SAPIENZA
UNIVERSITÀ DI ROMA

Smart and Green buildings: a bottom up approach

PhD School in Computer Science

Dottorato di Ricerca in Computer Science – XXXIII Ciclo

Candidate

Mauro Piva
ID number 1532581

Thesis Advisor

Prof. Gaia Maselli

Reviewers

Prof. Violet R. Syrotiuk
Prof. Stefano Basagni

Mar 2021

Thesis defended on 8 July 2021
in front of a Board of Examiners composed by:

Prof. Maurizio Bonuccelli (chairman)

Prof. Dario Catalano

Prof. Andrea Marin

Smart and Green buildings: a bottom up approach
Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: July 9, 2021

Author's email: mauro.piva@uniroma1.it

Abstract

Can we develop smart homes and offices which have a sustainable impact on the environment?
This crucial question has been raised by the recent dramatic rise in the diffusion of wireless IoT devices, coupled with the difficulty in disposing of exhausted batteries. In this thesis we investigated the technologies and the solutions that in the next years will populate our houses and offices, selecting and working with the most sustainable ones, paying attention to the users and to their security.
Our investigation, focused on the network side, started from the physical layer up to the application one, following the TCP/IP stack, and went through a large number of technologies, from machine learning to blockchain.
We worked on a battery free controller, named JoyTag [76]; a radio fingerprint machine learning model, which we introduce in chapter 3; a reinforcement learning based MAC protocol, APT-MAC [79]; an enhanced MAC protocol with Q-Learning and real time scheduling, ReLEDF [98]; an SDN-based smart building management software, SECY [75]; a smart building software emulator, SMARTEEX [78]; and a blockchain based framework for smart building security, called HyBloSe [77].

Contents

Chapter 1

Introduction

Thanks to the Internet of Things (IoT), a fast-growing number of buildings is becoming *smart*, connecting to the Internet an expected number of 55+ billion kitchen appliances, televisions, smartphones, utility meters, intra-body sensors, thermostats, etc. by 2025 [131, 137, 127, 152]. The proliferation of IoT devices is unstoppable, and the advantages are evident. However, are all these devices environmentally friendly? Are they secure?
During this Ph.D. we explored the sustainability and security challenges related to smart building.
The main goal is to propose, from the networking point of view, an infrastructure for smart building which is secure and environmentally sustainable.
To reach this goal we analysed the network TCP/IP stack, starting from the physical layer to the application one, and focused on the ones that have been less addressed by research.
With our goal clear and the path designed, we started our investigation meeting the first and maybe the most complex issue: the impressive number of batteries required to run IoT devices. While we could think of low power devices or rechargeable batteries, the solution is more simple: we must remove batteries. One way to remove them is through backscattering, a technology which allows to power and communicate with simple and battery-free IoT devices. With this technology we started building hardware appliances, analyzing their network performance, security and capability to resist to interference. Currently, several battery-less IoT devices have been proposed — a battery-less light switch [74], a battery-less joystick [76], a battery-free RFID camera [86], a battery-free cell-phone [128] — and commercially developed — an RF field detector, a electronic relay, and a magnetic field sensor [38]. A deep analysis of backscattering and what can be done with this technology will be introduced in chapter 2.
Due to energy budget limitations backscattering powered devices cannot use energy-hungry cryptography, and thus can be easily cloned. For this reason we investigated radio fingerprinting (RFP), a compelling approach that leverages the unique imperfections in the tag's wireless circuitry to achieve large-scale RFID clone detection. For this reason, we propose a novel training framework based on federated machine learning (FML) and data augmentation (DAG) to boost the accuracy. A description of our technique and the results is available in chapter 3.

From a single device we then moved to a set of devices, tackling Medium Access Control, but with battery free devices and inside a smart building. In particular, we consider the coexistence of several battery-free devices, with different transmission requirements — periodic, event based, and real-time — and propose a new adaptive and quick-to-learn MAC protocol, called APT-MAC, which dynamically collects information from devices without requiring any apriori knowledge of the environment. Extensive simulations clearly show the benefits of using APT-MAC, which is able to successfully deliver 97.7% of new data samples in complex scenarios, including several high traffic demanding devices such as joysticks and cameras. A complete description of APT-MAC is presented in chapter 4. However, we noticed that APT-MAC is limited in some scenarios: it does not have the concept of *context*. In practice, the same device may have completely different network needs in two different instants. To tackle this limit we leverage Q-Learning and real time scheduling, and propose a new communication protocol, called ReLEDF. This new protocol dynamically discovers devices in smart buildings, their active and nonactive status, and when active their current communication behavior — through a learning-based mechanism — and schedules transmission slots — through an Earliest Deadline First (EDF) based mechanism — adapting to different data transmission requirements. Combining learning and scheduling introduces a tag starvation problem, so we also propose a new mode-change scheduling approach. Extensive simulations clearly show the benefits of using ReLEDF, which has a comparable data delivery rate with respect to APT-MAC outperforming related solutions. Specifically, ReLEDF is up to 5 times faster in reaching an optimal channel allocation, even with more than 3 times the number of devices in APT-MAC. Real experiments are also conducted to demonstrate the applicability of ReLEDF and to validate the simulations. A complete description of ReLEDF is presented in chapter 5.

Given the communication at the MAC layer, we proceed to the management of packet routing between devices inside a smart building. It is easy to notice that today two logically connected IoT devices, even if at few meters, may be connected through a server thousand of kilometers distant. There is a complete separation between the logical flow and the network flow, that in same cases creates issues as slowness and service disruption in case of remote disconnections.
For this reason we proposed a smart building management software which connects logic and networking: when a logical connection is established between two devices, a set of SDN rules is installed in the smart building network, making the communication between the two devices efficient. We tested this tool through a smart building network emulator that we developed starting from Mininet, and which allows in few seconds to emulate hybrid scenarios that can contain hundred of simulated and real devices, connected in the same scenario.
Finally, we tackled the the security issues connected to smart building. As far as today, we noticed that the end user security is fully managed by producers of devices, and that the user must fully trust them. As we envision that an user should not trust external sources inside its house or building, we proposed a blockchain based framework for IoT device's connections, which defends users from misbehavior of remote control systems.

In the next chapters we will go through the TCP/IP stack, analysing in deep all these solutions we proposed. However, first things first. What is backscattering? We now present some brief introductions on the methodologies we used.

Backscattering

As a breakthrough in wireless communications, backscattering allows to power sensor devices and eliminate the need to have any inbuilt batteries. Many devices that operate at a relatively low power budget can be remotely powered through backscattering of RF signals, modulating the reflection of existing RF signals. Backscattering offers a considerably efficient and increasingly practical alternative to active radio circuits in existing sensor systems [149].

Backscattered signals can be of two types: ambient [45] or RFID [135] signals. Ambient backscattering harvests power from signals available in the environment such as TV [68], cellular [92], and Wi-Fi [59] transmissions. The main benefit of ambient backscattering is the exploitation of existing RF signals without requiring the deployment of a dedicated device to transmit a high-power signal to nearby devices. The main limitation is that ambient RF energy is not always available, and this can lead to reliability issues. In addition, techniques that have been demonstrated for ambient backscattering have low data rate (1kbps in the best signal conditions [68][59]). The low achievable data rate motivates applications involving occasional (or spot) data transmission, such as money transfer between smart cards or revealing misplaced objects in a grocery store, but are not suitable for applications requiring continuous and real-time communication. When the signal is weak the data rate decreases significantly. For example, when a Wi-Fi station is not transmitting, the achievable data rate using only periodic beacons reduces to $10 - 40$ bps, which is not sufficient for smart home applications. Signal availability is a related limitation. Although TV towers broadcast uninterrupted and continuous signals at all hours of the day and night, the ubiquity of the signal cannot be guaranteed, compromising the effectiveness of continuous and real-time data transmission. If the signal is weak, sensors cannot operate; they have to accumulate enough energy to perform the required action. Even in metropolitan areas where TV signals are supposed to be ubiquitous, they weaken significantly in indoor environments positioned at more than $8 - 10$ km from a TV tower.
RFID backscattering instead harvests power from signals emitted by a dedicated RFID reader [135]. In traditional RFID technology, the tags — battery free devices — absorb and reflect the high-power constant signal generated by the reader — a powered device — to send it their unique ID. With the advent of IoT, new applications of RFID technology have emerged: RFID tags can exploit the energy harvested from the reader to run some low power sensors and transmit sensed data [128][86][76][48]. However, the challenges in building battery free smart objects by exploiting RFID backscattering are multiple.

1. How to make battery-free devices to sense and transmit given that they cannot operate spontaneously on their own energy;

2. How to guarantee enough energy for multiple battery free devices at the same time;

3. How to simultaneously support heterogeneous sensor types, sensor requirements, and their different uses; and
4. How to effectively cover an entire home.

We will introduce the RFID devices we realised in chapter 2, a way to secure them in chapter 3, and two MAC protocols which allows these devices to communicate in chapter 4 and chapter 5.

Next question, what is Machine Learning?

Machine Learning

Machine learning is a sector of Artificial Intelligence devoted to the development of algorithms able to find patterns inside data and produce statistical prediction. The set of machine learning algorithms is divided into multiple categories: supervised learning, unsupervised learning and reinforcement learning. In this thesis we adopted both supervised learning and reinforcement learning.

Supervised Learning Supervised Learning algorithms require a *training* dataset of "labeled instances", which must be provided by an external actor. The algorithms use the provided dataset to extrapolate and generalize the correlation between data and labels, to learn how to classify unlabeled data. For example, these algorithms can be used to classify malicious data packages: in the learning phase, are collected a dataset of malicious and not malicious data packages; then from these labeled data packages, the algorithm infers the characteristics which indicate if a package is malicious or not; at the end of this learning phase (called also *training phase*), the algorithm can classify unseen data packages.

Reinforcement Learning Reinforcement Learning (RL) [126] is a branch of machine learning which aims at solving decision making problems. The RL setting involves an agent which has to decide an action, and an environment which receives the action, and answers to the agent with an environmental state and a reward. The *state* describes the situation in which the agent has to take an *action* and the *reward* encodes how "good" is the taken action.

Blockchain

Blockchain is a data structure in which each entry, called a "block", is sequentially and cryptographically linked to the previous one, so that none of them can be changed. The blockchain is based on the *distributed ledger* logic, which allows to track transactions in a distributed way. So far, the two most adopted blockchains are Bitcoin and Ethereum. The last is a public blockchain which also allows to deploy applications called "Smart Contracts". Once written in the blockchain, such applications cannot be modified, and are executed by a virtual machine running on all the nodes in the peer to peer network. A deeper explanation of "Smart Contracts" is available in [138]. Every account in the Ethereum Blockchain owns an amount of

Ether (ETH), i.e., coins that can be exchanged with Dollars, exactly like Euros or Yuan. An account can obtain Ether mining, i.e. doing work for the blockchain, or receiving Ether from other accounts, in exchange of services or real money. Each time that a user wants to write something on the blockchain, or wants to run some code already written in the blockchain, she has to pay a variable amount of Ether to the peers in the network behind the blockchain. As every content written on the blockchain is saved on all ledgers, writing on the blockchain is slow and has a cost.

Backscattering, Machine Learning and Blockchain. These are the three technologies that we envision will populate our smart buildings. After this overview of the technologies investigated, in the next chapters we will walk through the TCP/IP stack inside smart buildings, starting from a set of devices that have been built with the backscattering technology.

Take a seat, fasten your seat belt, relax and enjoy the ride.

Chapter 2

Battery-Free Smart Objects based on RFID backscattering

2.1 Chapter Introduction

Radio-Frequency IDentification (RFID) technology enables the re-design of personal wireless computing devices in a battery-less manner, representing a major leap forward in moving beyond chargers, cords and dying devices. Specifically, RFID backscattering powers tags up by harvesting power from signals emitted by an interrogator, (i.e., RFID reader), and makes them communicate by backscattering the incident signal [135]. The traditional RFID technology involves a set of tags — devices without any power source — that absorb and reflect the high-power constant signal generated by a powered device, namely the reader, which interrogates them to get their unique ID. With the advent of IoT, new RFID-based devices have been developed, namely sensor-augmented RFID tags, which harvest energy from the reader and exploit it to run some low power sensors and transmit sensed data. In this chapter we investigate the design of battery-free smart objects (or smart devices) based on RFID technology that can be deployed in a smart home and make the following contributions:

Build a representative set of battery-free RFID-based device types (we use the UMich Moo platform[1]) to illustrate the solution including devices that are real-time (e.g., a videogame controller, a microphone), periodic (e.g., a temperature sensor) and event based (detecting presence, a fire detector). Specifically, we present the development of a videogame controller, called SapyJoy, which is able to interact with several types of videogames.

Identify the types of devices that can be handled today and what is future work, and did extensive controlled experiments to evaluate the performance of different types of devices showing that multiple smart objects can be made battery free and what are the challenges for their coexistence in a same smart environment.

[1]The name Moo comes from the fact that it is the beefiest embedded platform in its class with the most code space and RAM for the least energy. The device also resembles a longhorn steer. https://spqr.eecs.umich.edu/moo/

Show through experiments that our newly developed devices are very fast in communicating with the corresponding applications, performing even better than commercial benchmarks. This chapter has been extracted from [76] and [18].

2.2 Battery-Free Smart Devices

Many important types of sensors — temperature, humidity, light, accelerometers, pressure buttons, analog joysticks, etc. — can be integrated with Moo tags to devise battery-free devices. The main constraint is related to tag energy consumption. Sensors should not require more than 3 V and each sensor should consume less than 10 mW in order to allow continuous sensor activity. With more demanding sensors, up to 100 mW, the sensor has to exploit a duty cycle in order to satisfy the energy constraints. As shown in [110], with deep duty cycle tuning, it is possible to power devices requiring up to 200 uA at 1.8 V with 10 hertz refresh rate. This implies that a Moo-based solution is effective for many smart devices.

In the following we present the set of battery-free devices that we built by leveraging UMich Moo Computational RFID tags. Specifically, we illustrate the breadth of solutions by building devices that are *periodic*, *event based*, and *real-time* (performing burst sensing). Then, to further show applicability we present other devices that could easily be developed, and, finally, discuss several devices that are already present in the literature (developed by others). Overall, this section demonstrates the wide variety of devices (with different rate requirements) that can be accommodated by our solution.

2.2.1 Newly Developed Devices

We built a representative set of Moo-based battery-free devices discussed in this subsection, including periodic, event based, and real-time devices.

Light Switch This is an event-based device, realized by mounting a button on the Moo Tag. When the user presses the button on the wireless and battery-less light switch, the system switches on a LED on an actuator. Depending on the application, it is possible to embed multiple buttons on the same Moo tag, to control different lights deployed inside a smart building. The logical connection between the tag switch and the corresponding light is placed inside the server.

Remote for a Tea Kettle This is an event-based remote able to switch on a kettle. It is realized by means of two Moo tags. The first one is equipped with a button and acts as a remote to activate the kettle. The second one acts as actuator, and is connected to the kettle through a relay that is activated by a reader message.

Video-game Controller This is a real-time device that is realized by mounting an analog joystick and two buttons on a Moo tag. The resulting wireless and battery-less videogame controller is able to interact with several types of videogames (e.g., adventure, action, puzzle, and role-playing games). Fig. 2.1 shows our videogame controller (called SapyJoy): a Printed Circuit Board (PCB) board connects the

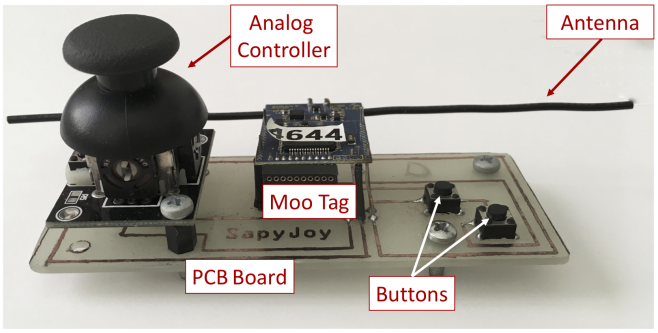


Figure 2.1. SapyJoy videogame controller: A PCB board connects the analog controller and the buttons with the Moo tag.

analog joystick and the two buttons with the Moo tag, which also has an accelerometer embedded, allowing for complex game experiences. Another version of the controller featuring only an accelerometer (no buttons and no analog joystick) was presented in [76].

Mouse A platform analogous to the SapyJoy can work as a wireless and battery-less mouse by interfacing its x and y axes with the pointer on the screen. We embedded the information regarding the analog controller inside packets transmitted by the tag and realized a virtual mouse driver able to decode this information and translate it into the pointer position.

2.2.2 Devices that can be Built

By studying the technical characteristics of different sensors and actuators we identified the set of devices that can be easily developed. The following is a description of some of the devices that can be built by leveraging Moo tags. This increases the applicability of our solution for battery-less smart homes.

Event Detector Embedding a smoke sensor on the Moo tag it is possible to devise a fire alarm[2]. Another detectable event is detecting presence through a motion sensor[3]. In general, any ultra low power sensor able to detect an event can be exploited to build an event detector.

Remote for Appliances Any appliance that can be actuated by a relay — coffee machines, shutters, doors, air fans, etc. — can be controlled by a battery-free remote, by mounting a button on a Moo tag — the remote — and connecting the appliance to another Moo tag — the actuator — through a relay.

Infrared Remote (IR) Commander Embedding an ultra low power IRDA emitter [4] on the Moo tag we can create an IR remote controller for any appliance

[2]https://goo.gl/jPLfva
[3]http://www.ti.com/lit/ug/tiducu5/tiducu5.pdf
[4]http://bit.ly/2HiipVj

equipped with infrared interface, prolonging the lifetime of less recent and technological appliances. In this case the Moo Tag must be placed in front of the IR receiver on the controlled device. Even if the IRDA emitter consumption is quite high (170 uA for transmission) we expect to have some seconds between a command and the next one, enough to recharge the accumulator.

Environmental Sensors Light, humidity, presence, and other sensors can be mounted on the Moo tag to allow environmental monitoring. The number of sensors that can be mounted on a Moo tag depends on the number of I/O ports on the microprocessor and the amount of energy available.

2.2.3 Already developed devices

There are a few battery free devices that have been already developed by others.

Temperature Sensor It is a device that periodically senses temperature and reports sampled data to the server to allow environmental monitoring [135].

Camera As shown in [86] it is possible to implement an RFID tag with an embedded camera able to take pictures and transfer them with the power harvested by the RFID antenna.

Cordless Phone As demonstrated in [128] it is possible to realize a simple phone, able to stream voice and audio from and to the reader. This device can be used as a phone, as a microphone, or a small sound diffusion system.

Information Display Integrating an ultra low-power electronic ink (E Ink) display on the moo tag it is possible to realize a display for several types of information. A first solution is given in [36], where a number of wearable displays (shoes, t-shirt, etc.) have been realized using electromagnetic induction and e-ink displays. We believe that other devices can be deployed. For example, displaying the current time it is possible to realize a battery free clock. The display can be useful also to show messages from authorized people outside the building. For example, in assisted living applications, remote relatives or caregivers can send reminders to take medication or perform some actions on people inside the home.

