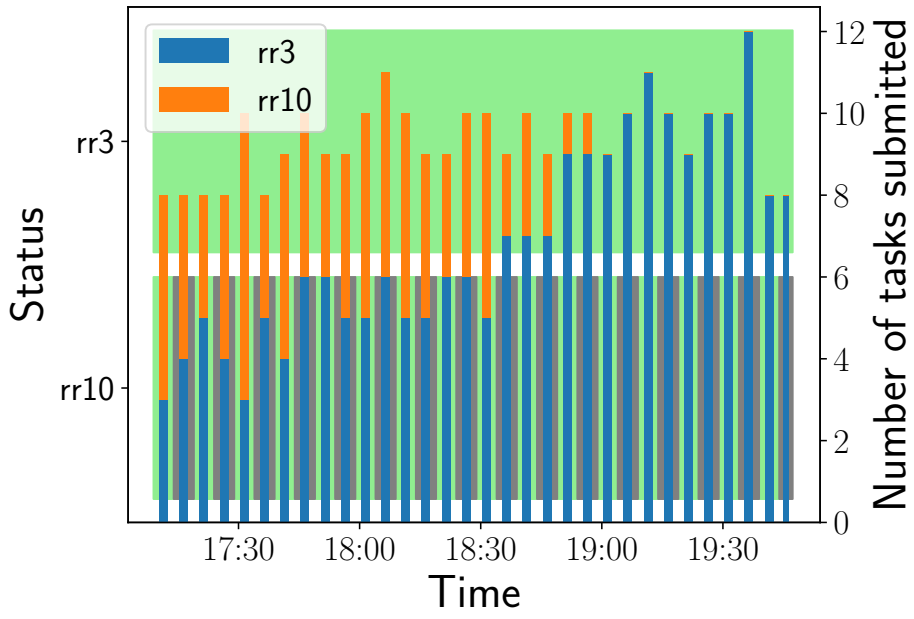


Figure 4.28. Number of tasks submitted to the edge nodes. Edge node rr3 is always good while rr10 is periodically malicious in this scenario.

Chapter 5

Conclusions

Deploying IoT systems and applications based on emerging long range connectivity is one of the interests that has recently emerged from industry. These new solutions will allow interested companies in successfully connecting distributed CPSs, with lower costs of installation. It is furthermore important to use these solutions in a safe way, without putting at risk the security of customers and the possible financial losses. Interest in investigating the aspects that can prevent the successful use of these technologies is then very high.
In this dissertation, we focused on showing that it is possible to have secure CPSs connected via an IoT network. The first outcome from our work is the empirical evaluation of the performance and feasibility of an IoT solution based on LPWAN. Our analysis provides the elements to determine which are the use cases these wireless networks can be used with security. Sigfox proved to be a technology reliable since all the messages were correctly delivered to destination. Being sure that all the data are correctly collected is an interesting aspect for companies. On the other hand, the large latency registered is surely a major setback, that prevents to use Sigfox in safety-critical scenarios in which humans interact with a CPS. The analysis of LoRaWAN and NB-IoT showed how latency is smaller when using these technologies. Knowing these performance aspects of the several connection possibilities is valuable to developers that need to deploy IoT solutions that match the reliability required by customers.
The realisation of a secure system is possible when continuous monitor-

ing is introduced. The analysis of the overall performance of a system is fundamental for users, in order to enable them to regulate the parameters of a monitoring infrastructure. Monitoring can actually be exploited for different purposes. We showed how it is possible to deploy a monitoring platform even on resource-constrained devices, that are typically involved in IoT solutions. Their impact on the resources is very limited, hence they do not represent a burden that hinders the correct functioning of the devices. One of the benefits introduced by the continuous monitoring is the capability of deciding how to use the available resources in order to have the best performance. We reported the case of a CPS employed in a search and rescue activity and showed how it is possible to almost double the utilization of the network. Further benefits are undoubtedly achievable in other contexts, adopting our solution with regard to other important resources in various use cases.

Another important benefit introduced by the monitoring of the network condition is the capacity of detecting which components of an IoT system are under attack, in order to keep users protected from malicious attackers. The proposed approach for the classification of the behaviour of an edge node proved to be successful in our experiments. All the reported plots showed how the sensor nodes were able to determine the presence of an edge node whose performance were lower than the other, and were able to automatically decide the good edge node they could trust. Our approach introduced the monitoring of the network performance to exclude malicious components in our system. The outcome of our research not only proved the viability of classification of corrupted entities based on network performance parameters, but it can also foster research in how to improve secure task offloading. This approach is in fact not well investigated, while it is important to deal with possible network outages in order to be always able to reach the destinations that we need to carry out the tasks of our application. This way, it is possible to deploy secure and reliable CPS solutions. The proposed approach for monitoring could be further improved by applying the principles of our proposed monitoring platform.

The empirical study of their feasibility will give an additional incentive to the diffusion of wireless networks for IoT. Their success will be the base not only for many business opportunities for companies, but also for improving the life and the everyday activities of customers. This success can

not exclude the need of security: for example, it is fundamental to protect our assets and our safety when IoT solutions enter our homes. Our work represents another step in this direction, excluding the intruders that threaten our security.

Bibliography

[1] The Things Industries. LoRaWAN Architecture. Available at https://www.thethingsnetwork.org/docs/lorawan/architecture/. Accessed on: Oct. 27, 2022.

[2] The Things Industries. Message types. Available at https://www.thethingsnetwork.org/docs/lorawan/message-types/. Accessed on: Oct. 27, 2022.

[3] Sigfox. Get to know Sigfox. Available at https://build.sigfox.com/sigfox. Accessed on: Oct. 27, 2022.

[4] Sigfox. Sigfox Cloud Integration. Available at https://build.sigfox.com/backend-callbacks-and-api. Accessed on: Sept. 27, 2022.

[5] The Things Industries. The Things Network Architecture. Available at https://www.thethingsnetwork.org/article/the-things-network-architecture-1. Accessed on: Sept. 27, 2022.

[6] Lorenzo Marconi, Claudio Melchiorri, Michael Beetz, Dejan Pangercic, Roland Siegwart, Stefan Leutenegger, Raffaella Carloni, Stefano Stramigioli, Herman Bruyninckx, Patrick Doherty, Alexander Kleiner, Vincenzo Lippiello, Alberto Finzi, Bruno Siciliano, Andrea Sala, and Nicola Tomatis. The SHERPA project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments. In *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–4. IEEE, 2012.

[7] Raspebrry Italia. Sunfounder Raspberry Pi Smart Robot Car Kit. Available at https://www.raspberryitalia.it/prodotto/sunfounder-raspberry-pi-smart-robot-car-kit/. Accessed on: Oct. 28, 2022.

[8] Philippe Reininger. 3GPP Standards for the Internet-of-Things. In *IoT Business & Technologies Congress, Singapore, November*, volume 30, 2016.

[9] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT express*, 5(1):1–7, 2019.

[10] Kishor Krishnan Nair, Adnan M Abu-Mahfouz, and Samuel Lefophane. Analysis of the narrow band internet of things (NB-IoT) technology. In *2019 conference on information communications technology and society (ICTAS)*, pages 1–6. IEEE, 2019.

[11] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications. *IEEE internet of things journal*, 4(5):1125–1142, 2017.

[12] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[13] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[14] Michael Chui, Mark Collins, and Mark Patel. The Internet of Things: Catching up to an accelerating opportunity. *McKinsey & Company, New York*, 2021.

[15] Bluetooth SIG. Bluetooth Core Specification Version 4.0. Available at https://www.bluetooth.com/specifications/bluetooth-core-specification. Accessed on: Sept. 1, 2022.

[16] Alexandre Adomnicai, Jacques JA Fournier, and Laurent Masson. Hardware security threats against Bluetooth mesh networks. In *2018 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2018.

[17] Kevin Townsend, Robert Davidson, and Carles Cufi. *Getting Started with Bluetooth Low Energy*. O'Reilly, 2014.

[18] Sode Pallavi and V Anantha Narayanan. An overview of practical attacks on BLE based IoT devices and their security. In *2019 5th International Conference on Advanced Computing Communication Systems (ICACCS)*, pages 694–698. IEEE, 2019.

[19] YW Prakash, Vishakha Biradar, Shenil Vincent, Minto Martin, and Anita Jadhav. Smart Bluetooth Low Energy security system. In *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 2141–2146. IEEE, 2017.

[20] Anthony Rose, Jason Bindewald, Benjamin Ramsey, Mason Rice, and Barry Mullins. Securing Bluetooth Low Energy locks from unauthorized access and surveillance. In Mason Rice and Sujeet Shenoi, editors, *Critical Infrastructure Protection XI*, pages 319–338, Cham, 2017. Springer International Publishing.

[21] Gaetano Patti, Luca Leonardi, and Lucia Lo Bello. A Bluetooth Low Energy real-time protocol for Industrial Wireless mesh Networks. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 4627–4632. IEEE, 2016.

[22] Inc. Bluetooth SIG. Bluetooth Technology Overview. Available at https://www.bluetooth.com/learn-about-bluetooth/tech-overview/. Accessed on: Dec. 12, 2022.

[23] Giwon Kwon, Jeehyeong Kim, Jaewon Noh, and Sunghyun Cho. Bluetooth low energy security vulnerability and improvement method. In *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–4. IEEE, 2016.

[24] Yanzhen Qu and Philip Chan. Assessing vulnerabilities in Bluetooth low energy (BLE) wireless network based IoT systems. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, pages 42–48. IEEE, 2016.

[25] Yasmin M Amin and Amr T Abdel-Hamid. Classification and analysis of IEEE 802.15. 4 PHY layer attacks. In *2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT)*, pages 1–8. IEEE, 2016.

[26] Yang Xiao, Hsiao-Hwa Chen, Bo Sun, Ruhai Wang, and Sakshi Sethi. MAC security and security overhead analysis in the IEEE 802.15. 4 wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2006:1–12, 2006.

[27] Roberta Daidone, Gianluca Dini, and Marco Tiloca. On experimentally evaluating the impact of security on IEEE 802.15.4 networks. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–6. IEEE, 2011.

[28] Adam Reziouk, Enzo Laurent, and Jonathan-Christofer Demay. Practical security overview of IEEE 802.15.4. In *2016 International Conference on Engineering MIS (ICEMIS)*, pages 1–9. IEEE, 2016.

[29] Naveen Sastry and David Wagner. Security considerations for IEEE 802.15. 4 networks. In *Proceedings of the 3rd ACM workshop on Wireless security*, WiSe '04, page 32–42, New York, NY, USA, 2004. Association for Computing Machinery.