Monitoring Systems The work in [150] shows how to create a series of sensors able to detect doors opening and monitor water usage of a drinking tap. Modifying the antenna circuit of tags, it is possible to open and close it in order to activate or deactivate the tag. The reader, depending on the tag status can understand if a door is open or closed. For example, when the door is open the tag is not activated, when it closes the tag activates.

2.3 Experiments with Battery-free Smart Devices

We now evaluate the performance of our battery-less smart devices, benchmarking their performance against those of commercial battery-powered devices.

2.3.1 Testbed

We implemented prototypes for two videogame controllers, a mouse, a light switch, and a temperature sensor, using the UMich Moo Computational RFID tag. To interact with our prototypes we use a USRP RFID reader equipped with two RFID antennas, and a server that interconnects the RFID reader with smart-home applications. The Moo tag receives the reader signal and uses it to harvest operating power using the RFID circuit. The harvested power runs on-board sensing, encoding of measurement data, cyclic redundancy check (CRC) error coding, and backscatter communication to wirelessly send data back to the reader. The communication protocol between the reader and the tags is based on the EPC Gen 2 Class 1 standard [44], that has been modified to acquire data from sensors and store them in the buffer that is traditionally used to maintain the tag ID. As only a few bits (e.g, 8 bits) are sufficient to represent the tag's ID, the remaining, typically 88 bits, can be used to send sensed data. We limited the data field to 1 byte for tag ID and 6 bytes for data samples (including 4 CRC bits). This number guarantees low packet error rate — confirmed by our experimental study — and enough space for data samples for all devices except the camera, which would require data fragmentation in case of longer payloads.

2.3.2 Metrics

We evaluated the performance of our prototypes by measuring the following metrics:

- **Reaction time** is the time since the generation of new sensor data to the corresponding action on the recipient application. This is an application layer metric. In the case of the joystick, it measures the time between an action on the joystick (e.g., a button pressure), and the corresponding event on the videogame application. In the case of an environmental sensor, this metric measures the time between the generation of new sensor data and the corresponding reaction on the recipient actuator (e.g., a presence sensor activating a camera).
- **Packet delay** is the time since the generation of new sensor data to its reception by the reader.
- **Throughput** is the number of bits that the reader receives per unit of time.
- **Packet error rate** is the fraction of incorrect packets received by the reader over the total number of sent packets.

While it is possible to measure the last three metrics (i.e., packet delay, throughput, and packet error rate), at the reader side, reaction time requires a more complex procedure because of synchronization issues between sensors and actuators (e.g., the player's action and the corresponding game reaction). Besides the packet delay at

Table 2.1. Reaction Time for different controllers.

Device	Reaction Time (ms)	CI
SapyJoy	92.92	[82.31-102.92]
Commercial Controller	104.58	[96.31-112.85]
Commercial Mouse	110.41	[103.09-117.74]

the network layer, reaction time includes also the time it takes for the packet to proceed up the protocol stack at the recipient. For these reasons we use a digital camera to measure reaction time. The camera frames the sensor and the actuator at the same time, so that we have a unique clock to record events (e.g., in the joystick case, the camera frames the controller and the screen to record button pressures and corresponding actions on the screen). In this way we can measure time also for commercial devices for which is impossible to act at the software level.

2.3.3 Results on Single Devices

We now evaluate the feasibility of devices we built.

Video-game Controller

The first battery-free device we evaluate is our video-game controller, SapyJoy, that is compared with two commercial Bluetooth devices: a Logitech controller per console (cordless precision controller for playstation3) and a Logitech wireless mouse (cordless optical mouse for notebooks). The three controllers were used to play with navigating video-games — which have an update rate of 30 frames per second (fps) — as well as shooting video-games — which have an update rate of 60 fps. The three controllers were all good and we could not notice any difference in playing ability. To quantify this ability, and considering the difficulty in identifying a reaction to a user's action in a video-game, we implemented a simple application that represents the joystick through arrows and buttons through circles. When the player moves the joystick, the corresponding arrow changes color on the screen (for example, if the player moves the joystick ahead, then the top arrow changes color). Analogously, when the player presses a button (i.e., the right one), the corresponding circle (i.e., the circle on the right of the screen) changes color (see Fig. 2.3).

Table 2.1 shows the observed reaction time (with 5% confidence interval) for the three devices, measured through a videocamera framing at the same time the controller and the screen (the update rate of the viedocamera is 60 fps). SapyJoy takes on average 92.92 ms to see the outcome of a button pressure on the videogame, while the two commercial devices — controller and mouse — take, respectively, 104 ms and 110 ms to perform the same operation. These results show that SapyJoy is even faster than battery-powered devices.

Reaction time includes the packet delay at the network layer, plus the time to deliver the packet from the reader to the server, plus the time to produce the game commands corresponding to the actions performed by the user and send them to the videogame application. Thus, if we measure only the packet delay at the network layer, SapyJoy takes on average only 4.79 ms to deliver sensed data to the reader

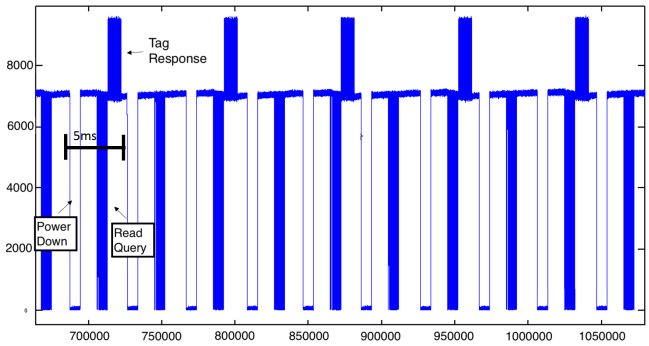


Figure 2.2. Matched filter for SapyJoy.

Table 2.2. Reaction time for battery-free light switch and mouse.

Device	Reaction time (ms)	CI
Light Switch	62.91	[67.41 - 73.41]
Mouse	92.92	[82.91 - 102.92]

(note that we cannot measure this metric for the commercial devices because they are not programmable).

Analyzing the matched filter for our SapyJoy we observed that although the packet delay is below 5 ms, to achieve the best performance —avoid any reader-tag collision due to any possible delay from the tag — the reader can issue a new query every 6 ms (see Fig. 2.2). By querying tags at this interval of time, the throughput at the reader is 6.6 Kb/s (including sensor data and protocol control bits), with less than 1% packet error rate.

Light Switch and Mouse

Now we evaluate our battery-free light switch and mouse. We use again a videocamera framing at the same time the sensor and actuator. In the case of the light switch, the sensor is the tag equipped with a pressure button while the actuator is a tag with a LED on-board. In the case of the mouse we use the same platform as for the joystick. The mouse communicates with an application showing cursor movements and button pressures through a circle that moves on the screen and changes color when a button is pressed (see Fig. 2.3).

Table 2.2 shows the reaction for the two devices. The light switch takes only 62.91 ms to collect data from the pressed button, send it to the actuator, and switch on the LED. Although we do not have benchmarks to compare with, we believe that this time would satisfy any stringent application requirements.

Reaction time increases to 92.92 ms in the case of the mouse, as for the videogame, data has to reach the final application on the server, taking some time to ascend the protocol stack. However, even in this case the system is very reactive, with the user perceiving a real-time communication.

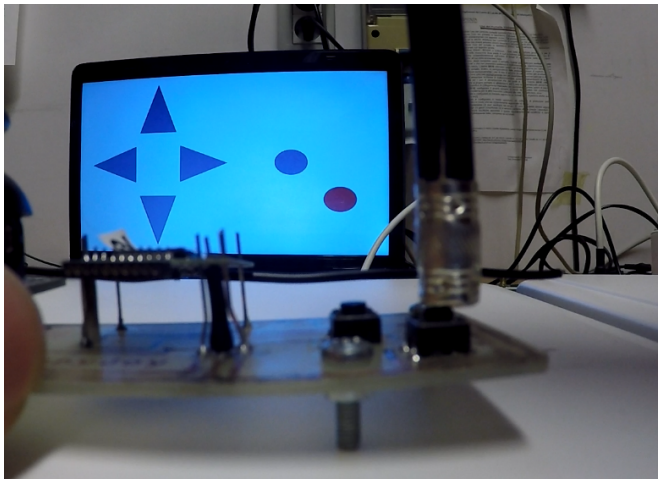


Figure 2.3. Button pressure on the battery-free mouse and corresponding action on the screen.

2.3.4 Results with Multiple Devices

We now present experimental results that evaluate multiple battery-free devices simultaneously, so that we can evaluate their interoperability. We run experiments with three devices working at the same time: two environmental sensors — temperature and presence — and a video-game controller (our SapyJoy). The devices are queried (and hence transmit sensed data) following a time division multiple access approach, which provides different time slots to different devices in a cyclically repetitive frame structure. The first difference compared to experimentation with a single device is reaction time. If a device is queried at each slot, the reaction time is clearly shorter compared to the case in which it is queried once every multiple slots. The outcome of our experimentation is that the reaction time increases significantly (i.e., 200 ms) compared to when it works alone (i.e., 92.92 ms). This delay would certainly increase if the number of transmitting devices increases, making interoperability a challenge as the joystick may experience too long delays.

2.4 Lesson Learned

Our experimentation highlights two big challenges for the design and deployment of battery free environments, like smart homes, in which there are many sensors and smart devices, such as surveillance cameras, smoke, presence, temperature, light sensors, smart meters, and many others.

The first challenge concerns interoperability of devices. Although results clearly show the feasibility of battery-free RFID-based smart objects, whose performance are comparable to that of the battery-powered counterparts, their coexistence cannot be taken for granted. When multiple devices operate simultaneously, the reaction time increases significantly compared to the case of a device working alone. In addition, an equal assignment of channel resources would not satisfy devices needs. Multi-kind multiple battery-free devices, operating simultaneously, have widely varying communication requirements, in terms of data transmission, ON/OFF activity, and

deadlines. To pick an example, a joystick may sense no changes for hours (while it is OFF), and then start sensing new data (while used for playing) at very different rates (from a few milliseconds to one or more seconds), depending on the game type and player activity. Thus, a communication protocol for battery-free devices should schedule channel access such that device requirements are satisfied and data is delivered in time. A first solution in this direction is given in [79].

The second big challenge regards operational limits of RFID technology: communication range is a major obstacle for the real-world implementation of this low-cost technology. The transmission power of our reader is $P_t = 0.5W$, and the communication range between the antennas and the tags is below one meter. With this technology it is possible to realize smart devices such as the joystick or the light switch, but not a videocamera, which requires real-time streaming. Increasing the power of the reader (e.g., up to $P_t = 1W$) would allow a longer transmission range (up to 3 meters) between the reader's antennas and the tags, but it would not satisfy real-time frequencies. The need for technological improvement is clear. A first attempt toward more efficient devices in terms of bit rate, distance, and energy is given in [148], where the RFID device is powered not only by RF harvesting but also by a small solar panel (3 cm x 3 cm), reaching a transmission range of 21 feet and a maximum bit rate of 21.7 kb/s. This trend is confirmed in [57], where the use of photovoltaics increases the transmission range by providing additional power to the RFID tag integrated circuit.

2.5 Can we really use Backscattering?

The last decade has witnessed an explosion of wireless devices that created an ever-increasing demand for batteries. In this chapter, we demonstrate that RFID technology is a key enabler for realizing many battery-less smart devices, performing real-time, periodic, and event based sensing. Most of these devices are doable now — we realized light switches, remotes for a tea kettle, videogame controllers, the mouse, and studied how to realize event detectors, IF remote commanders, and remotes for general appliances — while others are more difficult to realize (e.g., videocameras). Results clearly show the feasibility of our approach, but also highlight the need for new communication protocols that can distinguish between less and more demanding devices.

However, the strong power limits of this technology introduces a security challenge on how to defend RFID tags from cloning and man in the middle attacks. In the next chapter we investigate a radio finger print based approach which allows to distinguish between two tags that are apparently the same.

Chapter 3

The Tags Are Alright: Robust Large-Scale RFID Clone Detection Through Federated Data-Augmented Radio Fingerprinting

3.1 Chapter Introduction

Millions of RFID tags are pervasively used all around the globe to inexpensively identify a wide variety of everyday-use objects. One of the key issues of RFID is that tags cannot use energy-hungry cryptography, and thus can be easily cloned. For this reason, Radio Finger-Printing (RFP) is a compelling approach that leverages the unique imperfections in the tag's wireless circuitry to achieve large-scale RFID clone detection. Recent work, however, has unveiled that time-varying channel conditions can significantly decrease the accuracy of the RFP process. Prior art in RFID identification does not consider this critical aspect, and instead focuses on custom-tailored feature extraction techniques and data collection with static channel conditions. For this reason, we propose the first large-scale investigation into RFP of RFID tags with dynamic channel conditions. Specifically, we perform a massive data collection campaign on a testbed composed of 200 off-the-shelf identical RFID tags and a software-defined radio (SDR) tag reader. We collect data with different tag-reader distances in an over-the-air configuration. To emulate implanted RFID tags, we also collect data with two different kinds of porcine meat inserted between the tag and the reader. We use this rich dataset to train and test several convolutional neural network (CNN)-based classifiers in a variety of channel conditions. Our investigation reveals that training and testing in different channel conditions drastically degrades the classifier's accuracy. For this reason, we propose a novel training framework based on FML and DAG to boost the accuracy. Extensive experimental results indicate that (i) our FML approach improves accuracy by up to 48%; (ii) our DAG approach improves the FML performance by up to 19% and the

single-dataset performance by 31%. To the best of our knowledge, this is the first work experimentally demonstrating the efficacy of FML and DAG on a large device population. To allow full replicability, we are sharing with the research community our fully-labeled 200-GB RFID waveform dataset, as well as the entirety of our code and trained models [1].

3.2 Introduction to Radio Fingerprinting

The cost-effectiveness of Radio Frequency Identification (RFID) tags is becoming a fundamental driver for their significant expansion into the Internet of Things (IoT) ecosystem [136, 4].

Since RFID can be attached to cash and other valuable objects, and also implanted into animals and people, their widespread usage has raised serious security and privacy concerns [10, 55, 2]. Critically, low-cost RFID tags cannot support complex cryptography beyond hash functions [73], and lightweight cryptographic techniques have been proven as insecure [109, 129, 153]. For this reason, RFID tags can be easily tampered with and their functionality compromised [11, 134]. In this chapter, we consider the *cloning* security vulnerability, where tag data is eavesdropped during the reading process by an adversary and then replayed by a rogue device, for example, a SDR [3]. It is intuitive that the resilience of RFID tags to cloning attacks is strongly correlated to their applicability in critical applications. For example, if thousands of cloned tags are injected into the supply chain, companies would lose track of their assets and thus suffer severe financial loss [72]. Perhaps even more worrisome, cloning tags could have serious consequences to the well-being of individuals, as they are extensively used in credit cards, passports, badges, and health care, among others [22].

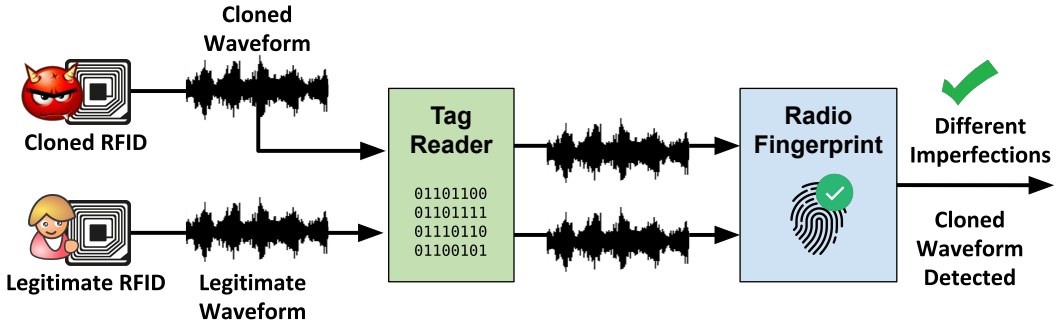


Figure 3.1. Radio fingerprinting leverages unique circuitry imperfections that are hardly replicable by an adversary to identify cloned RFID tags.

Approach. To address the tag cloning issue without relying on cryptography, some prior work has proposed approaches based on RFP [34, 108, 95, 145, 146, 19, 33, 49, 28]. Figure 3.1 summarizes the RFID tag cloning issue and how RFP addresses the issue. In short, RFP leverages small-scale hardware-level imperfections typically found in off-the-shelf RFID front-end circuitry, such as frequency and sampling offset, I/Q imbalance, phase noise, and harmonic distortions [54]. By estimating the impairments on the received waveform and associating them with a given device, a

[1] The repository is available at mauropv.github.io

unique identification of the device can be obtained [141]. Prior work – discussed in details in Section 3.3 – relies on protocol-specific feature-extraction techniques such as dynamic wavelet fingerprint [19] or minimum power response at multiple frequencies [95] to extract hardware impairments, which can only be applied to a given family of RFID tags. In stark contrast, in this work we leverage machine learning (ML)-based techniques – specifically, CNNs – to create feature-agnostic, general-purpose, and optimizable RFP classifiers [105]. The key advantages of CNNs with respect to traditional ML are that (i) CNNs employ a high number of parameters and thus can distinguish very high population of devices; and (ii) by learning filters operating over unprocessed I/Q samples, they avoid application-specific computational-expensive feature extraction/selection algorithms [103, 91].

Existing Issues. Prior research has hinted that dynamic propagation environments may cause the CNN's accuracy to plummet significantly [5]. To validate the severity of the problem, *we experimentally show in Section 3.6.2 that training and testing on different datasets reduce accuracy by 90% on the average.* This is not without a reason. Indeed, one of key assumptions of CNNs, is that training and testing datasets are independent and identically distributed (i.i.d.). On the other hand, this is hardly the case in the wireless domain, where (i) time-varying oscillations in temperature and voltage may cause the hardware impairments to change over time; and (ii) interference and noise levels can modify waveforms in a dynamic and unpredictable manner. Existing strategies leverage the computation of finite input response (FIR) filters applied at the transmitter's side to partially compensate the channel action [101]. However, this strategy cannot be used for RFID, *since tags are not software-defined and cannot modify their physical-layer waveforms in any circumstance.* Also, these strategies maximize the fingerprinting accuracy for a given wireless node only, while we want to improve the performance of the whole population of RFID tags. For this reason, we need to find alternative strategies. This critical issue is further complicated by the *current lack of rich, large-scale datasets for RFP of RFID tags, which makes the replication of existing research impossible.* It is easy to observe that without a common benchmark, all the innovation in the field will be stymied, since every paper can claim to be "better than the previous one". This calls for the generation of public-domain datasets that can be used by the research community at large.