[30] Roberta Daidone. Experimental evaluations of security impact on IEEE 802.15.4 networks. In *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–2. IEEE, 2011.

[31] IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pages 1–314, 2011.

[32] Syed Muhammad Sajjad and Muhammad Yousaf. Security analysis of IEEE 802.15. 4 MAC in the context of Internet of Things (IoT). In *2014 Conference on Information Assurance and Cyber Security (CIACS)*, pages 9–14. IEEE, 2014.

[33] Noureddine Boucif, Frederik Golchert, Alexander Siemer, Patrick Felke, and Frederik Gosewehr. Crushing the Wave–new Z-Wave vulnerabilities exposed. *arXiv preprint arXiv:2001.08497*, 2020.

[34] Jonathan D. Fuller, Benjamin W. Ramsey, Mason J. Rice, and John M. Pecarina. Misuse-based detection of Z-Wave network attacks. *Computers & Security*, 64(C):44–58, 2017.

[35] Muneer Bani Yassein, Wail Mardini, and Ashwaq Khalil. Smart homes automation using Z-wave protocol. In *2016 International Conference on Engineering & MIS (ICEMIS)*, pages 1–6. IEEE, 2016.

[36] Loïc ROUCH, Jérôme François, Frédéric Beck, and Abdelkader Lahmadi. A Universal Controller to Take Over a Z-Wave Network. In *Black Hat Europe 2017*, pages 1–9, London, United Kingdom, 2017.

[37] Christopher W. Badenhop, Scott R. Graham, Benjamin W. Ramsey, Barry E. Mullins, and Logan O. Mailloux. The Z-Wave routing protocol and its security implications. *Computers & Security*, 68:112–129, 2017.

[38] Muneer Bani Yassein, Wail Mardini, and Taha Almasri. Evaluation of security regarding Z-Wave wireless protocol. In *Proceedings of the Fourth International Conference on Engineering & MIS 2018*, pages 1–8, 2018.

[39] Katherine Hoskins. Security Vulnerabilities in Z-Wave Home Automation Protocol. *Tufts University Department of Computer Science*, 14, 2016.

[40] Daniel Celebucki, Maj Alan Lin, and Scott Graham. A security evaluation of popular Internet of Things protocols for manufacturers. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6. IEEE, 2018.

[41] Ulya Sabeel and Nidhi Chandra. A smart and contemporary home security system using 802.15. 4 standard. In *2013 5th International Conference and Computational Intelligence and Communication Networks*, pages 374–379. IEEE, 2013.

[42] ZigBee Standards Organization. ZigBee Smart Energy Standard. Available at https://zigbeealliance.org/wp-content/uploads/2019/11/docs-07-5356-19-0zse-zigbee-smart-energy-profile-specification.pdf. Accessed on: Sept. 1, 2022.

[43] Olayemi Olawumi, Keijo Haataja, Mikko Asikainen, Niko Vidgren, and Pekka Toivanen. Three practical attacks against ZigBee security: Attack scenario definitions, practical experiments, countermeasures, and lessons learned. In *2014 14th International Conference on Hybrid Intelligent Systems*, pages 199–206. IEEE, 2014.

[44] Niko Vidgren, Keijo Haataja, José Luis Patiño-Andres, Juan José Ramírez-Sanchis, and Pekka Toivanen. Security Threats in ZigBee-Enabled Systems: Vulnerability Evaluation, Practical Experiments, Countermeasures, and Lessons Learned. In *2013 46th Hawaii International Conference on System Sciences*, pages 5132–5138, 2013.

[45] Xueqi Fan, Fransisca Susan, William Long, and Shangyan Li. Security analysis of ZigBee. *MWR InfoSecurity*, pages 1–18, 2017.

[46] Meng Qianqian and Bao Kejin. Security analysis for wireless networks based on ZigBee. In *2009 International Forum on Information Technology and Applications*, volume 1, pages 158–160. IEEE, 2009.

[47] Gianluca Dini and Marco Tiloca. Considerations on security in ZigBee networks. In *2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pages 58–65. IEEE, 2010.

[48] Hongwei Li, Zhongning Jia, and Xiaofeng Xue. Application and analysis of ZigBee security services specification. In *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, volume 2, pages 494–497. IEEE, 2010.

[49] Connectivity Standards Alliance. ZigBee - Complete IoT Solution. Available at https://csa-iot.org/all-solutions/zigbee/. Accessed on: Nov. 14, 2022.

[50] Usman Raza, Parag Kulkarni, and Mahesh Sooriyabandara. Low Power Wide Area Networks: An Overview. *IEEE Communications Surveys & Tutorials*, 19(2):855–873, 2017.

[51] Sigfox Foundation. sigfox security white paper.

[52] Sigfox Foundation. Sigfox Cloud Integration. Available at https://build.sigfox.com/backend-callbacks-and-api. Accessed on: Sept. 1, 2022.

[53] Alexandru Lavric, Adrian I Petrariu, and Valentin Popa. Long Range SigFox Communication Protocol Scalability Analysis Under Large-Scale, High-Density Conditions. *IEEE Access*, 7:35816–35825, 2019.

[54] George Margelis, Robert Piechocki, Dritan Kaleshi, and Paul Thomas. Low throughput networks for the IoT: Lessons learned from industrial implementations. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 181–186. IEEE, 2015.

[55] Smilty Chacko and Deepu Job. Security mechanisms and Vulnerabilities in LPWAN. *IOP Conference Series: Materials Science and Engineering*, 396(1):012027, 2018.