Technical Contributions. We summarize the novel contributions to the state of the art provided:

• We perform a massive data collection campaign on a testbed composed of 200 off-the-shelf identical EPC RFID tags [44] and a USRP2 SDR tag reader. We collect data with different tag-reader distances (20cm, 50cm and 100cm) in an over-the-air configuration. To obtain even more challenging propagation scenarios and emulate implanted RFID tags, we also collect data with two different kinds of porcine meat (i.e., with different texture and thickness) inserted between the tag and the reader, and two reader distances (20cm and 50cm). This dataset is used to train and test several CNN-based classifiers based on classic cross-entropy minimization. Our experimental results on the collected 7 datasets reveals that training and testing on different channel conditions decreases the RFP performance by up to 90%;

• We propose a federated data-augmented training framework for RFID fingerprinting to address the lack of generalization of standard techniques. Our framework

is based on FML, which has recently emerged as a powerful tool at the intersection of artificial intelligence and edge computing [83]. In short, FML allows the periodical fusion of locally-trained models by averaging the parameters over a number of training epochs, thus eliminating the need to stream waveforms captured by the tag readers to a centralized server. *To the best of our knowledge, no prior work has ever explored the usage of FML for RFP improvement.* Our framework also leverages the concept of DAG to improve the robustness of the original models [94, 115, 84]. In short, the key rationale of DAG is that by inserting into the dataset "noisy" inputs obtained by modified original inputs, the classifier will be stronger to changing spectrum environments;

• We perform an exhaustive training and testing campaign where our two training strategies are extensively evaluated and compared with baselines. Experimental results indicate that (i) our FML-based training improves accuracy by up to 48% with respect to the single-dataset scenario; and (ii) our DAG-based training improves the FML performance by up to 19% and the global performance by 31%. In stark contrast with existing work, *we pledge to share with the research community our fully-labeled 200-GB RFID waveform dataset, as well as the entirety of our code and trained models.* This will allow complete replicability and verification of results as well as a benchmark for further work in the field.

3.3 Background and Related Work

Radio fingerprinting (RFP) has been extensively used in the Wi-Fi [21, 132, 5, 101] and ZigBee standards [93]. RFID anti-cloning through RFP has also been investigated over the last years [34, 108, 95, 145, 146, 19, 33, 49, 28]. For an excellent and exhaustive survey on the topic, the reader is referred to [22]. The seminal work [34] applies RFP to 50 HF tags, achieving 2.43% error rate. Among others, the features used are based on the Hilbert transform. Romero *et al.* [108] analyze how electromagnetic signatures can help detect counterfeit HF tags. However, their analysis leverages an oscilloscope with a sampling rate of 20 GHz, which is beyond the capability of off-the-shelf readers. Periaswamy *et al.* [95] leverage the minimum power response at multiple frequencies (MPRMF) to distinguish RFID tags, achieving an accuracy of 90.5% with a population of 100 tags. Zanetti *et al.* [145] use time- and spectrum-level domains to fingerprint 70 UHF tags, achieving 71% accuracy. Bertoncini *et al.* [19] consider 146 tags belonging to 3 different manufacturers, with a sampling rate of 1.5MegaSamples/s. Protocol-specific techniques such as wavelet packet decomposition and higher order statistics are leveraged to transform the signal into an image which is then processed by a support vector machine (SVM). They leverage the full EPC transmission, while we focus on the RN16 portion to avoid learning protocol-dependent features. More recently, Han *et al.* [49] proposed a study where tag-reader distance and orientation are considered in evaluating the robustness of the RFP process. However, these experiments were conducted with a small population of 30 devices, while we consider 200 tags in this work. Chen *et al.* proposed a large-scale study by considering tags using C1G2 standard [28]. Unfortunately, we cannot compare our learning-based approach with [49, 28] and the rest of previous work since the datasets were not shared with the community.

Convolutional neural networks (CNNs) have been recently used for a number of

different wireless applications [70, 51, 103]. Although some work has explored the usage of CNNs for RFP purposes [112, 5, 106, 101], to the best of our knowledge no prior work has explored the usage of CNNs for RFP of RFID tags. Moreover, as far as we know, no work has proposed the usage of federated machine learning (FML) to improve the performance of CNN-based RFP. Although this problem is extremely relevant, only recently it has received attention from the community [101, 139, 116]. Although very similar in target, the solution proposed in [101] assumes that the transmitter is able to modify the transmitted waveform, which is not the case for RFID tags since they are passive devices. Most importantly, the approach in [101] optimizes the accuracy for a single device at a time. In this work, we propose techniques that improve the model accuracy for all the devices in the dataset. Closer to our work, Xie *et al.* [139] and Soltani *et al.* [116] presented DAG schemes for RFP. The latter introduced a scheme that involves DAG at the transmitter's side, which is not applicable to RFID tags. Moreover, the schemes in [139, 116] are validated on simulated datasets with a population of only 10 devices. Neither of the datasets are shared, including the large-scale dataset in [116], which makes the results not replicable.

3.4 Proposed Training Framework

We now present our framework for federated data-augmented radio fingerprinting. We first describe the considered neural network in Section 3.4.1, then we introduce federated RFP and data-augmented RFP in Section 3.4.2 and 3.4.3, respectively.

3.4.1 Neural Network Description

In line with recent existing work on Wi-Fi RFP [101, 113, 112], we leverage a convolutional neural network (CNN) to extract and classify the unique imperfections of RFID tags. Specifically, a convolutional layer (CVL) drastically reduces the number of trainable parameters in dense networks by computing filters that "move" across an input tensor and "react" to a given pattern in the input [65]. More formally, by defining d and w as depth and width, a convolutional layer consists of a set of F filters $\mathbf{Q}_f \in \mathbb{R}^{d\times w}, 1 \leq f \leq F$, where F is also called the layer depth. Each filter generates a *feature map* $\mathbf{Y}^f \in \mathbb{R}^{n'\times m'}$ from an input matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$ according to the following:

$$\mathbf{Y}^f_{i,j} = \sum_{k=0}^{h-1}\sum_{\ell=0}^{w-1} \mathbf{Q}^f_{h-k,w-\ell} \cdot \mathbf{X}_{1+s\cdot(i-1)-k,1+s\cdot(j-1)-\ell} \tag{3.1}$$

where $1 \leq i \leq n'$ and $1 \leq j \leq m'$. For simplicity, Equation (3.1) assumes input and filter dimension equal to 2. This formula can be generalized for tensors having dimension greater than 2. The $s \geq 1$ is called *stride*, $n' = 1 + \lfloor n + d - 2 \rfloor$ and $m' = 1 + \lfloor m + w - 2 \rfloor$. The matrix $\mathbf{X}$ is assumed to be padded with zeros, *i.e.*, $X_{ij} = 0\ \forall i \notin [1, n],\ j \notin [1, m]$. For more details on CNNs, the reader may refer to [46].

Figure 3.2 depicts the structure of the CNN we utilize. Our proposed CNN has been implemented in Pytorch and is available at [71]. Our network takes as input

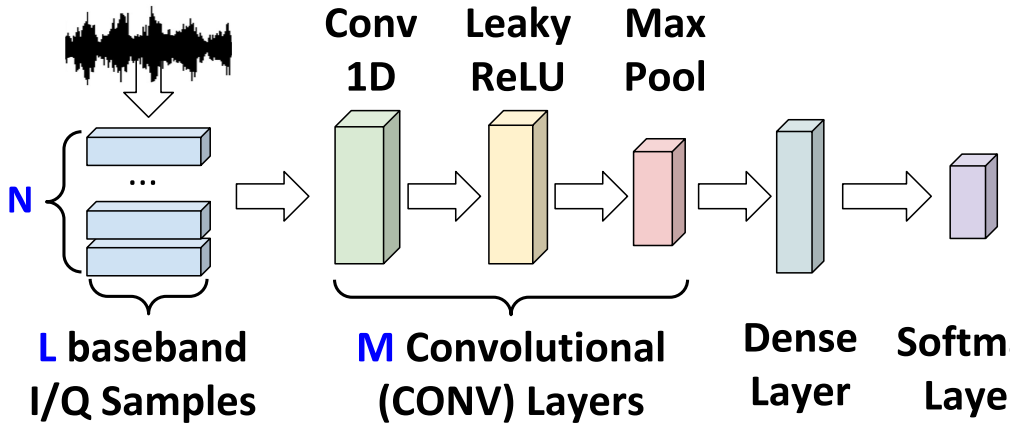


Figure 3.2. Our CNN model for RFID RFP.

a tensor of $N \times L$ size, where L is the number of baseband I/Q samples in each slice, N is the number of available signal slices. To adapt the CNN concept to our problem, as well as reducing computational burden and number of parameters, we will use a series of convolutional (CONV) layers, each composed of a Conv1D layer, a leaky rectified linear unit (LeakyReLU) as activation function, *i.e.*, $\sigma(x) = 0.1 \cdot x$ if $x < 0$ and x otherwise, and a *max pooling layer* (MaxPool), which computes the maximum value out of 1×2 regions of the output (and thus, cuts the output in half). The input of the network is thus convolved by a series of M CONV layers. Conv1D layers are composed of 25 filters with kernel size 3.

The last MaxPool is connected to a dense layer which has as input the dimension of the last MaxPool and as output the size of the population of RFID tags. We also add a flattening layer, which converts the data into a 1-dimensional vector. This vector is finally fed to the fully-connected dense layer, which is connected to a softmax layer. The loss value of each iteration is calculated with the Cross Entropy Loss, which represents a combination of a softmax and a negative log likelihood, and can be formalized as

$$\mathcal{L}(\Psi, \tau) = -\Psi[\tau] + log\left(\sum_{i=0}^{|\Psi|} e^{\Psi[i]}\right) \tag{3.2}$$

with Ψ as the array of the scores of each class, and τ as the index of the correct class. This means that effectively, in the released implementation, the softmax layer is not included in the network definition, but is executed at the loss calculation. We train our network using an Adam optimizer [60] with a learning rate of 0.001. The network has been trained for up to 100 epochs.

3.4.2 Federated Radio Fingerprinting

To boost the accuracy of locally-trained CNN models, we propose a training framework based on the concept of federated machine learning (FML). Specifically, FML attempts to address distributed learning problems where the data is non identical and independently distributed (i.i.d.). Our key intuition is to leverage this capability

to address the channel problem in RFP, since any particular tag reader's local dataset cannot capture all the possible channel distributions. Finally, FML is perfect in scenarios where tag readers are handled by different entities, which may not want to share their local dataset for privacy reasons.

Another key reason is that training the model locally and sending only the parameters reduces drastically the bandwidth utilization, which can be a critical parameter in an edge-based environment. To give an example, in this work we sample the RFID signal at 5 mega samples per second (MS/s) and produce for each sample a complex number representing the I/Q value, which occupies 4 bytes. This yields a throughput of 20 MB/s. Such a throughput would severely congest the network, especially in an IoT scenario. The weights of our models occupy up to 2.48 MB, and are only exchanged periodically as explained below.

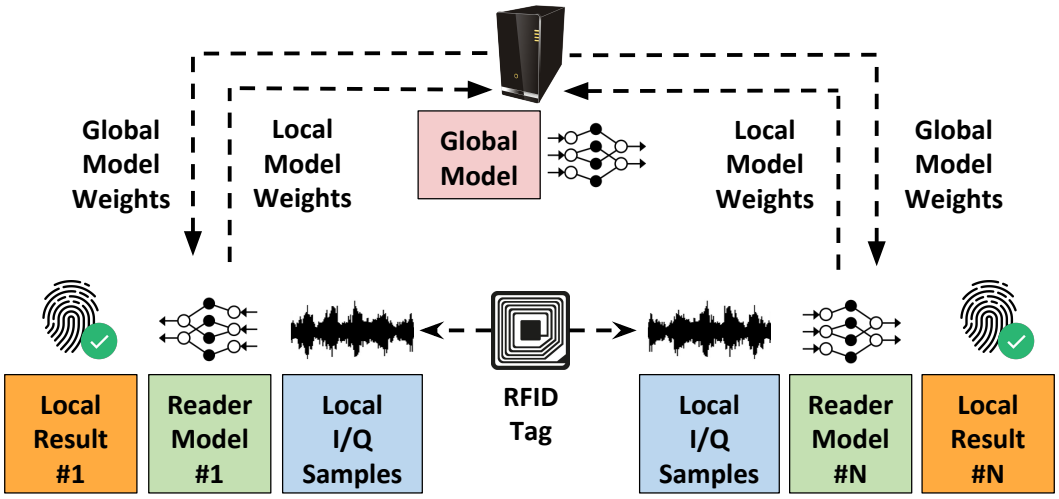


Figure 3.3. Federated Radio Fingerprinting framework.

Figure 3.3 summarizes our federated RFP approach. We consider a scenario where R tag readers (i.e. clients) collect I/Q samples from a tag, train a local model M_r and adopt the following synchronous update schema. Intuitively, at each round all the readers perform a training of the local model M_r with the locally available data. Each reader then sends the weights of M_r to a central server. The weights received from all the readers are averaged by the server and then sent back to the tag readers, which update the local models with the updated weights.
More formally, for the training we assume that each reader $r \in R$ has a set of inputs X_r and the corresponding set of labels Y_r, and that $X = \bigcup_{r \in R} X_r$, $Y = \bigcup_{r \in R} Y_r$. The target of the server is to minimize a global loss function $f(w) = loss(X, Y, w)$, where *loss* refers to the prediction of (X, Y) with weights w. Without RFP, we could describe the global loss function with the next two equations:

$$\min_{w \in \Re^d} f(X, Y, w) \quad \text{and} \quad f(X, Y, w) = \frac{1}{|X|} \sum_{(x,y) \in (X,Y)} f(x, y, w) \tag{3.3}$$

Introducing the RFP, we can reformulate formula 3.3 as follows:

$$f(w) = \sum_{r \in R} \frac{|X_r|}{|X|} F_i(X_r, Y_r, w) \quad \text{with} \tag{3.4}$$

.

$$F_i(X_r, Y_r, w) = \frac{1}{|X_r|} \sum_{(x,y)\in(X_r,Y_r)} f(x, y, w) \tag{3.5}$$

In our implementation of the federated averaging we expect the whole set of readers to be always active, as differently from other federated scenarios they are not power constrained. For each tag reader we defined a fixed learning rate $\eta = 0.001$. Each tag reader then computes $fp_r = \nabla F_r(w_t)$, as the gradient obtained with the weight w_t at time t. The central server aggregates these weights producing

$$w_{t+1} = \sum_{r\in R} \frac{1}{R} F_r(w_t).$$

This means that for each round of the federated learning all the readers run an epoch of the model training, and then share the resulting weights with the central server. This approach however produces high instability through the first epochs, with the local accuracy which is drastically reduced at every federated average. In order to stabilize the model early, we let a single reader produce an initial model training for up to 10 epochs. This model is then shared with the other readers, and used as initialization for the training, letting local models start from this early trained model, and not from an empty one. Starting local models from a random state seems to produce local models that have a very weak result when federated. This bootstrap produced an improvement in the number of epochs required to make the accuracy converge, with a reduction of up to 20 epochs. We also evaluated a trade off between data segregation and sharing, sharing a subset of all the data and centrally training a model, which is then used as starting point for the local model training. While this change produces an increased initial improvement on the accuracy, it does not produce an effective advantage, as both the models with data shared and without converge to a comparable accuracy.

We show in Section 3.6.3 that our federated RFP improves accuracy by up to 48% with respect to a local-only learning scenario.

3.4.3 Data-Augmented Radio Fingerprinting

The federated learning approach introduced in Section 3.4.2 is able to improve the robustness of a model merging data coming from multiple sources. To further improve the robustness of local models and learn channel- and interference-independent features, we leverage federated data augmentation (DAG). *We will show in Section 3.6.4 that through our approach, we are able to make our model more robust, and to dramatically improve the performance of the federated approach up to 19%.*

DAG is a machine learning technique that produces a slightly modified version of a dataset to increase its size and improve robustness [84, 115, 94]. To the best of our knowledge, this is the first time that this technique is applied to RFID tag fingerprinting and with a population of 200 devices. To perform RFP, it is possible to exploit DAG on both the devices involved in a communication, i.e. tag and reader [116]. However, to apply data augmentation on the tag side we would have to modify it, reducing the real applicability of this work.

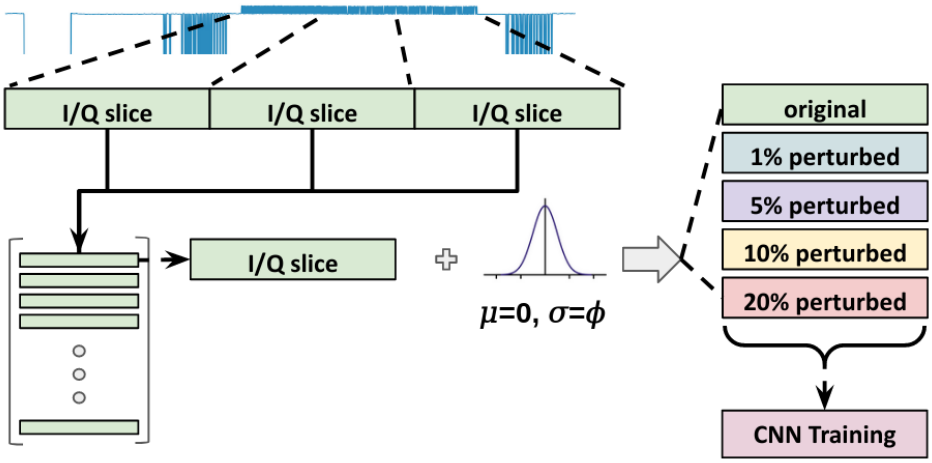


Figure 3.4. Data-augmented Radio Fingerprinting.

Figure 3.4 shows our data-augmented RFP approach. As we do not want to modify commercial RFID tags, we can apply data augmentation only at the tag reader's side, and we did it through Additive White Gaussian Noise (AWGN). Specifically, for each tag t and for each scenario we collected around 2000 communications. Each communication, C_t, is composed of 3400 I/Q sampled at 5 MS/s. We defined an array of perturbation coefficients $\Phi = \{0.20, 0.10, 0.05, 0.01\}$. To transform the signal in a machine learning readable input, we take a communication C, composed of s complex numbers, and decompose the real and the imaginary components in a matrix of shape (s,2). Depending on the window size L, we slice these matrices in contiguous sub-matrices of length L. Each of these matrices represents a tensor that will be fed to the machine learning model. These sub-matrices compose X, which is the set containing the input of the machine learning model. Thus, considering X as the model input:

$$X_{aug} = \bigcup_{\phi \in \Phi} \{A(x, \phi)_{\forall x \in X}\} \tag{3.6}$$

$$A(x, \phi) = \begin{cases} x^r = x^r + \mathcal{N}(\mu = 0, \sigma = \phi * avg(x^r)) \\ x^i = x^i + \mathcal{N}(\mu = 0, \sigma = \phi * avg(x^i)) \end{cases} \tag{3.7}$$

with x^r and x^i representing the real and imaginary parts of the sampled inputs, relatively the first and the second columns of the matrices in X, and $\mathcal{N}$ the normal distribution drawn with $\mu = 0$ and $\sigma = \phi * avg(x)$, with $avg(x)$ as the mean of x. The final input for the machine learning model is composed of the union of the sampled signal and the augmented one, $X = X \cup X_{aug}$.

3.5 Data Collection Campaign

We first introduce some background information on the EPC-Gen2 standard in Section 3.5.1, then we introduce our experimental setup and dataset structure in Section 3.5.2.

3.5.1 Background on EPC-Gen2

On the software side, almost all the UHF RFID tags communicate through the EPC-Gen2 standard [44]. In short, the reader delivers power to the tags by emitting a Continuous Wave (CW) and begins all reader-to-tags signaling with either a preamble or a frame synchronization. The preamble precedes a Query command and denotes the start of an inventory round, specifying the number of slots that will be issued in the frame. Each slot is signaled by the reader with a frame-sync message. Each tag randomly selects one slot and replies by sending a 16-bit random or pseudo-random number (RN16) when such slot occurs. If no tags reply, the slot is idle, the reader truncates it and signals the next slot. If one or more tags reply, the reader sends back the received RN16 that can be single (if only one tag transmitted) or colliding (if multiple tags transmitted concurrently). In case of single transmission, the transmitting tag recognizes its RN16 and sends back its unique EPC ID. The slot then terminates with the reader acknowledging the received ID. If instead multiple tags replied together, the reader sends back a RN16 that is the sum of the different tags sequences. As no tag recognizes its RN16, the slot terminates without the transmission of any EPC ID. Each message includes a cyclic redundancy checks (CRC) code to enable error detection. Specifically, the RN16 includes a 5-bit CRC, while the EPC ID includes a 16-bit CRC.

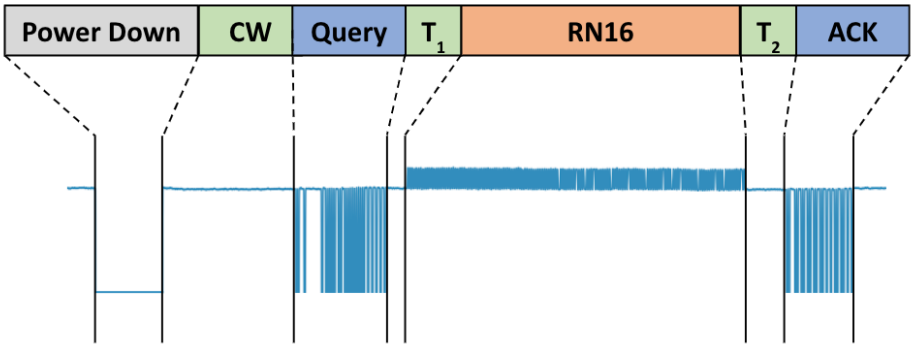


Figure 3.5. Reader-tag communication during the sampling process.

In this work, we exploit the randomness of RN16 to sample each tag over multiple different transmitted messages. Figure 3.5 shows the structure of a communication between our reader and a EPC-Gen2 tag during the sampling process. The interaction begins with the reader emitting a CW to energize the tag and allow it to receive and act on reader commands. After receiving this message, the tag takes time T_1 to react and reply by sending a RN16. The reader after receiving the response needs a reaction time T_2 before being able to acknowledge the tag by issuing an ACK. The interaction repeats multiple times — depending on the number of samplings per tags — with the reader energizing again the tag and issuing a new query. Reaction time depends on device characteristics and measures the reaction time from the end of a message reception to the start of a message transmission. On the tag side, it is estimated as $T_1 = 10/R$, while on the reader side it is estimated as $T_2 = 1/R$, where R is the datarate (according to EPC specifications, see [44], p. 43)). The sampling process relies only on RN16 collection. This is not without a reason. Sampling a tag

over a sequence that changes at each transmission allows the model to effectively learn hardware specific features. Instead, sampling a tag that transmitted always the same sequence (i.e., EPC ID) would make the model learn features related to a software defined ID.

3.5.2 The RFID16-2021 Dataset

We performed a massive waveform collection campaign with an experimental setup composed of an Ettus USRP 4 and two Flex 900 daughterboards, as depicted in figure 3.6. We used 200 AZ 9662 Alien H3 73.5x21.2mm UHF tags [143], depicted in figure 3.6. These tags are produced by YaronTech, widely adopted and easily purchasable over the web. Tags have been interrogated with the EPC-Gen2 standard [44], through a USRP implementation [23]. The antennas used in our setup are 902-928 MHz RFID antennas produced by Laird Connectivity, model S9028PCL [64].

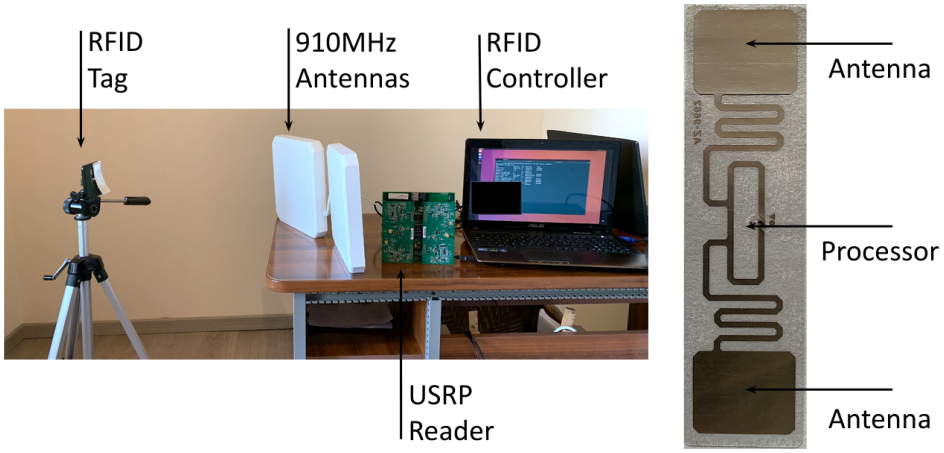


Figure 3.6. (left) Experimental testbed; (right) RFID tag.

The RFID16-2021 dataset contains RN16 exchange signals collected in 7 different scenarios. Each scenario, encoded as *SCEN-dist-obst*, is defined by 2 features: (i) the tag-reader distance (encoded by *dist*) and (ii) the eventual obstacle present between a tag and the reader (encoded by *obst*). We performed data collection at three distances, 20cm, 50cm and 100cm, relatively encoded as *020, 050, 100*. We defined three different scenarios: *Over-The-Air (OTA)*, which represents a data collection without obstacles between the reader and a tag, *Porcine Meat 0 (PM0)* which represents a data collection with a fat porcine meat 0.5cm thick obstacle between the reader and the tag, as depicted in figure 3.7, and *PM1* in which we use a low-fat meat of 3cm thick as obstacle. As an example, SCEN-050-PM1 represents a data collection at a distance of 50 centimeters between the reader and the tag an a low fat porcine meat as obstacle. The PM0 and PM1 subdatasets have 20 tags only, while OTA subdatasets have up to 200 tags each.

We used porcine meat as propagation media because it has similar properties to human muscular tissues, so as to simulate the presence of an under skin sensor (PM0) and a more deeply implanted one (PM1), as the PM1 porcine meat of 3 cm

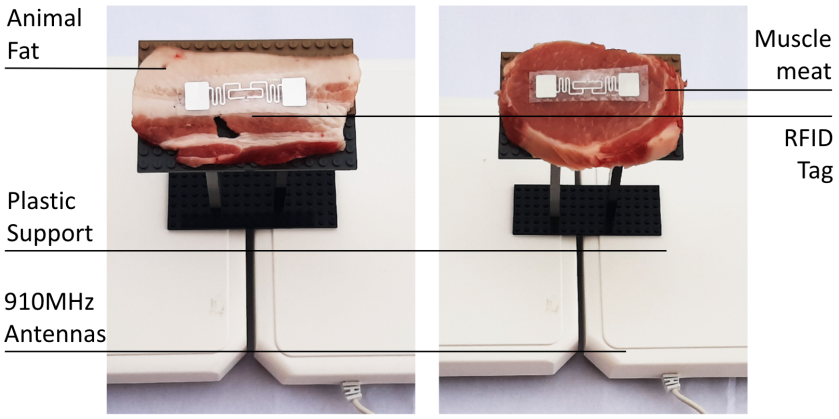


Figure 3.7. Data collection with animal fat(left), and animal muscle meat(right).

of thick simulates the presence of a muscle. The tags are the same throughout the whole data collection, meaning that the tag with label 0 is the same hardware in all the subsets. For our results we split every subset in the ratio $0.8, 0.1, 0.1$ for train, validation, and test. The data collection process has been performed over a 2-week periods for the subsets $OTA20, OTA50, OTA100$, where the first 100 tags have been collected through the first week and the other 100 during the successive one. The data collection process have been done in a normal office, and data are thus affected by uncontrolled and real interference. The RFID16-2021 dataset is publicly available [71].

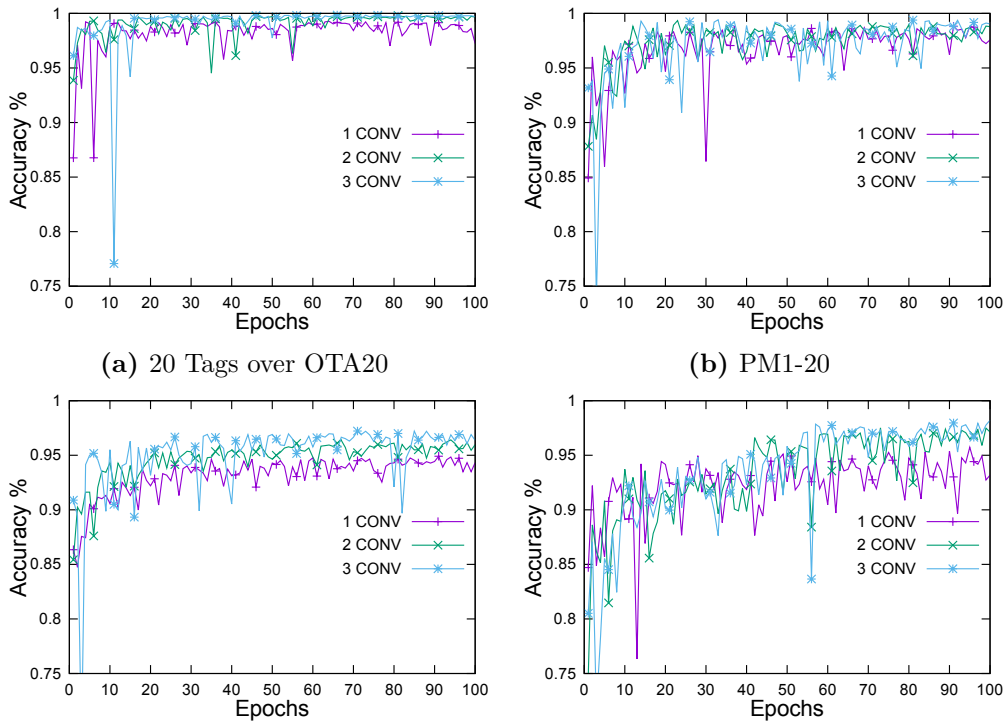
3.6 Experimental Results

In this section, we present extensive experimental results obtained from the RFID16-2021 dataset. To the best of our knowledge, this is the first publicly available dataset for RFP with RFID samples. This makes these experimental results completely reproducible, and a benchmark which today is not available. We recall that the RFID16-2021 is composed of 7 subdatasets, $OTA20$, $OTA50$, $OTA100$, $PM0-20$, $PM0-50$, $PM1-20$, $PM1-50$, as described in Section 3.5.2, and that the tags are the same through the whole data collection, meaning that the tag with label 0 is the same hardware in all the subsets. We split each subdataset following the 0.8/0.1/0.1 ratio for training, validation and testing, respectively. In the following, we will refer to the term "accuracy" to indicate the accuracy computed on the test set.

3.6.1 Hyperparameter Evaluation

We first investigate the impact of the number of CONV layers in the CNN described in Section 3.4.1 on the accuracy. Figure 3.8 shows the accuracy by changing the number of convolutional boxes with different scenarios. We made a test choosing the first twenty tags from OTA20. The model with 1 convolution has a lower accuracy compared to the other two, but with an accuracy drop of less then 2%, making the results of the three models comparable. We tried again with twenty tags but with

porcine meat as obstacle, using the set PM1 at 20 centimeters of distance between the tag and the reader, and still the results of the three models are comparable, even if with a general accuracy drop of few points compared to the first test we presented.



(a) 20 Tags over OTA20

(b) PM1-20

(c) 200 Tags over OTA20

(d) PM1-50

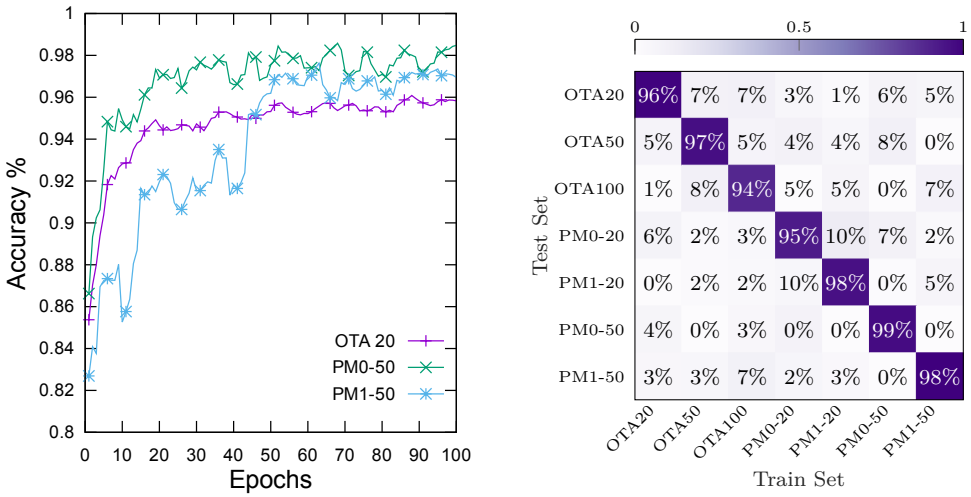
Figure 3.8. Results varying the number of convolutions, on OTA20 and PM1

With 200 tags and OTA20, we notice a general drop of few points which is interestingly more accentuated in the model with 1 convolution. Models with 2 and 3 convolutions are instead comparable. The most complex case for the neural network is represented by the scenario with porcine meat at 50 cm, PM1-50, where the accuracy of the 1 convolution model keeps dropping. The accuracy of the models with 2 and 3 CONV layers is instead stable an comparable, and we thus decided to utilize the 2-CONV model.

3.6.2 Impact of the Wireless Channel

Our next investigation is aimed at understanding the impact of the wireless channel on the classifier's performance. Therefore, we trained the neural network described in Section 3.4.1 on all 7 subdatasets, and tested it only on the test part of the same subset S, e.g., we trained the model on the train part of $OTA20$ and tested it on the test part of $OTA20$. As shown in Figure 3.9a, the neural network reaches an accuracy of up to 96% in free air communication with 200 tags and 98% in porcine meat with 20 tags. Interestingly, the neural network does not seem to be affected by the presence of obstacles like the porcine meat. The test on PM1-50 reports

an accuracy of 98%, higher than the one achieved performing the same test with the tags and reader closer and without the porcine meat obstacle. Increasing the number of tags (200) the accuracy lowers (95%) with respect to the other scenarios.



(a) Test accuracy on three subsets of RFID16-2021.

(b) Cross accuracy testing between differen subsets.

Figure 3.9. Impact of the wireless channel.

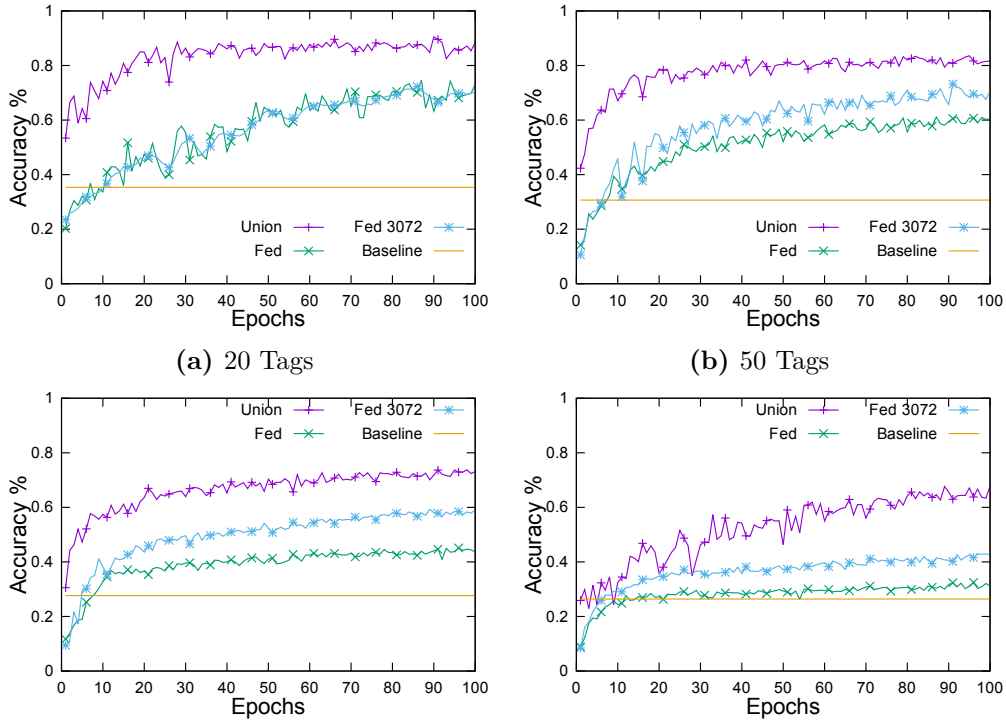
These results confirm findings in prior literature, where the radio fingerprinting is performed without specifying the channel condition. However, a simple cross-test shown in Figure 3.9b enlightens the limits of previous approaches, as a model trained on a subdataset X is not able to generalize and recognize the same tag with a model trained on another sub-dataset. Indeed, Figure 3.9b shows that whenever training and testing are performed over different sets - which is the case of every element that is not on the diagonal - the accuracy is very low (color intensity in the confusion matrix shows accuracy), down to 0% in some cases. **These results allow us to conclude that an RFID tag can be classified correctly if both the training and testing data have been collected with the same channel conditions. When a model is trained on a channel condition and tested, with the same tag, in another channel condition, we have a complete disruption of the results.**

3.6.3 Federated Radio Finger Printing Evaluation

The key aspect of FML is that a set of clients, with different data, can work together to train a shared model, without sharing the data directly. We made experiments without and with obstacles. In the first case we defined three clients, each one representing a different reader, which communicate with the same tag but at different distances. We created this scenario assigning the subset OTA20 to the first client, OTA50 to the second and OTA100 to the third. Each client trains a local model and, every epoch, sends the model weights to the central server, as shown in figure 3.3. This central server produces a federated average of the weights it receives, and shares back the federated weights with the clients. For the federated learning

we adopted the same neural network described in the preceding section 3.4.1.