[56] Radek Fujdiak, Petr Blazek, Konstantin Mikhaylov, Lukas Malina, Petr Mlynek, Jiri Misurec, and Vojtech Blazek. On track of Sigfox confidentiality with end-to-end encryption. In *Proceedings of the 13th international conference on availability, reliability and security*, pages 1–6, 2018.

[57] Pietro Di Gennaro, Domenico Lofu, Vitanio Daniele, Pietro Tedeschi, and Pietro Boccadoro. WaterS: A Sigfox-compliant prototype for water monitoring. *Internet Technology Letters*, 2(1):6, 2018.

[58] Guillaume Ferré and Eric Pierre Simon. An introduction to Sigfox and LoRa PHY and MAC layers. working paper or preprint, 2018.

[59] Brecht Reynders, Wannes Meert, and Sofie Pollin. Range and coexistence analysis of long range unlicensed communication. In *2016 23rd International Conference on Telecommunications (ICT)*, pages 1–6. IEEE, 2016.

[60] Konstantin Mikhaylov, Juha Petaejaejaervi, and Tuomo Haenninen. Analysis of capacity and scalability of the LoRa low power wide area network technology. In *European Wireless 2016; 22th European Wireless Conference*, pages 1–6. VDE, 2016.

[61] Keith E Nolan, Wael Guibene, and Mark Y Kelly. An evaluation of low power wide area network technologies for the Internet of Things. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 439–444. IEEE, 2016.

[62] Ismail Butun, Nuno Pereira, and Mikael Gidlund. Analysis of LoRaWAN v1. 1 security. In *Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects*, SMARTOBJECTS '18, New York, NY, USA, 2018. Association for Computing Machinery.

[63] Emekcan Aras, Gowri Sankar Ramachandran, Piers Lawrence, and Danny Hughes. Exploring the security vulnerabilities of LoRa. In *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*, pages 1–6. IEEE, 2017.

[64] Jaehyu Kim and JooSeok Song. A simple and efficient replay attack prevention scheme for LoRaWAN. In *Proceedings of the 2017 the 7th International Conference on Communication and Network Security*, ICCNS 2017, page 32–36, New York, NY, USA, 2017. Association for Computing Machinery.

[65] Florian Laurentiu Coman, Krzysztof Mateusz Malarski, Martin Nordal Petersen, and Sarah Ruepp. Security Issues in Internet of Things: Vulnerability Analysis of LoRaWAN, Sigfox and NB-IoT. In *2019 Global IoT Summit (GIoTS)*, pages 1–6. IEEE, 2019.

[66] Eszter Kail, Anna Banati, Erdödi Lászlo, and Miklós Kozlovszky. Security survey of dedicated IoT networks in the unlicensed ISM bands. In *2018 IEEE 12th international symposium on applied computational intelligence and informatics (SACI)*, pages 000449–000454. IEEE, 2018.

[67] Mohamed Eldefrawy, Ismail Butun, Nuno Pereira, and Mikael Gidlund. Formal security analysis of LoRaWAN. *Computer Networks*, 148, 2018.

[68] Bogdan Oniga, Vasile Dadarlat, Eli De Poorter, and Adrian Munteanu. Analysis, design and implementation of secure LoRaWAN sensor networks. In *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 421–428. IEEE, 2017.

[69] LoRa Alliance. LoRa Alliance. Available at https://lora-alliance.org/. Accessed on: Nov. 14, 2022.

[70] Stefano Tomasin, Simone Zulian, and Lorenzo Vangelista. Security analysis of LoRaWAN join procedure for Internet of Things networks. In *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6. IEEE, 2017.

[71] Kishor Krishnan Nair, Adnan M Abu-Mahfouz, and Samuel Lefophane. Analysis of the Narrow Band Internet of Things (NB-IoT) Technology. In *2019 Conference on Information Communications Technology and Society (ICTAS)*, pages 1–6. IEEE, 2019.

[72] Jun Xu, Junmei Yao, Lu Wang, Zhong Ming, Kaishun Wu, and Lei Chen. Narrowband Internet of Things: Evolutions, Technologies, and Open Issues. *IEEE Internet of Things Journal*, 5(3):1449–1462, 2018.

[73] Vinod Kumar, Rakesh Kumar Jha, and Sanjeev Jain. NB-IoT Security: A Survey. *Wireless Personal Communications*, 113(4):2661–2708, 2020.

[74] Almudena Díaz Zayas, Francisco Javier Rivas Tocado, and Pilar Rodríguez. Evolution and testing of NB-IoT solutions. *Applied Sciences*, 10(21), 2020.

[75] Oriol Solà Campillo. *Security issues in Internet of Things.* PhD thesis, UPC, Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, 2017.

[76] QUALCOMM®. Paving the path to Narrowband 5G with LTE Internet of Things (IoT). Available at https://www.qualcomm.cn/media/documents/files/paving-the-path-to-narrowband-5g-with-lte-iot.pdf. Accessed on: Sept. 9, 2022.

[77] Yuesong Lin, Fuqiang Jiang, Zhu Wang, and Zhuping Wang. Research on PUF-Based Security Enhancement of Narrow-Band Internet of Things. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pages 702–709. IEEE, 2018.