To evaluate the accuracy of the federated model *we defined both an optimum scenario and a lower bound.* In the optimum scenario, which we call Union, we have a unique client which can access all three datasets, making training and testing on the union of the datasets. For the lower bound, which we call Baseline, we have n clients, each one with a model m_n and a dataset d_n. The Baseline is equal to the average of the accuracy of all the m_n models tested on all the d_n datasets.



(a) 20 Tags **(b)** 50 Tags

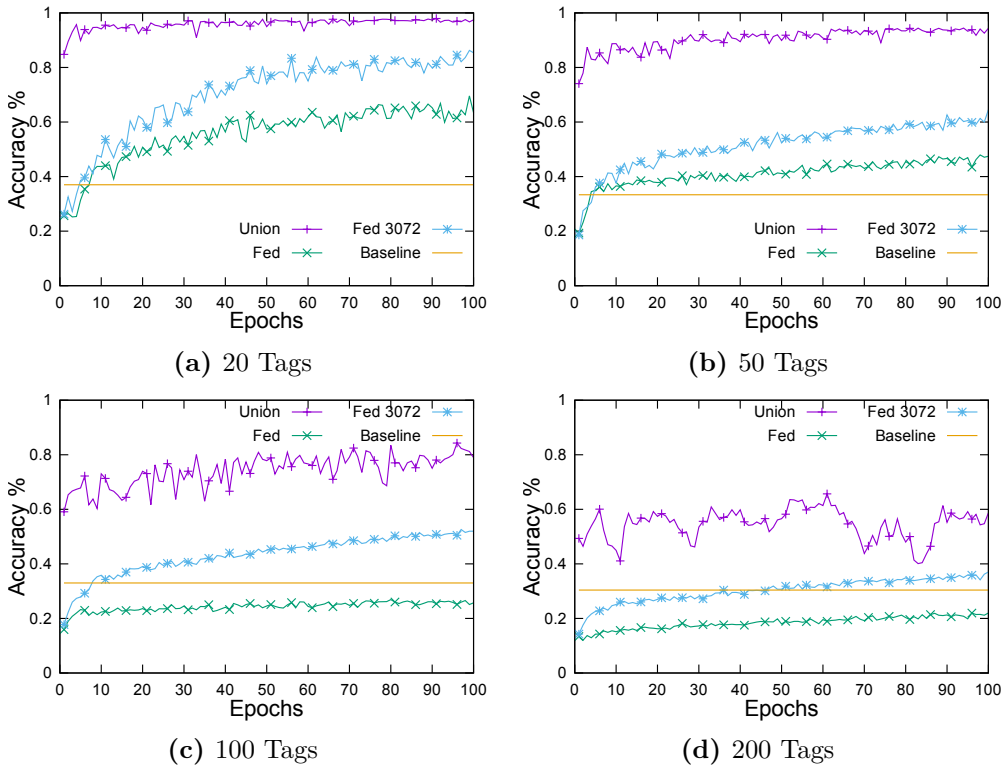
(c) 100 Tags **(d)** 200 Tags

Figure 3.10. Accuracy of Federated RFP with 10 percent of data.

Figure 3.10 and 3.11 depict the performance of federated RFP with a portion (10%) and the total (100%) of the dataset. This was done to evaluate its performance with fewer data available. As depicted in Figure 3.10, Union is able to generalize and recognize the tags at different distances, with an accuracy that is always at least two times better than the lower bound. This demonstrates how sharing the entire dataset produces a benefit on the accuracy. The results show that the federated RFP approach, with only the 10% of the data available, improves more than 10 times with respect to the Baseline. With 20 tags, federated RFP is less then 10% from Union. This distance however increases with the number of tags. However, the increase in the number of tags produces a drop of the accuracy even on the optimum, which highlights a limit on the data available more than on the federated approach.

For this reason, we also investigated the size of the learning windows, increasing it from 1024 to 3072 samples. With this change, the model is able to find more impairments per input, and thus we obtain advantages mainly in the scenarios with more nodes and more data. This suggests that more I/Q samples are needed to distinguish signals when the population is higher, which is consistent with prior art.

In scenarios with fewer tags, like depicted in fig.3.10, it is instead easier to find unique tags features and the window size increase does not produce benefits. However, this approach still seems to be not robust, as with 200 tags and 100% data it reaches up to 40% of accuracy. This drop may still be related to the model learning channel features and interference. **It is thus clear that the federated RFP approach, with only the 10% of the data available, produces a huge improvement on results, obtaining an accuracy more than 10 times higher compared to the Baseline. On the other hand, this approach does not appear to be robust to node population.**



(a) 20 Tags **(b)** 50 Tags

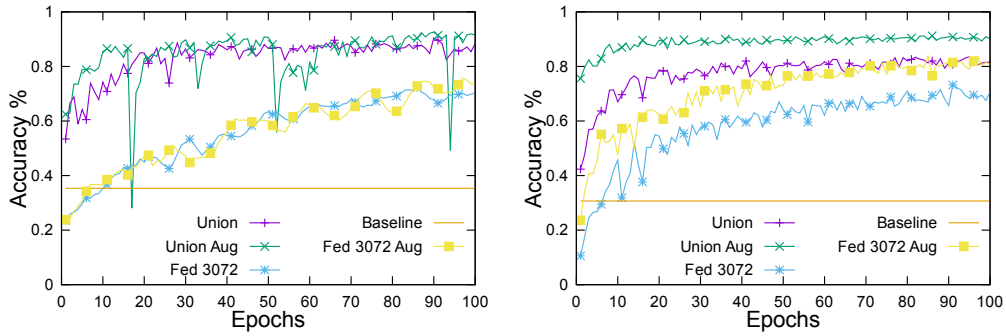
(c) 100 Tags **(d)** 200 Tags

Figure 3.11. Accuracy of Federated RFP with 100 percent of data.

3.6.4 Data Augmentation

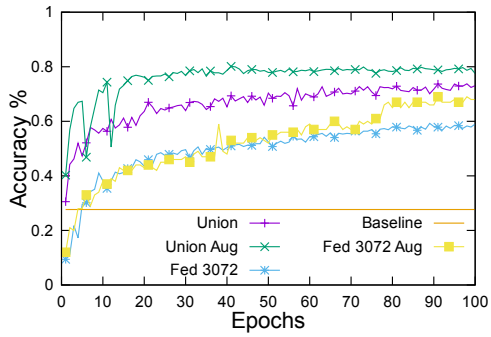
We now evaluate the data-augmented RFP technique described in Section 3.4.3. Figure 3.12 shows the results with 10% of the data obtained on the OTA subdatasets. As done earlier, we report the results obtained with Union and the Baseline. We can see the advantages of this approach, especially when the population size increases. **These results show that with only 10% of the dataset, the accuracy of federated RFP is increased by an impressive 20% with 200 tags, becoming comparable with the accuracy of the Union dataset.** This improvement makes the accuracy obtained by a federated model comparable with the accuracy of the optimum model, confirming the capabilities of the federated approach to deal with noisy and variable channel environments. This impressive effectiveness of data augmentation is also confirmed with the optimum model. Applying data

augmentation on that model shows an improvement in the accuracy of up to 10%, as shown with 50 tags. This improvement pattern may be related to the extensive number of different channels recorded with many tags. We recall that the tags collection has been performed over 2 weeks, and that the first tags may have a completely different set of interference as the data collection has been performed in a normal office. Thus, data augmentation seems to be particularly efficient in realistic scenarios, as it is able to deal with temporary and different interference.

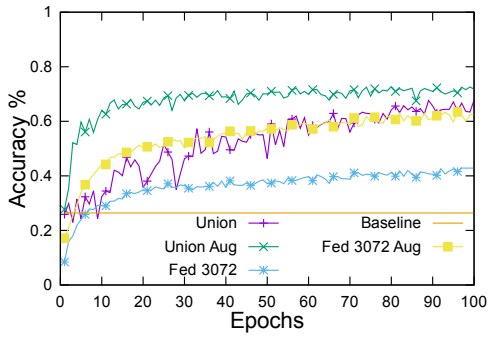


(a) 20 Tags

(b) 50 Tags



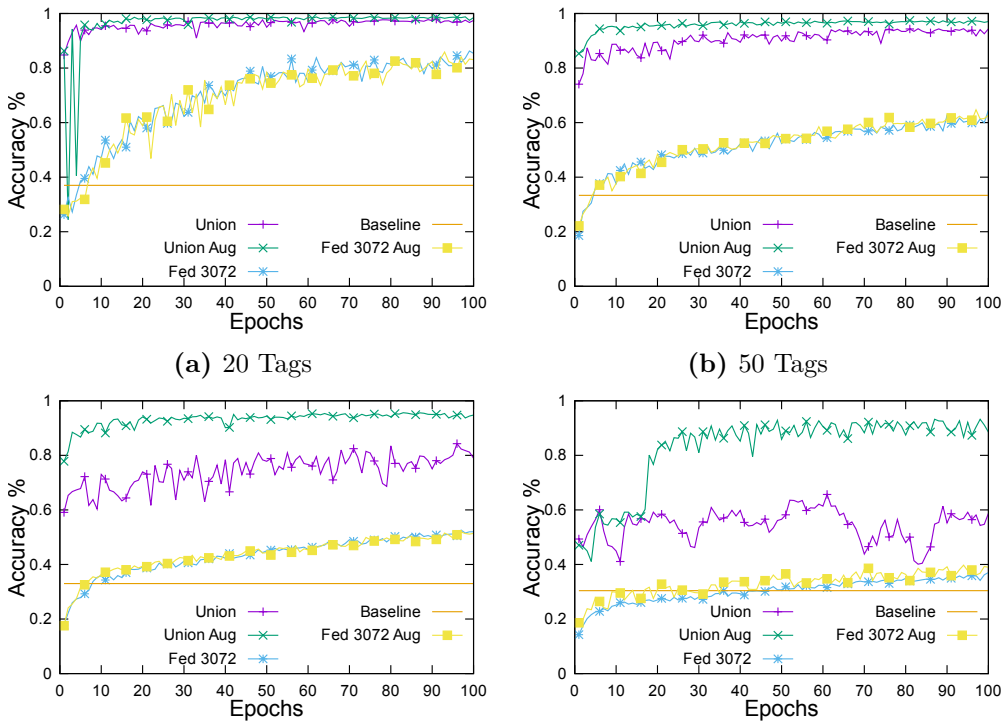
(c) 100 Tags



(d) 200 Tags

Figure 3.12. Accuracy of Data-Augmented RFP with 10 percent of data.

Figure 3.13 shows the results with 100% of the OTA subdatasets. First, the results indicate that with more data, the accuracy decreases in all the federated RFP scenarios considered. This effect is particularly observed as the population size increases, where the accuracy drops by 20% in the case of 200-tag when going from 10% to 100% of the data. While this result may seem counter intuitive and in contrast with prior art on ML, it is rooted in the principle that *more data does not necessarily mean a better model, if it causes more confusion in the classifier*. Indeed, we see that also in the Union dataset, the learning curves show erratic trends, especially in the 100- and 200-tag datasets. This tells us that by adding more data, we are introducing more confusion into the models, which try to "overfit the different channels" without improving the generalization power. This effect is felt stronger in federated RFP, which trains on local datasets only and therefore it takes much more time to converge than with less data. **Nevertheless, we notice that DAG on the Union dataset returns impressive results, with an accuracy improvement that increases with the population size, and reaching a 38%**



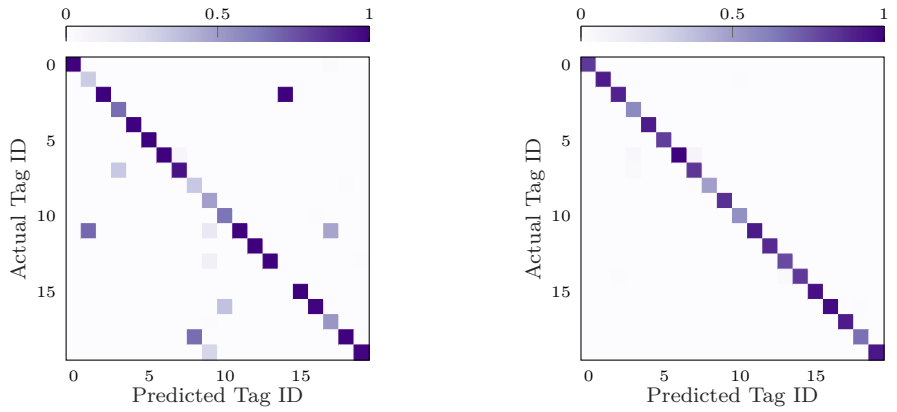
(a) 20 Tags **(b)** 50 Tags

(c) 100 Tags **(d)** 200 Tags

Figure 3.13. Accuracy of Data-Augmentated RFP with 100 percent of data.

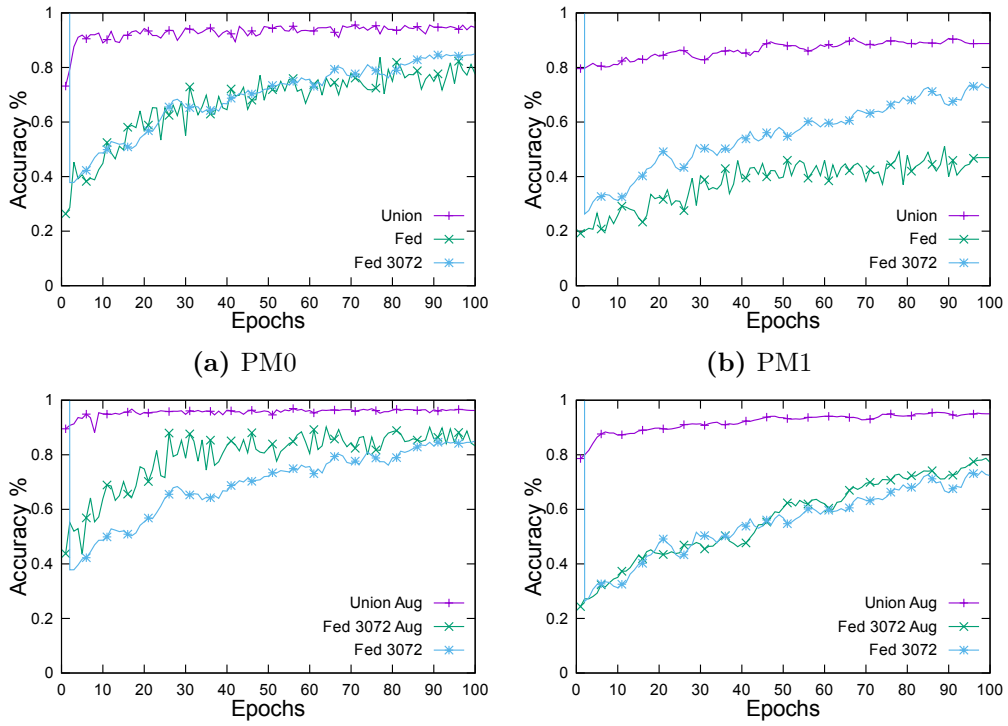
improvement in the case of 200 tags. This is also represented in Figure 3.14, where we depict a subportion of the 200-device confusion matrix for the bottom (i.e., lowest accuracy) 20 devices in the dataset, with and without data augmentation.

Finally, in Figure 3.15a we report the results obtained by applying DAG on the PM0 and PM1 datasets. With PM0,1 we mean a federated environment where one client uses PM0,1-20 as dataset and the other one uses PM0,1-50. Even in this case, we verify that federated RFP is able to generalize by reaching an accuracy of 83% in the PM0 datasets, within 10% of the Union accuracy. While with PM0 the federated approach obtains acceptable results with a smaller input size of 1024 I/Q samples, PM1 needs to increase the window size to 3072 to obtain results close to Union. The reason is that the PM1 were collected with much thicker porcine meat, which appears to introduce more distortion to the received waveform and thus partially hide the impairments, thus requiring more I/Q samples to achieve an accuracy similar to PM0. Figure 3.16 provides a visual indication of how impactful DAG is on the PM1 dataset. Finally, in line with the OTA results, we observe that DAG does not benefit federated RFP when the population size is low (PM0 and PM1 have only 20 tags). **Consistently with the OTA case, these results confirm that DAG achieves better performance in more complex scenarios such as PM1, where the Union accuracy goes from 87% to 97%.**



(a) Without Data Augmentation **(b)** Without Augmentation

Figure 3.14. Bottom 20 devices of the OTA20, OTA50 and OTA100 union with 200 tags and 100% of data



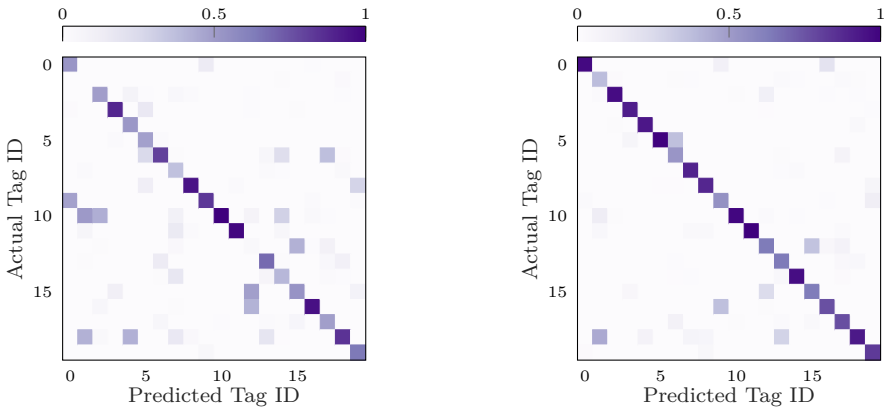
(a) PM0 **(b)** PM1

(c) PM0 with Data Augmentation **(d)** PM1 with Data Augmentation

Figure 3.15. Results with porcine meat with 100% of data.

3.7 Conclusions

In this chapter we demonstrated the first large-scale investigation into RFP of RFID tags with dynamic channel conditions. This investigation is composed of three main contributions: first, a massive data collection campaign on a testbed composed of 200 off-the-shelf identical RFID tags and a SDR tag reader, even with porcine meat



(a) PM1 Union without data augmentation **(b)** PM1 Union With data Augmentation

Figure 3.16. Porcine Meat

as communication obstacle to emulate an implanted RFID. Second, the definition and the relative evaluation on the dataset of several CNN-based classifiers in a variety of channel conditions. Our investigation reveals that training and testing on different channel conditions drastically degrades the classifier's accuracy. Third, a novel training framework based on FML and DAG able to boost the accuracy of the model. Extensive experimental results indicate that (i) our FML approach improves accuracy by up to 48% with respect to the single-dataset scenario; and (ii) our DAG approach improves the accuracy by up to 31%. Results also highlight how FML and DAG should be applied to tackle different problems, and have different results depending on the scenario and the number of tags. With this work we conclude our steps through the physical layer, and we start working on the coexistence of multiple battery free RFID devices. At the MAC layer, today there are no available protocols able to allow an efficient communication for a set of RFID devices.