[78] Ijaz Ahmad, Tanesh Kumar, Madhusanka Liyanage, Jude Okwuibe, Mika Ylianttila, and Andrei Gurtov. 5G security: Analysis of threats and solutions. In *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 193–199. IEEE, 2017.

[79] Ijaz Ahmad, Shahriar Shahabuddin, Tanesh Kumar, Jude Okwuibe, Andrei Gurtov, and Mika Ylianttila. Security for 5G and beyond. *IEEE Communications Surveys & Tutorials*, 21(4):3682–3722, 2019.

[80] Godfrey Anuga Akpakwu, Bruno J Silva, Gerhard P Hancke, and Adnan M Abu-Mahfouz. A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges. *IEEE Access*, 6:3619–3647, 2018.

[81] Maria Rita Palattella, Mischa Dohler, Alfredo Grieco, Gianluca Rizzo, Johan Torsner, Thomas Engel, and Latif Ladid. Internet of Things in the 5G era: Enablers, architecture, and business models. *IEEE Journal on Selected Areas in Communications*, 34(3):510–527, 2016.

[82] Jesus Sanchez-Gomez, Dan García Carrillo, Ramon Sanchez-Iborra, José L Hernández-Ramos, Jorge Granjal, Rafael Marin-Perez, and Miguel A Zamora-Izquierdo. Integrating LPWAN technologies in the 5G ecosystem: A survey on security challenges and solutions. *IEEE Access*, 8:216437–216460, 2020.

[83] 3GPP. *Security architecture and procedures for 5G system*, 2018. v15.1.0.

[84] Xueying Yang, Evgenios Karampatzakis, Christian Doerr, and Fernando Kuipers. Security Vulnerabilities in LoRaWAN. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 129–140. IEEE, 2018.

[85] Roger Piqueras Jover. LTE security, protocol exploits and location tracking experimentation with low-cost software radio. *CoRR*, abs/1607.05171, 2016.

[86] Seth Sevier and Ali Tekeoglu. Analyzing the security of Bluetooth Low Energy. In *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, pages 1–5. IEEE, 2019.

[87] Xianghui Cao, Devu Manikantan Shila, Yu Cheng, Zequ Yang, Yang Zhou, and Jiming Chen. Ghost-in-ZigBee: Energy depletion attack on ZigBee-based wireless networks. *IEEE Internet of Things Journal*, 3(5):816–829, 2016.

[88] Niko Vidgren, Keijo Haataja, Jose Luis Patino-Andres, Juan Jose Ramirez-Sanchis, and Pekka Toivanen. Security threats in ZigBee-enabled systems: Vulnerability evaluation, practical experiments, countermeasures, and lessons learned. In *2013 46th Hawaii International Conference on System Sciences*, pages 5132–5138. IEEE, 2013.

[89] Mike Ryan. Bluetooth: With Low Energy Comes Low Security. In *7th USENIX Workshop on Offensive Technologies (WOOT 13)*, Washington, D.C., 2013. USENIX Association.

[90] Behrang Fouladi and Sahand Ghanoun. Security evaluation of the Z-Wave wireless protocol. *Black hat USA*, 24:1–2, 2013.

[91] Vladislav Skorpil, Vaclav Oujezsky, and Ludek Palenik. Internet of Things Security Overview and Practical Demonstration. In *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 1–7, 2018.

[92] Nuno Torres, Pedro Pinto, and Sérgio Ivan Lopes. Security Vulnerabilities in LPWANs—An Attack Vector Analysis for the IoT Ecosystem. *Applied Sciences*, 11(7):3176, 2021.

[93] JungWoon Lee, DongYeop Hwang, JiHong Park, and Ki-Hyung Kim. Risk analysis and countermeasure for bit-flipping attack in LoRaWAN. In *2017 International Conference on Information Networking (ICOIN)*, pages 549–551. IEEE, 2017.

[94] John Thomas, Season Cherian, Saranya Chandran, and Vipin Pavithran. Man in the middle attack mitigation in LoRaWAN. In *2020 International Conference on Inventive Computation Technologies (ICICT)*, pages 353–358. IEEE, 2020.

[95] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[96] Jonathan D Fuller and Benjamin W Ramsey. Rogue Z-Wave controllers: A persistent attack channel. In *2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops)*, pages 734–741. IEEE, 2015.

[97] Emekcan Aras, Nicolas Small, Gowri Sankar Ramachandran, Stéphane Delbruel, Wouter Joosen, and Danny Hughes. Selective Jamming of LoRaWAN Using Commodity Hardware. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous 2017, page 363–372, New York, NY, USA, 2017. Association for Computing Machinery.

[98] Aveek K Das, Parth H Pathak, Chen-Nee Chuah, and Prasant Mohapatra. Uncovering privacy leakage in BLE network traffic of wearable fitness trackers. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, HotMobile '16, page 99–104, New York, NY, USA, 2016. Association for Computing Machinery.

[99] Syed Rafiul Hussain, Mitziu Echeverria, Omar Chowdhury, Ninghui Li, and Elisa Bertino. Privacy attacks to the 4G and 5G cellular paging protocols using side channel information. *Network and Distributed Systems Security (NDSS) Symposium2019*, 2019.

[100] Scapy. Scapy - Packet crafting for Python2 and Python3. Available at https://scapy.net/. Accessed on: Nov. 14, 2022.

[101] S Sandhya and KA Sumithra Devi. Analysis of Bluetooth threats and v4.0 security features. In *2012 International Conference on Computing, Communication and Applications*, pages 1–4. IEEE, 2012.