Chapter 4

Adaptive Communication for Battery-Free Devices in Smart Homes

4.1 Chapter Introduction

While there has been substantial work on the underlying methods for RF-powered computing, practical applications of this technology has largely been limited to scenarios that involve simple tasks.

In this chapter we want to demonstrate how RFID technology, typically used to implement object identification and counting, can be exploited to realize a battery-free Smart Home. In particular, we consider the coexistence of several battery-free devices, with different transmission requirements — periodic, event based, and real-time — and propose a new adaptive and quick-to-learn MAC protocol, called APT-MAC, which dynamically collects information from devices without requiring any apriori knowledge of the environment. Extensive simulations clearly show the benefits of using APT-MAC, which is able to successfully deliver 97.7% of new data samples in complex scenarios, including several high traffic demanding devices such as joysticks and cameras.

The challenges in building battery free smart objects by exploiting RFID backscattering are multiple.

1. How to make battery-free devices to sense and transmit given that they cannot operate spontaneously on their own energy;

2. How to guarantee enough energy for multiple battery free devices at the same time;

3. How to simultaneously support heterogeneous sensor types, sensor requirements, and their different uses; and

4. How to effectively cover an entire home.

In this chapter we explore the design of a Battery-Free Smart Home from a communication point of view and make the following contributions:

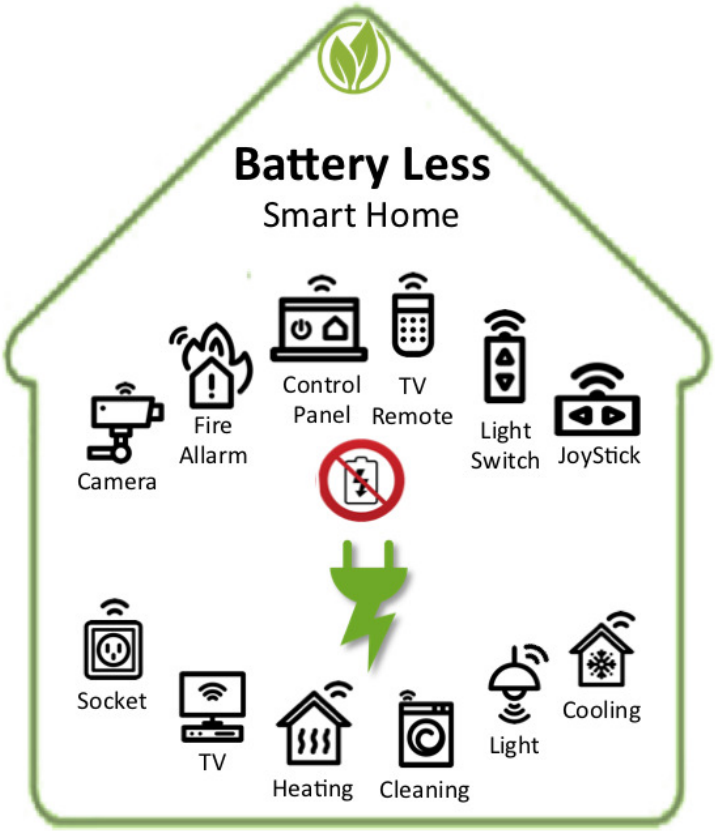


Figure 4.1. Battery free smart home.

1. Invented a new MAC protocol to collect information from smart devices that improves response time and data delivery. Our protocol, called APT-MAC, quickly learns transmission rate requirements of active devices, without having any a-priori knowledge on the type of devices, and adapt information collection to such requirements. This is critically important since smart homes are outfitted with many heterogeneous devices and people will not want to reconfigure their smart home every time they add or delete devices or applications that use these devices.

2. Showed through simulations that our MAC protocol is able to self adapt to devices requirements, reporting very low packet delay — $33ms$ in a scenario including up to 40 devices — and minimum data loss — below 5% for each type of device.

3. Studied what is necessary to cover an entire smart home.

4. Addressed energy consumption and health issues.

4.2 Battery-Free Smart Home: Architecture

Smart Homes are outfitted with a myriad of sensors and smart devices | cameras, presence sensors, smoke sensors, light sensors, thermostats, smart meters, etc. | that are used to reduce resource consumption and improve the quality of life. In a home there are also a variety of everyday devices, such as TV remotes, cooling system remotes, light switches, and video game controllers, that are not "smart", but are easily used by a vast majority of people to control many home devices/appliances. In this work, we consider both classes of devices and make them operate without electric cords or batteries in order to realize a battery free Smart Home (see Fig. 4.1).

To achieve this goal we exploit RFID technology, which is considered a key technology for identification of smart objects and hence it is deployed in any smart environment [85]. In RFID, battery free devices — the tags — send their unique ID to a powered device — the reader — by reflecting the high-power constant signal generated by it. In our system, the reader interacts with sensor-augmented RFID tags, such as the UMich Moo Computational RFID tag [147]. These tags are equipped with on-board sensors and/or actuators to provide not only static information such as their ID, but also dynamic and real-time information about the state of the tagged object or the environment where these objects reside. The energy necessary for sensing and transmission is harvested from the reader's transmitted signal.

At the hardware level the system architecture includes several battery-free smart devices and a RFID reader, equipped with multiple antennas. Battery-free devices that can be realized through sensor augmented RFID tags are: videogame controllers [76], cameras [86], presence detectors, light switches, remotes for appliances, temperature sensors. They are scattered in different rooms of the home. The RFID reader is equipped with multiple antennas, a transmitting and a receiving antenna for each room. Each antenna has a transmission range that is able to reach all devices in the same room. When the reader issues a query, all the transmitting antennas broadcast the query at the same time, reaching all the devices in the home. When tags receive a query, only the queried tag (whose ID is indicated inside the query message) backscatters the received signal to send its sensed data to the reader; all the other tags use the received signal to power on-board sensors and store the sensed data in a local buffer. At each query only one device in the home is interrogated. Multiple receiving antennas may receive its response (e.g., antennas in nearby rooms). However, the energy emitted by tags naturally decreases with distance; for example the rate of decrease is typically proportional to the inverse fourth power of the distance between the tag and antenna. Therefore, the closest antenna will receive the best signal, while the others will receive a weak signal. The reader will decode only the best signal and discard all the others. In case the reader receives multiple decodable signals, it can decode all of them, getting redundant information. Figure 4.2 depicts an example of our system architecture, showing for clarity only two temperature sensors and a smart-phone running the air-conditioning app. The reader is connected to a local server that receives sensed data from the reader and dispatches the data to recipient applications. The server can interact with Smart Home applications running on several devices. For example, in a videogame application the server receives the joystick data from the reader and produces game

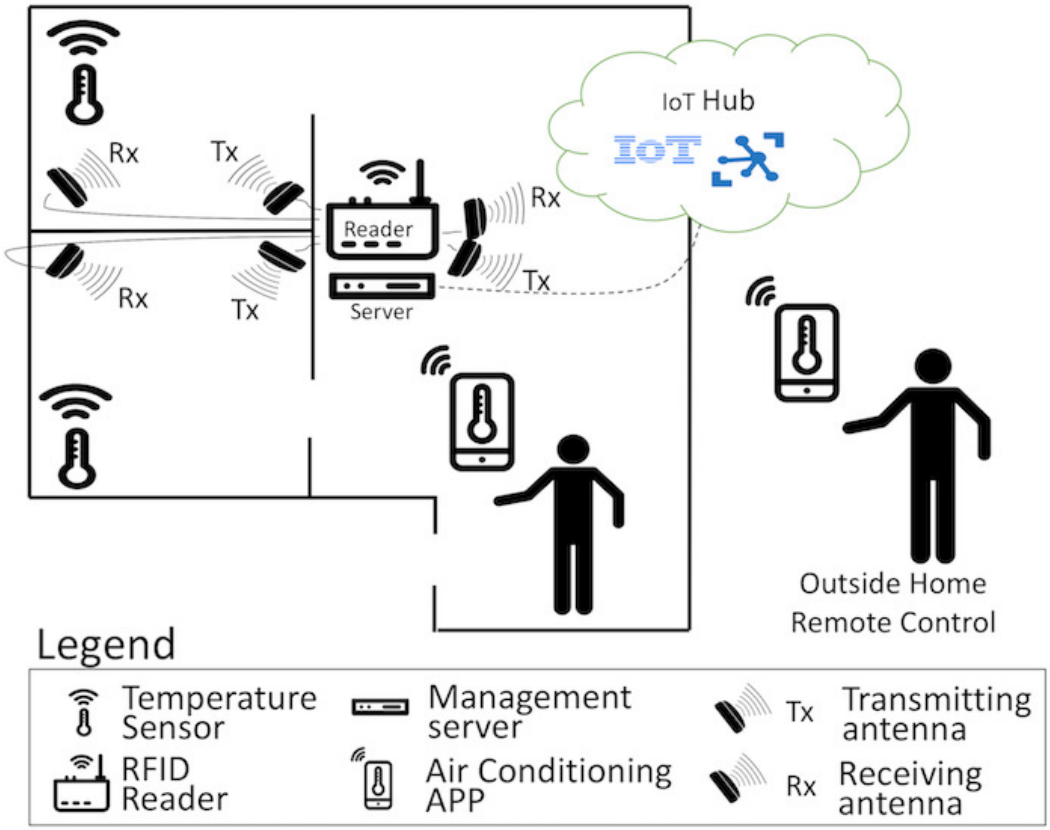


Figure 4.2. Smart Home architecture: the system includes a RFID reader, equipped with a transmitting (T_x) and a receiving (R_x) antenna for each room. Temperature sensors are deployed in different rooms and send data to the air-conditioning app — running on a smart phone — through the RFID reader that is connected to a server. An IoT hub allows for interaction with home devices from both inside and outside the home.

commands that are sent to the videogame console. In an air-conditioning application the server can send commands to the air-conditioning unit depending on the received temperature data, or can show sensed data on a local screen. The server can also interact with client apps running on smart phones, allowing for interaction from both inside or outside the home — through an Internet IoT hub.

4.3 Advantages and Limitations of a RFID based Approach

4.3.1 Advantages

There are important advantages to make smart homes battery free.

E-waste reduction - the majority of batteries currently used in houses and offices are eliminated, significantly reducing electronic waste. The study in [117] reports that China produced 570'000 tons of batteries only in 2013; and this value is growing year by year. The reduction of such an amount of waste is imperative, as the

toxic metals required to make rechargeable batteries are harmful to the environment.

Energy consumption reduction - decreasing the need for batteries implies diminishing also energy consumption for their recharging. It is true that our system requires a continuously operating RFID reader — which consumes energy (see Sec. 4.8 for consumption estimation) — but any smart environment involves the deployment of a RFID reader, as RFID is considered a key enabling technology for many IoT applications [85]. The article in [107] remarks on this aspect by stating that RFID technology should be deployed as part of a building's physical infrastructure, just like running water, lights and heat. Exploiting such readers, our proposed devices can work with the emitted energy, without requiring batteries or other sources of energy.

Time saving - There is no need for batteries charging and/or replacing, for set up or configuration. Devices are always ready to work.

Increased device portability - most of devices become highly portable as electric cords or wires are removed. Consider the case of a light switch, realized through a tag equipped with a press button. It can be moved wall to wall without the need for cables and without batteries, with an almost unlimited lifetime. It can send messages to smart connected bulbs, in order to switch on and off. The smart bulb needs only a power socket, but it does not need to be directly connected with the switch as in current homes; power implants are much simplified, as we just have to bring power without paying attention to circuits.

No need for direct visibility - most battery free devices can trigger actions without being in the same room as the actuators. For example the light switch can turn on a light that is located in another room. Similarly, any remote for appliances can power on devices that are located anywhere inside the home (under the reader coverage).

Increased environmental sustainability - device lifetime becomes unlimited: there is no more need for battery recharging or replacement.

4.3.2 Limitations

The main disadvantages are related to RFID technology and its operational limits.

Limited range - antennas coverage ranges between $1m$ and $3m$ depending on transmission power (around $0.5 - 1W$ for current prototypes), significantly limiting the operational range of devices. However, in sec. 4.7 we discuss how to improve this weakness.

No outdoor operation - battery-free devices cannot operate outdoors (around the house), or inside where the reader's antennas are not deployed. Although this represents a significant disadvantage, there are positive implications in terms of privacy as none of these devices can be accessed (queried) from outside.

4.4 A Zero Configuration MAC Protocol

4.4.1 Protocol Description

A common vision for the IoT foresees a large number of devices in smart homes in the near future [119]. The major issues in dense sensing and actuation environments are

system configuration, reconfiguration, and management, as sensors and applications are added and deleted. However, most people want to use systems without configuring them: manual configuration is tedious and error-prone. Thus, the need for automatic and configuration/management free protocols. We propose a MAC protocol, called APT-MAC, which quickly learns transmission rate requirements of active devices, without having any a-priori knowledge of the type of devices. We use reinforcement learning to build and continuously update device transmission behavior. Then APT-MAC adapts information collection (and hence channel access) to such requirements. This avoids configuration overhead.

Algorithm 1 APT-MAC: pseudocode for slot assignment

```
1: Master M;                                          ▷ The reader
2: Set D;                                             ▷ All devices
3: Map R:(d∈D)→double;                                ▷ Rewards Map
4: /* Initialization of the Rewards map */
5: for d ∈ D do
6:     R[d] = 1.0;
7: end for
8: R = softmax(R);
9: /* Each cycle corresponds to a time slot */
10: while true do
11:     Device next = chooseNext(R);
12:     Bool goodQuery = M.query(next);
13:     if goodQuery then
14:         R[next]=updateReward(next,bonus);
15:     else
16:         R[next]=updateReward(next,malus);
17:     end if
18:     R = softmax(R);
19: end while
```

We now describe the sequence of stages involved in the APT-MAC protocol. At setup, the system sends inventory queries to *discover* how many *devices* — sensor-augmented RFID tags — are in the house, and assigns each a unique identifier. At this point the reader knows all devices in the environment and can *query* them to *collect data* sensed by their on-board sensors.

The key problem is: In which order should the reader query tags? A TDMA approach with fixed slot assignment queries tags sequentially without giving priority to most demanding devices; it does not scale. Our APT-MAC protocol respects device transmission needs and properly rules channel access. We employ reinforcement learning to learn sensor behavior and requirements: it discovers environment patterns, changes in those patterns, and required rates for reading. As an example the protocol is able to understand when a video-game controller is started, and to allow it to send data more frequently than periodic-based devices (e.g., environmental sensors or information displays). The main goal of the protocol is to query tags so as to minimize the time between the generation of new sensor data and its delivery to the reader. To achieve this goal the reader implements the Multi-Arm Bandit

algorithm [124], a reinforcement learning algorithm, based on the action-reaction paradigm. This algorithm involves 5 components: 1) the agent that performs the actions on the environment, i.e., the reader; 2) the set $A = a_i, i = 1, ..., n$, (n is the number of sensors) of actions the agent can perform, e.g., query tag i, query tag j, etc.,; 3) the set S of states in which the agent can be (in our case the only state is "ready to perform a new query"); 4) the formula $Q(a_i)$ to evaluate the expected reward of action a_i and 5) The vector Q to store the expected reward of each action.

Time is slotted and each slot, also called epoch, involves a reader's action, (to query a tag). The pseudocode for slot assignment is given in Algorithm 1. The tag with the highest expected reward is selected to be queried next (*chooseNext* method). Let us suppose the reader is starting epoch $n+1$ by querying tag $next = i$ and thus taking action a_i. When the reader receives response from tag i, it updates the expected reward $Q(a_i)$ for the taken action by summing the expected reward $Q(a_i)(n)$ calculated at the previous query, and the difference between the current outcome and the expected reward at the previous query, as defined in eq. 4.1,

$$Q(a_i)(n+1) = Q(a_i)(n) + \alpha(Reward - Q(a_i)(n)) \tag{4.1}$$

where α is the learning rate and is fixed to 0.1 (we empirically found this as the best value). *Reward* is the outcome of the current query: it is a positive value if the queried tag sent fresh data ($R[next] = updateReward(next, bonus)$), and negative if the sensor mounted on the queried tag did not produce new data since the last query ($R[next] = updateReward(next, malus)$). Specifically, the reward is calculated as $Reward = bonus - malus$, where $bonus = 0.4$ and $malus = 0.01$. If the tag sent new data then $Reward = 0.39$. In case the reader queried a tag that did not have new data since the last query, the reward is negative, i.e., $Reward = -0.01$. After each reward update, we perform a Softmax [124] on the Q vector ($R = softmax(R)$). With Softmax we compress the Q vector's values into the range of [0,1]. All the values of Q after Softmax sum to 1.

A positive reward allows a tag to gain and keep channel access: a videogame controller (i.e., joystick) just started will send new data every time it is queried and thus will have a very high reward, accessing the channel very often. However, no sensor is so fast to generate new data every slot (experimental results [76] show that slots last $6ms$). Furthermore, tags need a query to read data from on-board sensors and record new values. As a consequence, a tag cannot be queried at each slot, and thus we fixed a Minimum Query Delay (MinQD), that is the minimum interval of time at which tags can be queried (MinQD is fixed at $50ms$ based on empirical study). To avoid any starvation problem, in which a tag is never queried because the others have always higher reward values, we also set a Maximum Query Delay (MaxQD), that is the interval of time at which tags have to be queried regardless the expected reward, meaning that each tag will not wait longer than this time between two consecutive queries.

MaxQD time guarantees fairness. While MinQD can be fixed to a common value for all devices, MaxQD depends on devices sampling rate requirements — a joystick or a camera needs to send data more frequently than an environmental sensor. Thus, MaxQD is dynamically set for each device during the start-up phase. When the system starts, MaxQD is fixed at $2000ms$ for each device. While the system is

operating, the reader updates the MaxQD based on the data loss it observes for each device. Data loss is calculated over intervals of 50 queries (per device) as the number of new data samples received by the reader over the number of new data samples produced by the device (this information is provided by the device that is able to keep a counter of changes, i.e., the number of new samples between two consecutive queries). If the number of sample changes is greater than the number of new data samples received by the reader, then the reader should query the tag more frequently. Specifically, we set a 15% the tolerable data loss. When the reader loses more than 15% of new samples it reduces the MaxQD of $150ms$. This process continues until the data loss is below 15% or the system reaches the MinQD (i.e., $50ms$).

To allow new tags to join the system and at the same time less demanding tags to be queried more often than once every the MaxQD, action selection is probabilistic: the reader queries the device with the highest expected reward with high probability, $1 - \epsilon$, and with a small probability, ϵ, it queries another randomly chosen device. The epsilon value is set to 0.1 based on an empirical tuning.

The APT-MAC protocol can be deployed on any RFID reader running EPC Global standard [44], by exploiting EPC primitives. The `inventory` command can be adopted to discover devices in the environment, the `read` command can be used to query devices, and the `write` command to send commands to actuator devices. The reader continuously sends commands to devices. In case of sensors' readings, commands are scheduled through the mechanism based on reinforcement learning. In the case of an actuator writing instead, the command is scheduled based on requirements coming from the application: the reader receives a request from the application and interleaves the write command between scheduled readings.

4.4.2 Computation Time

In terms of time, APT-MAC protocol costs $O(n)$, where n is the number of devices. At each slot, the reader chooses the next device to query by executing the *chooseNext* method that searches for the maximum value in R, analyzing in the worst case all n devices ($|R| = n$). Even the *softmax* operation can be executed in $O(n)$, as it goes by every element in R and performs a constant time operation for each (i.e., division), while the *updateReward* function can be executed in $O(1)$. Thus at each time slot, the time complexity of APT-MAC is $O(n)$. Compared to TDMA, the choice of the next device to query is slower in APT-MAC, as TDMA has a predefined slot assignment that has constant cost $O(1)$. However, APT-MAC's cost is negligible if we consider that in smart homes the number of devices is in the order of tens.

4.5 Performance Evaluation

We perform simulations to evaluate the performance of our APT-MAC protocol and compare it with a TDMA protocol that sequentially queries all tags, and the optimal query strategy, called *Optimum*, that always knows the best action to perform. At each epoch, the Optimum knows which sensor has produced new data, and queries exactly that device, performing optimal data collection.

Table 4.1. Workload Scenarios Description
(Env. sensors include temperature and presence sensors)

# of Sensors	Scenario	Joystick	Remote	Env. Sensors
20	Case 1	1	2	17
	Case 2	2	3	15
	Case 3	3	3	14
	Case 4	4	4	12
30	Case 1	1	2	27
	Case 2	2	3	25
	Case 3	3	3	24
	Case 4	4	4	22
40	Case 1	1	2	37
	Case 2	2	3	35
	Case 3	3	3	34
	Case 4	4	4	32

4.5.1 Scenarios

We considered a wide range of use of devices that should cover most homes: the number of battery-free devices ranges from 20 to 40 ($n = 20, 30, 40$), including different types of devices to realize different workloads. Specifically, for each number n of devices we simulated four different cases. Table 4.1 details the numbers of different types of devices included in each case. Environmental sensors include temperature and presence sensors, as their transmission rate requirement is similar and very low. Complexity increases from case 1 to case 4 and from 20 to 40 sensors, as the number of real-time devices increases.

4.5.2 Metrics

In our performance evaluation we focus on the following metrics: packet delay and data loss rate.

- **Packet delay**: the time between the generation of new sensor data and its delivery to the reader. In case a tag reads new data multiple times before being queried, the packet delay is measured since the first, undelivered, data collection.

- **Data Loss**: the amount of new data samples delivered to the reader over the amount of new data samples generated by the sensor. Thus data loss reflects how much new data is lost because tags are not queried on time. As tags have a small memory when they read new data from the on-board sensors they have to overwrite the previous data to store the new one. However, they can maintain a counter of changes that keeps the number of data updates performed since the last data transmission.

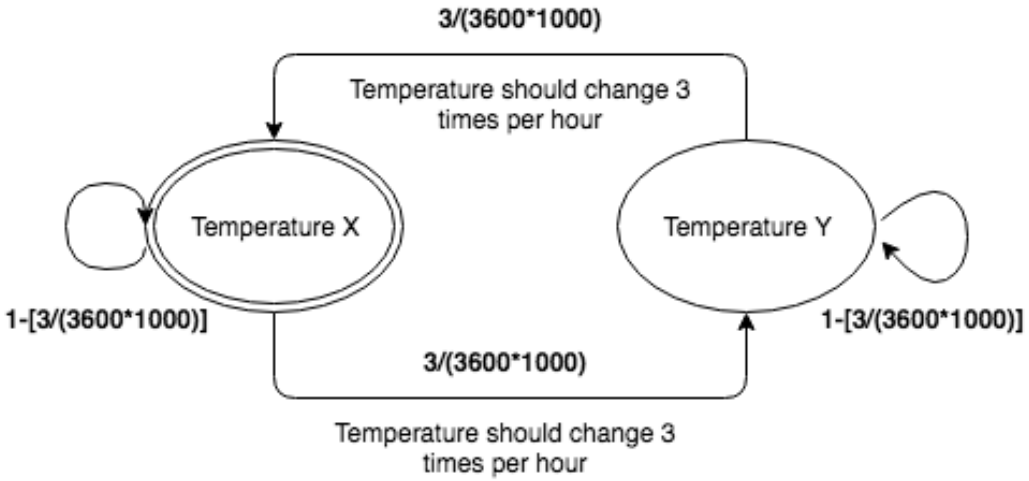


Figure 4.3. Temperature sensor model.

4.5.3 Device Model

Smart homes include mainly three types of devices: real-time (e.g., joysticks, cameras), periodic (e.g., temperature sensors) and event based (presence detectors, remotes for appliances). To build a realistic simulation environment we modeled devices behavior through Markov Chains. We watched real operation for a temperature sensor, a TV remote, and a joystick, and built device models based on the observed behavior. The goal of our model is to focus on state changes rather than on specific device states — values sampled by the on-board sensors. In the case of a temperature sensor, the model represents if the current value is different (it may be higher or lower) with respect to the previous reading, but does not keep track of the sampled value. Thus a temperature sensor model includes only two states: Temperature X and Temperature Y with a transition from a state to the other whenever a new temperature value is observed. To introduce randomness in device behavior, state transition is probabilistic and is calculated as the number of expected changes over time.

Each state is characterized by a *stand-by time*, intended as the minimum amount of time in which the state cannot change due to physical aspects and user's speed in performing actions. For example, we measured the user speed to press buttons on a device, and we found that the interval of time between two consecutive presses on the same button does not last less than $70ms$. This time increases to about $98ms$ in the case of two different buttons. Environmental devices, not sensing user actions, present also a stand-by time, due to physical sensor aspects (e.g., sensing resolution, refresh rate, ADC conversion, etc.). Considering these factors we fixed the stand-by time at $50ms$: if at time x the sensor has sensed value y then the sensor will report the value y for the following $50ms$ (up to time $x + 50ms$). This time motivates also the choice of fixing the MinQD at $50ms$ (see Sec. 4.4).

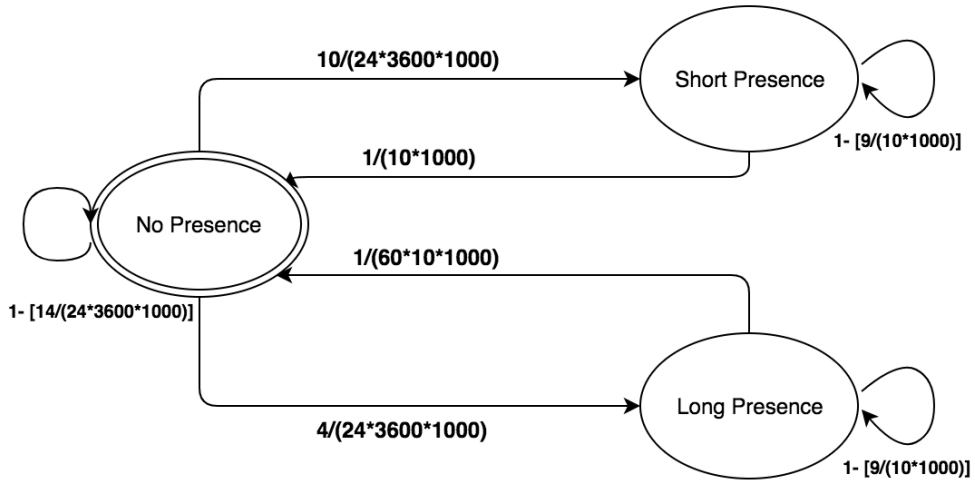


Figure 4.4. Presence sensor model.

Temperature Sensor The model for this sensor (see Fig. 4.3) has only two states — Temperature X and Temperature Y — and the transition probability between them is $3/(3600 * 1000)$, where 3 represents the number of times we expect the temperature to change, and $3600 * 1000$ represents the interval of time — one hour — in milliseconds in which we expect these 3 changes to happen (3600 is the number of seconds in an hour and 1000 is the number of milliseconds in a second). This model is confirmed by results we obtained measuring temperature with our sensor: on average temperature may change by 1 degree three times per hour, not only due to changes in daily climate, but also to temporarily opening of windows/doors.

Presence Sensor The model for this sensor (see Fig. 4.4) has 3 states: No Presence, Short Presence, Long Presence. From the No Presence state, we may transit in the Short Presence state 10 times a day $(10/(24 * 3600 * 1000))$, or in the Long Presence 4 times a day $(4/(24 * 3600 * 1000))$. As a presence sensor detects people movements, there is a permanence for each state, that we modeled as Short Presence state for a number of seconds less than or near to ten $(1/(10 * 3600))$ seconds, and as Long Presence for times near to 10 minutes $(1/(60 * 3600 * 1000))$.

TV Remote For the simulation of a TV remote we modeled 4 states: Not in Use, Configuration A, Configuration B and Not Pressing (see Fig. 4.5). From the Not in Use state we expect to start using the TV Remote 8 times per day. Remote buttons can be pressed with the same probability, and a button press makes the system transit to a different state. After pressing a button, it is possible to move only to the Not Pressing state, which is different from the Not in Use state, as it represents the situation in which the user is going to press another button. We expect the user to change state about every second, and to perform around 8 presses before stopping use of the TV Remote.

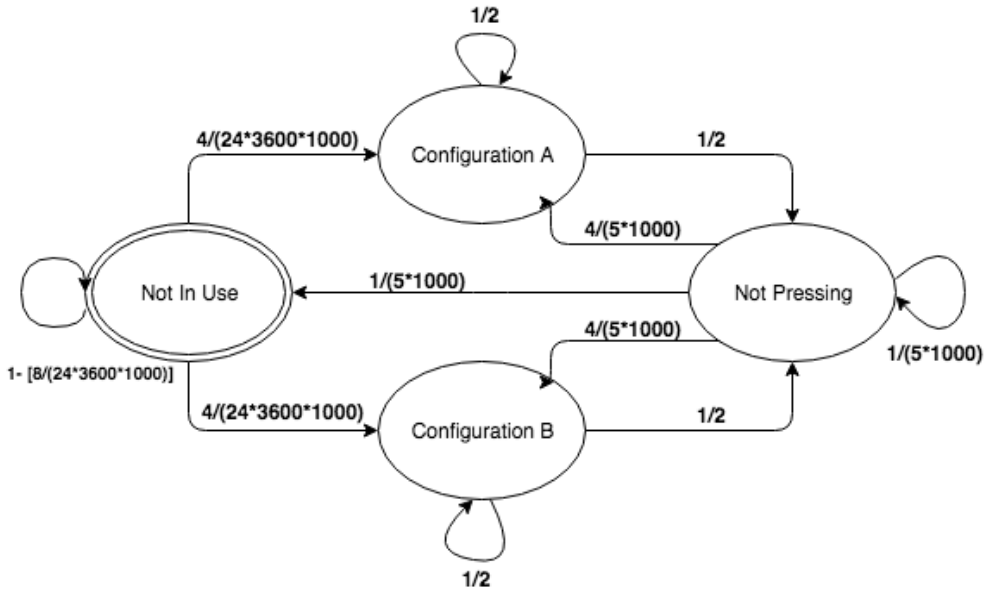


Figure 4.5. TV Remote model.

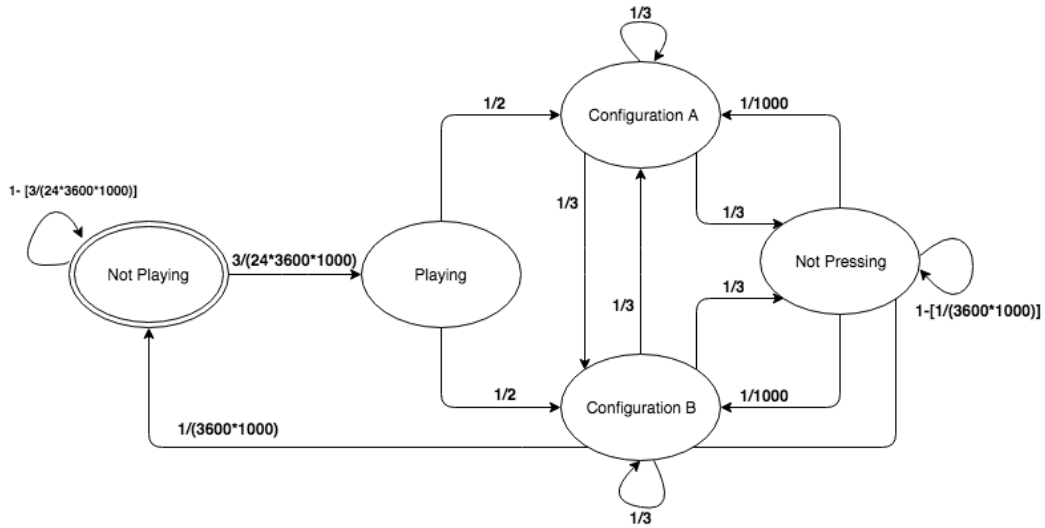


Figure 4.6. Joystick model.

Joystick The joystick represents the most complex device to model (see Fig. 4.6). It includes five states: Not Playing, Playing, Configuration A, Configuration B, Not Pressing. Although having only five states may lead to the consideration of an oversimplified model, they are sufficient to realistically represent the behavior of such a device, as we are interested in changes and not in values. The state Configuration A represents a state in which multiple buttons may be pressed and the analog joystick reports some coordinates. When there is a change in the sensed values (even just one press button reports a different value) the system transits in the Configuration B state — something has changed, we do not care what. Then, if there is another change (for example the joystick sensed a new value while buttons reports the same last values) the system transits again in Configuration A. This means that each state, Configuration A and Configuration B, correspond to different instances of values. If the system transits from Configuration A to Configuration B and then again to Configuration A, the current sensors values may be different from the values sensed when transiting the previous time to Configuration A. This model has been created to simulate a user who uses the controller three times a day and each time plays for one hour or less. These values have been validated by observing real players.

Camera The camera is the most demanding device in terms of required data rate. Shot images need to be fragmented before being transferred. We modeled a camera that takes a shot and sends the image fragmented in multiple packets at the highest rate allowed by the network. Only after sending the last fragment it can take a new shot. If the image size is 25KB [86], and time slots last $6ms$, the camera can send 40 bits at each query. Thus it can take a new shot every $30s$ (if the number of slots required to send the entire image is $(25 * 10^3 * 8)b/40b = 5000$, then it takes $5000 * 6ms = 30000ms = 30s$ to send an entire image). As other devices also need to access the channel and send their data, the camera takes a few seconds more to send an image, depending on how many devices are active at the same time (see sect. 4.5.4).

4.5.4 Results

The evaluation looks at the speed and reliability of our APT-MAC protocol. In terms of speed we first evaluate the "start-up" transient time — where devices have larger losses and then they settle into notable performance — and then the packet delay at steady state.

The "start-up" transient time is the time the protocol takes to learn transmission device requirements and minimize data losses (tuning the MaxQD value). Fig. 4.7 shows a snapshot of the number of lost data samples per device, when devices are started for the first time. Each device keeps a counter of sample changes since the last query and sends this value together with the last sample when queried by the reader.

The joystick and the presence sensor take very short time — respectively $1.5s$ and $1.29s$ — to settle into great performance, eliminating sample losses. The remote instead needs longer time to reduce sample losses — after $20s$ it loses on average 1 sample every $200ms$. As working sessions for remotes last on average $20s$, at the

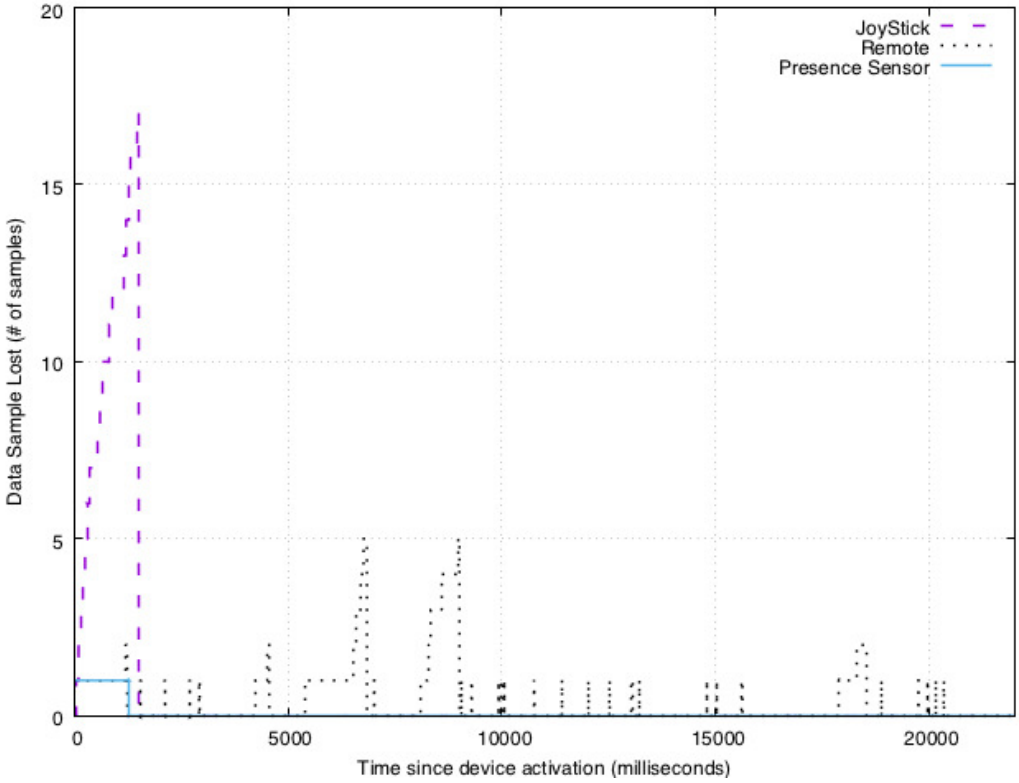


Figure 4.7. Devices transient time.

end of the first session the remote has reduced data loss, but has not reached the minimum MaxQD, so it will need another session to reach tolerable data loss.

The packet delay is a measure of the system speed to deliver sensed data to the reader in the steady state (once the reader has tuned the MaxQD per device). We estimate the transient state as the first 12 hours of operation, as the user may start a device, such as TV remotes, joysticks, light switches, even few hours after the system is launched.