[102] Ishaq Unwala, Zafar Taqvi, and Jiang Lu. IoT security: Z-Wave and Thread. In *2018 IEEE Green Technologies Conference (GreenTech)*, pages 176–182. IEEE, 2018.

[103] Andréa Voisin-Grall, Obabiolorunkosi Olaoluwapo Malaolu, Yingbo Zhu, Tanveer Ahmed, Shahriar Abdullah Al-Ahmed, and Muhammad Zeeshan Shakir. Remote Condition Monitoring: a Prototype Based on Pycom Development Board FiPy and Pysense. In *2019 UK/China Emerging Technologies (UCET)*, pages 1–6. IEEE, 2019.

[104] Kriddikorn Kaewwongsri and Kittasil Silanon. Design and Implement of a Weather Monitoring Station using CoAP on NB-IoT Network. In *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 230–233. IEEE, 2020.

[105] Matthias Herlich and Ferdinand von Tüllenburg. Introduction to narrowband communication. In *Proceedings of the Wireless Congress: Systems and Applications*, 2018.

[106] Stephen Ugwuanyi, Jidapa Hansawangkit, and James Irvine. NB-IoT testbed for industrial Internet of Things. In *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6. IEEE, 2020.

[107] Muhammad Izzam Muzammir, Husna Zainol Abidin, Syahrul Afzal Che Abdullah, and Fadhlan Hafizhelmi Kamaru Zaman. Performance analysis of LoRaWAN for indoor application. In *2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, pages 156–159. IEEE, 2019.

[108] Luiz Oliveira, Joel JPC Rodrigues, Sergei A Kozlov, Ricardo AL Rabêlo, and Vasco Furtado. Performance assessment of long-range and Sigfox protocols with mobility support. *International Journal of Communication Systems*, 32(13):e3956, 2019.

[109] Xujia Zhou and Eduardo Flora. Comparison of performance and power consumption between GPS and Sigfox positioning using Pycom modules, 2018.

[110] Manuel Pérez, Fabian Eduardo Sierra-Sánchez, Fabián Chaparro, Diego Méndez Chaves, Carlos-Ivan Paez-Rueda, Gabriel Perilla Galindo, and Arturo Fajardo. Coverage and Energy-Efficiency Experimental Test Performance for a Comparative Evaluation of Unlicensed LPWAN: LoRaWAN and SigFox. *IEEE Access*, 10:97183–97196, 2022.

[111] Shie-Yuan Wang, Jui-En Chang, Hsin Fan, and Yi-Hsiu Sun. Performance comparisons of NB-IoT, LTE Cat-M1, Sigfox, and LoRa moving at high speeds in the air. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2020.

[112] T Perković, P Šolić, H Zargariasl, D Čoko, and Joel JPC Rodrigues. Smart parking sensors: state of the art and performance evaluation. *Journal of Cleaner Production*, 262:121181, 2020.

[113] Valiallah Monajjemi, Jens Wawerla, and Richard Vaughan. Drums: A middleware-aware distributed robot monitoring system. In *Proceedings of the 2014 Canadian Conference on Computer and Robot Vision*, CRV '14, page 211–218, USA, 2014. IEEE Computer Society.

[114] Sean Rivera, Antonio Ken Iannillo, Sofiane Lagraa, Clément Joly, and Radu State. ROS-FM: Fast Monitoring for the Robotic Operating System(ROS). In *2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 187–196. IEEE, 2020.

[115] Alexander Kleiner, Gerald Steinbauer, and Franz Wotawa. Towards automated online diagnosis of robot navigation software. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 159–170. Springer, 2008.

[116] Gerald Steinbauer, Martin Mörth, and Franz Wotawa. Real-time diagnosis and repair of faults of robot control software. In *Robot Soccer World Cup*, pages 13–23. Springer, 2005.

[117] Vandi Verma, Geoff Gordon, Reid Simmons, and Sebastian Thrun. Real-time fault diagnosis [robot fault diagnosis]. *IEEE Robotics & Automation Magazine*, 11(2):56–66, 2004.

[118] Sathish Vallachira, Michal Orkisz, Mikael Norrlöf, and Sachit Butail. Data-driven gearbox failure detection in industrial robots. *IEEE Transactions on Industrial Informatics*, 16(1):193–201, 2019.

[119] Arda Inceoglu, Gökhan Ince, Yusuf Yaslan, and Sanem Sariel. Failure detection using proprioceptive, auditory and visual modalities. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2491–2496. IEEE, 2018.

[120] Haodong Lu, Miao Du, Kai Qian, Xiaoming He, and Kun Wang. GAN-based data augmentation strategy for sensor anomaly detection in industrial robots. *IEEE Sensors Journal*, 2021.

[121] Mahbuba Afrin, Jiong Jin, Ashfaqur Rahman, Yu-Chu Tian, and Ambarish Kulkarni. Multi-objective resource allocation for Edge Cloud based robotic workflow in smart factory. *Future Generation Computer Systems*, 97:119–130, 2019.

[122] Junhui Zhao, Qiuping Li, Yi Gong, and Ke Zhang. Computation Offloading and Resource Allocation For Cloud Assisted Mobile Edge Computing in Vehicular Networks. *IEEE Transactions on Vehicular Technology*, 68(8):7944–7956, 2019.

[123] Hamed Shah-Mansouri and Vincent W. S. Wong. Hierarchical Fog-Cloud Computing for IoT Systems: A Computation Offloading Game. *IEEE Internet of Things Journal*, 5(4):3246–3257, 2018.

[124] Hoa Tran-Dang and Dong-Seong Kim. FRATO: fog resource based adaptive task offloading for delay-minimizing IoT service provisioning. *IEEE Transactions on Parallel and Distributed Systems*, 32(10):2491–2508, 2021.

[125] Hayat Bashir, Seonah Lee, and Kyong Hoon Kim. Resource allocation through logistic regression and multicriteria decision making method in IoT fog computing. *Transactions on Emerging Telecommunications Technologies*, 33(2):e3824, 2022.

[126] Xianling Meng, Wei Wang, and Zhaoyang Zhang. Delay-constrained hybrid computation offloading with Cloud and Fog computing. *IEEE Access*, 5:21355–21367, 2017.

[127] Xiaolong Xu, Qingxiang Liu, Yun Luo, Kai Peng, Xuyun Zhang, Shunmei Meng, and Lianyong Qi. A computation offloading method over big data for IoT-enabled cloud-edge computing. *Future Generation Computer Systems*, 95:522–533, 2019.

[128] Jianbo Du, Liqiang Zhao, Jie Feng, and Xiaoli Chu. Computation Offloading and Resource Allocation in Mixed Fog/Cloud Computing Systems With Min-Max Fairness Guarantee. *IEEE Transactions on Communications*, 66(4):1594–1608, 2018.

[129] Victor Kathan Sarker, J Peña Queralta, Tuan Nguyen Gia, Hannu Tenhunen, and Tomi Westerlund. Offloading SLAM for Indoor Mobile Robots with Edge-Fog-Cloud Computing. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pages 1–6. IEEE, 2019.

[130] Ali Shakarami, Ali Shahidinejad, and Mostafa Ghobaei-Arani. A review on the computation offloading approaches in mobile edge computing: A game-theoretic perspective. *Software: Practice and Experience*, 50(9):1719–1759, 2020.

[131] Farzad Samie, Vasileios Tsoutsouras, Lars Bauer, Sotirios Xydis, Dimitrios Soudris, and Jörg Henkel. Computation offloading and resource allocation for low-power IoT edge devices. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 7–12. IEEE, 2016.

[132] Lingjun Pu, Xu Chen, Jingdong Xu, and Xiaoming Fu. D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration. *IEEE Journal on Selected Areas in Communications*, 34(12):3887–3901, 2016.

[133] Alessio Sacco, Flavio Esposito, and Guido Marchetto. Resource Inference for Sustainable and Responsive Task Offloading in Challenged Edge Networks. *IEEE Transactions on Green Communications and Networking*, 5(3):1114–1127, 2021.

[134] Phillip Taylor, Lina Barakat, Simon Miles, and Nathan Griffiths. Reputation assessment: a review and unifying abstraction. *The Knowledge Engineering Review*, 33, 2018.

[135] Han Yu, Zhiqi Shen, Cyril Leung, Chunyan Miao, and Victor R. Lesser. A Survey of Multi-Agent Trust Management Systems. *IEEE Access*, 1:35–50, 2013.

[136] Ray Chen, Fenye Bao, MoonJeong Chang, and Jin-Hee Cho. Dynamic trust management for delay tolerant networks and its application to secure routing. *IEEE Transactions on Parallel and Distributed Systems*, 25(5):1200–1210, 2013.

[137] Fenye Bao, Ray Chen, MoonJeong Chang, and Jin-Hee Cho. Hierarchical trust management for wireless sensor networks and its applications to trust-based routing and intrusion detection. *IEEE transactions on network and service management*, 9(2):169–183, 2012.

[138] Dexiang Wu, Guohua Shen, Zhiqiu Huang, Yan Cao, and Tianbao Du. A trust-aware task offloading framework in mobile edge computing. *IEEE Access*, 7:150105–150119, 2019.

[139] Shengli Pan, Zhiyong Zhang, Zongwang Zhang, and Deze Zeng. Dependency-Aware Computation Offloading in Mobile Edge Computing: A Reinforcement Learning Approach. *IEEE Access*, 7:134742–134753, 2019.

[140] Audun Josang and Roslan Ismail. The Beta Reputation system. In *Proceedings of the 15th bled electronic commerce conference*, volume 5, pages 2502–2511. Citeseer, 2002.

[141] Ehab ElSalamouny, Vladimiro Sassone, and Mogens Nielsen. HMM-based trust model. In *International Workshop on Formal Aspects in Security and Trust*, pages 21–35. Springer, 2009.

[142] Matthew Bradbury, Arshad Jhumka, and Tim Watson. Trust Trackers for Computation Offloading in Edge-Based IoT Networks. In *IEEE INFOCOM*

2021 - IEEE Conference on Computer Communications, pages 1–10. IEEE, 2021.

[143] Ragib Hasan, Mahmud Hossain, and Rasib Khan. Aura: An incentive-driven ad-hoc IoT cloud framework for proximal mobile computation offloading. *Future Generation Computer Systems*, 86:821–835, 2018.

[144] Sobit Thapa and Qijun Gu. Originator data security for collaborative task execution among weak devices. In *2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems*, pages 421–422. IEEE, 2013.

[145] Elisa Rondini, Stephen Hailes, and Li Li. Load sharing and bandwidth control in mobile P2P wireless sensor networks. In *2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 468–473. IEEE, 2008.

[146] Matthew Bradbury, Arshad Jhumka, and Tim Watson. Trust Assessment in 32 KiB of RAM: Multi-application Trust-based Task Offloading for Resource-constrained IoT Nodes. In *The 36th ACM/SIGAPP Symposium on Applied Computing*, SAC'21, pages 1–10, Virtual Event, Republic of Korea, 2021. ACM.

[147] Matthew Bradbury, Arshad Jhumka, Tim Watson, Denys Flores, Jonathan Burton, and Matthew Butler. Threat-Modeling-Guided Trust-Based Task Offloading for Resource-Constrained Internet of Things. *ACM Transactions on Sensor Networks*, 18(2):41, 2022.

[148] Pycom. FiPy. Available at https://pycom.io/product/fipy/. Accessed on: Sept. 27, 2022.

[149] Damien George. MicroPython. Available at http://micropython.org/. Accessed on: Sept. 27, 2022.

[150] Pycom. Pycom documentation. Available at https://docs.pycom.io/. Accessed on: Oct. 3, 2022.

[151] The Things Industries. The Things Network - Devices. Available at https://www.thethingsnetwork.org/docs/devices/. Accessed on: Sept. 27, 2022.

[152] The Things Industries. Fair Use Policy explained. Available at https://www.thethingsnetwork.org/forum/t/fair-use-policy-explained/1300. Accessed on: Sept. 27, 2022.

[153] The Things Industries. Limitations of LoRaWAN. Available at https://www.thethingsnetwork.org/docs/lorawan/limitations/. Accessed on: Sept. 27, 2022.

[154] The Things Industries. The Things Network - Network Architecture. Available at https://www.thethingsnetwork.org/docs/network/architecture.html. Accessed on: Sept. 27, 2022.

[155] The Things Industries. HTTP Integration. Available at https://www.thethingsnetwork.org/docs/applications/http/. Accessed on: Sept. 27, 2022.

[156] The Things Industries. The Things Network - Gateways. Available at https://www.thethingsnetwork.org/docs/gateways/. Accessed on: Sept. 27, 2022.

[157] Pycom. Vodafone NB-IoT Prepaid Subscription. Available at https://pycom.io/product/vodafone-nb-iot-prepaid-subscription/. Accessed on: Sept. 27, 2022.

[158] Pycom. Pybytes. Available at https://pybytes.pycom.io. Accessed on: Sept. 27, 2022.

[159] Pycom. Pybytes Integrations. Available at https://docs.pycom.io/pybytes/integrations/. Accessed on: Sept. 27, 2022.

[160] Giovanni Stanco, Alessio Botta, Flavio Frattini, Ugo Giordano, and Giorgio Ventre. On the performance of IoT LPWAN technologies: the case of Sigfox, LoRaWAN and NB-IoT. In *ICC 2022 - IEEE International Conference on Communications*, pages 2096–2101. IEEE, 2022.

[161] Open Robotics. ROS - Introduction. Available at http://wiki.ros.org/ROS/Introduction. Accessed on: Oct. 3, 2022.

[162] Jason M. O'Kane. *A Gentle Introduction to ROS*. Independently published, 2013. Available at http://www.cse.sc.edu/~jokane/agitr/. Accessed on: Oct. 3, 2022.

[163] Open Robotics. ROS 2 Documentation. Available at https://docs.ros.org/en/galactic/index.html. Accessed on: Oct. 3, 2022.

[164] Inc. Open Source Robotics Foundation. ROS on DDS. Available at https://design.ros2.org/articles/ros_on_dds.html. Accessed on: Oct. 3, 2022.

[165] Alessio Botta, Alberto Dainotti, and Antonio Pescapè. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531–3547, 2012.

[166] Giovanni Stanco, Alessio Botta, and Giorgio Ventre. DewROS: A Platform for Informed Dew Robotics in ROS. In *2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 9–16. IEEE, 2020.

[167] Nordic Semiconductor. nRF52840 DK. Available at https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dk. Accessed on: Sept. 29, 2022.

[168] Nordic Semiconductor. nRF52840. Product Specification 4413_417, 2019. V1.1.

[169] George Oikonomou, Simon Duquennoy, Atis Elsts, Joakim Eriksson, Yasuyuki Tanaka, and Nicolas Tsiftes. The Contiki-NG open source operating system for next generation IoT devices. *SoftwareX*, 18:101089, 2022.

[170] Sunfounder Learn. Smart Video Car Kit for Raspberry Pi. Available at https://www.sunfounder.com/learn/category/Smart-Video-Car-for-Raspberry-Pi.html. Accessed on: Oct. 18, 2022.

[171] CISCO. CGM-SRV IOx Module. Available at https://www.cisco.com/c/en/us/support/docs/cloud-systems-management/iox/212038-Configure-CGM-SRV-IOx-Module-on-CGR1xxx.html. Accessed on: Oct. 18, 2022.

Author's Publications

- Giovanni Stanco, Alessio Botta, and Giorgio Ventre. DewROS: platform for informed dew robotics in ROS. In *2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 9–16, 2020.
- Giovanni Stanco, Alessio Botta, Flavio Frattini, Ugo Giordano, and Giorgio Ventre. On the performance of IoT LPWAN technologies: the case of Sigfox, LoRaWAN and NB-IoT. In *ICC 2022 - IEEE International Conference on Communications* , pages 2096–2101, 2022.