Figure 4.8 shows the packet delay for 20 devices by varying device complexity (see Table 4.1). The Optimum shows the minimum achievable delay, i.e., $7.5ms$ regardless the case complexity. APT-MAC takes on average between $29ms$ and $35ms$. Compared with TDMA, APT-MAC is up to 2.8 times faster (in Case 3, TDMA takes on average $85.56ms$ to deliver data to the reader while APT-MAC is able to send data on average in only $29.9ms$). Although both solutions seem reasonable from an application requirements point of view — at the user level the difference in time is not noticeable — there is significant improvement by using APT-MAC in terms of data loss. As shown in Fig. 4.9, TDMA loses between 38.61% (Case 1) and 40.46% (Case 4) of new data samples, while APT-MAC does not lose more than 1.8% of new data in the worst case scenario (Case 4), being very close to the Optimum — whose data loss is always 0 — in the other three cases.

Figures 4.10, 4.12, and figures 4.11, 4.13, report, respectively, packet delay and data loss for 30 and 40 devices. The results do not deviate from those that we have seen for 20 devices — irrespective of the number of devices and their complexity, we

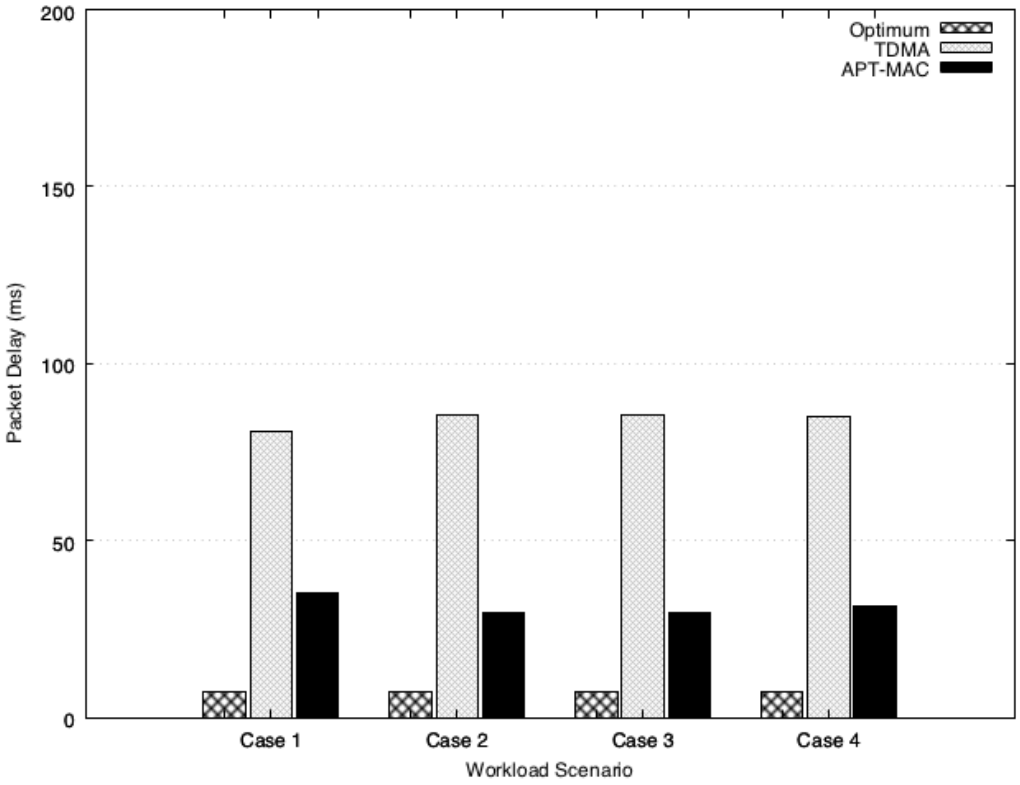


Figure 4.8. Packet Delay with 20 devices.

generally see that APT-MAC is always superior to TDMA. In the case of 40 devices, TDMA doubles the packet delay (164.38ms in Case 4) with respect to the scenario with 20 devices, while APT-MAC takes no more than 35.8.5ms in the same case to deliver new data to the reader (4.59 times faster than TDMA). Also data loss increases significantly with TDMA, up to 56.74% (Case 4 with 40 devices), while it increases slightly with APT-MAC, which loses 2.3% of the new data samples. Thus, our first set of results clearly shows the benefits of using a reinforcement learning based approach.

The second set of simulations investigates system performance in case of stressed conditions. We fix the number of devices at 40 and add a camera for home surveillance, which always has new data to send (continuous burst of data). Specifically, we consider three different cases. In the first the camera sends data at the maximum allowed rate, almost saturating the channel. In this case the reader gets a new image about every 33s. This value decreases to 30s in case the camera is the only active device in the system, meaning that the system cannot support cameras with higher shooting frequency. In the other two cases the camera reduces its shooting frequency to 45s and 60s, giving more chance to the other devices to access the channel. Table 4.2 reports the average percentages of packet losses per type of devices in absence and presence of camera. The global data loss does not include data generated by the camera because it never experiences loss: it takes a new shot only after having sent all data related to the previous shot. When the camera is off, the system loses 2.61% of data samples, with comparable loss for the different

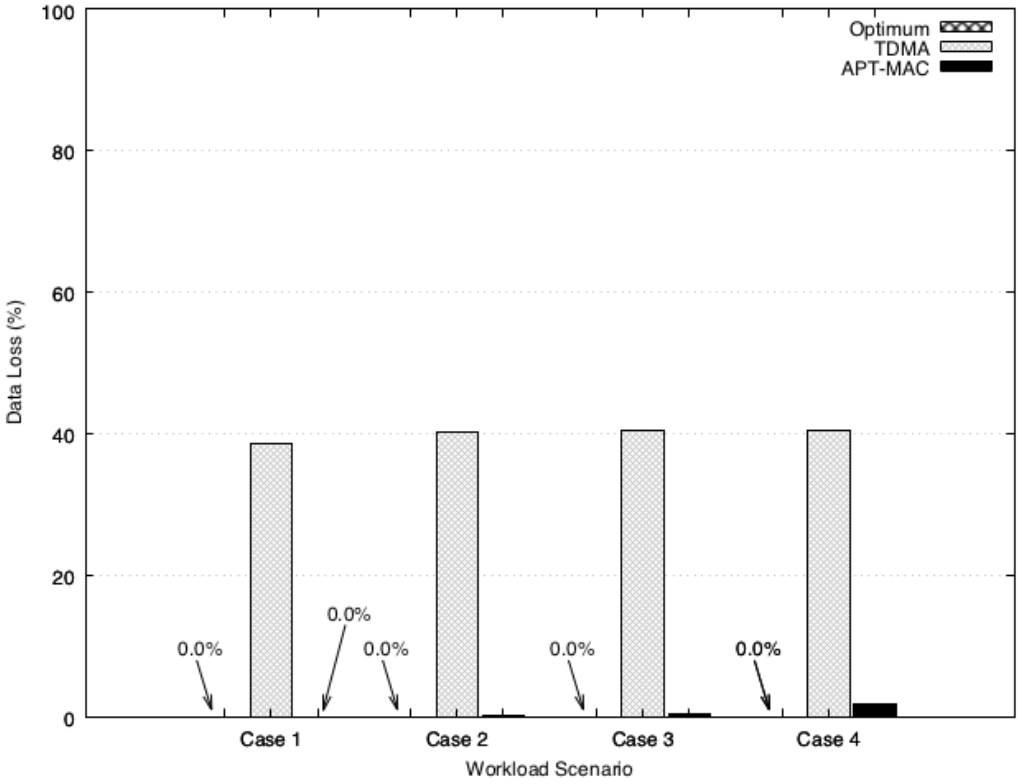


Figure 4.9. Data Loss with 20 devices.

types of devices (i.e., 1.18% for environmental devices, 2.86% for remotes and 2.61% for joystick). When the camera is on the global data loss increases as the shooting frequency increases (from 2.88% of data loss with shots every 60s to 4.91% for shots every 30s). In the last case, joysticks experience the highest data loss (i.e., 5.10%), that however remains tolerable to the user. The system degrades its performance if the number of cameras increases. In the case of 2 cameras with a shooting frequency of about 30s each, the system data loss grows to 10.5%, becoming more difficult to tolerate from a user perspective.

Table 4.2. Global and per-device percentage of data loss in presence of a camera.

	Global	Env	Remote	Joystick
Camera OFF	2.61	1.18	2.86	2.61
Camera ON (60sec)	2.88	1.30	4.04	3.68
Camera ON (45sec)	3.19	1.38	4.07	3.07
Camera ON (30sec)	4.91	1.23	4.77	5.10

In conclusion, a reinforcement learning based approach, employing the bandit algorithm, is able to efficiently manage systems of 40 battery free smart devices, including up to 4 joysticks and 4 remotes, and a camera with a shooting frequency of one image every 30s, without requiring any reconfiguration when devices enter and leave. Our approach is less suitable to more complex systems, involving multiple

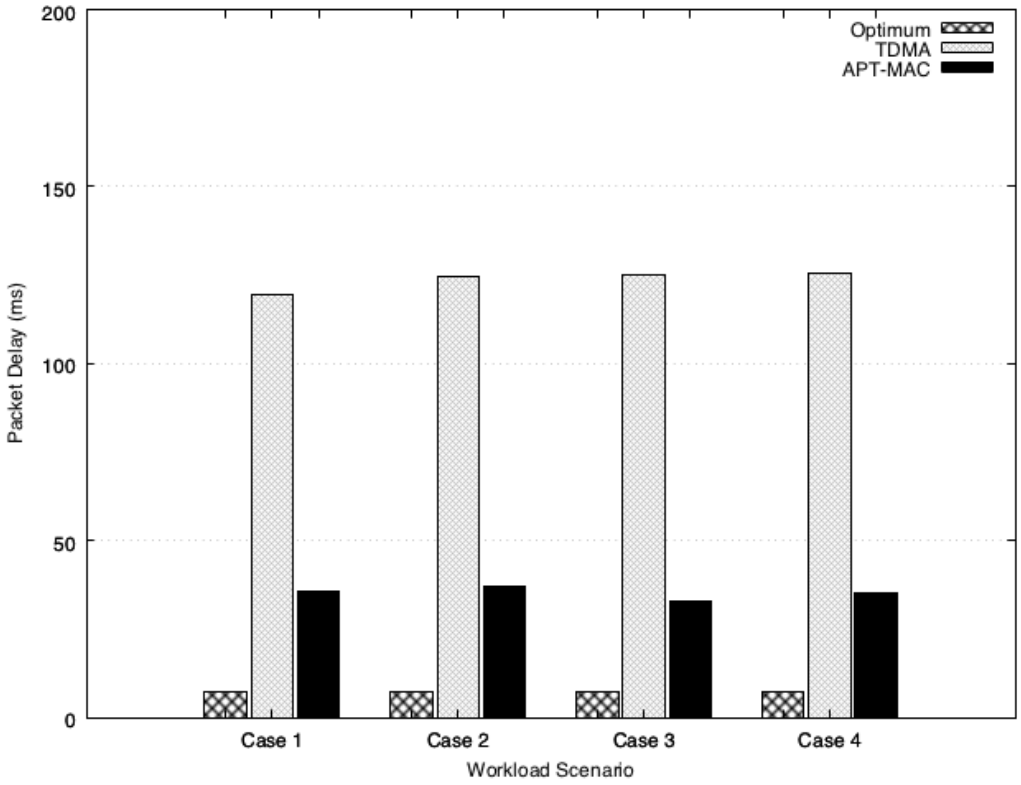


Figure 4.10. Packet Delay with 30 devices.

cameras — the algorithm becomes less efficient in assigning transmission slots to devices — or video cameras — due to limitations of the involved RFID technology, which does not allow sending images more frequently than every $30s$[86].

4.5.5 Evaluation in a Noisy Environment

Our discussion so far assumes that the communication channel is free of errors. However, as battery-free smart devices rely on backscattering, the signal they reflect back to the reader is weak and subject to interference from any nearby device operating on the same frequency, such as IEEE 802.11ah, amateur radio, ISMs, walkie talkies, and old cordless phones, (depending on the regional allocation of the radio spectrum).

The obvious question is how APT-MAC behaves in presence of a noisy channel. RFID transmissions — both reader-to-tag and tag-to-reader — include a cyclic redundancy check (CRC) code that allows to detect accidental changes to raw data [44]. If the CRC verification fails the received packet is discarded, without sending any feedback to the sender.

To evaluate the impact of a noisy channel, we evaluate protocol performance in the case of 40 devices (Case 1 as defined in table 4.1) by including a packet error rate, $PER = \{5\%, 10\%, 20\%\}$ on the communication channel. Fig. 4.14 presents data loss by varying PER. The Optimum shows no impact regardless the PER value. APT-MAC experiences an increasing $(+2.9\%, +3.9\%, +6.9\%)$ but still tolerable data

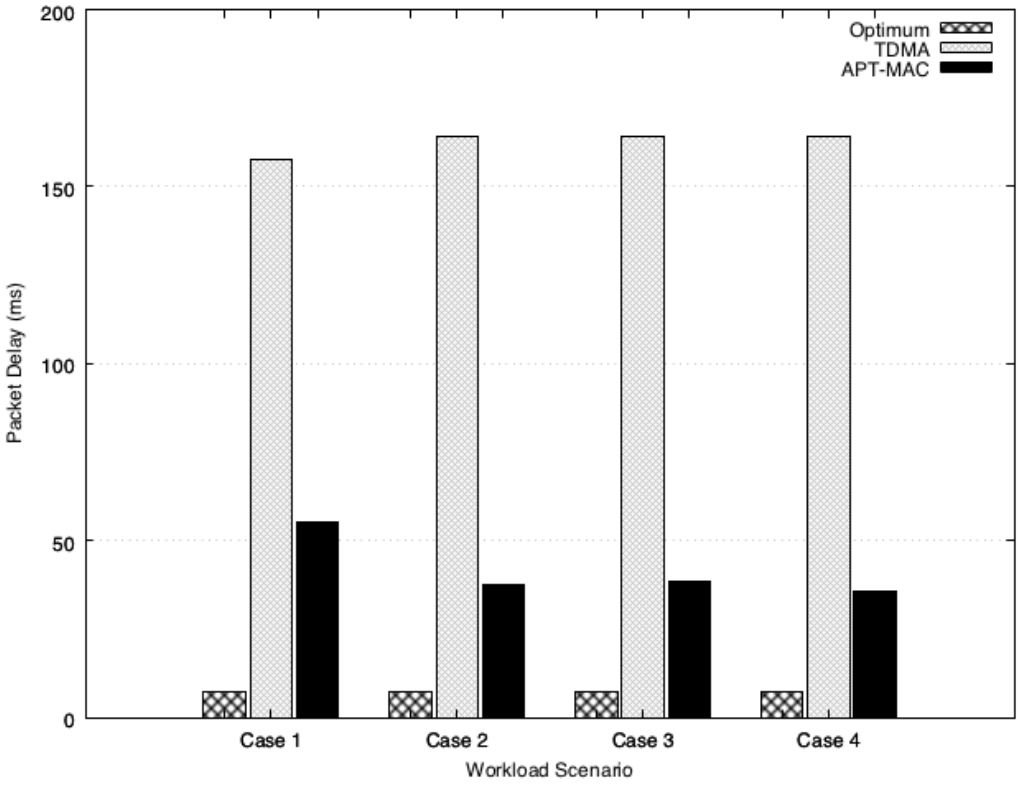


Figure 4.11. Packet Delay with 40 devices.

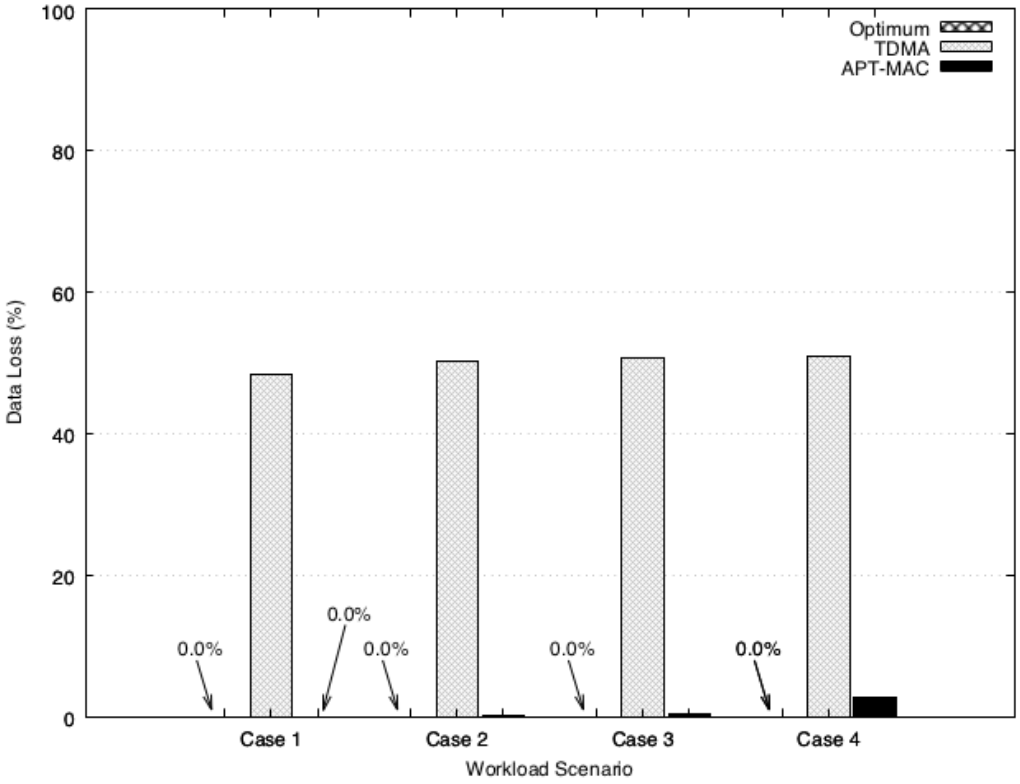


Figure 4.12. Data Loss with 30 devices.

loss, with respect to the case of no error, while TDMA worsens its performance achieving almost 50% of lost packets.

The main ability to tolerate data loss of APT-MAC is provided by data redundancy: the reader queries sensors at a higher frequency than that one at which sensors generate new data samples. For this reason, a lost packet does not imply a lost data sample. In case instead a new data sample is lost, the APT-MAC protocol automatically reacts by querying the sensor more frequently, diminishing the amount of lost data. In addition, in many cases, applications that use such data are implemented to be robust to some data loss.

Results on packet delay (see Fig. 4.15) show how the time between the generation of new sensor data and its delivery to the reader remains below $40ms$, which is reasonable for most of home applications. A commercial joystick, one of the most demanding devices inside a smart home, can work with a delay of about $100ms$, as found in [80].

4.5.6 Fairness

We now show that APT-MAC protocol guarantees a fair access to the channel, fairly distributing data loss among all devices. We define the fairness index — based on Jain's fairness [52] — as described in eq. 4.2.

$$\psi(x_1, x_2, ..., x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \cdot \sum_{i=1}^{n} x_i^2} \tag{4.2}$$

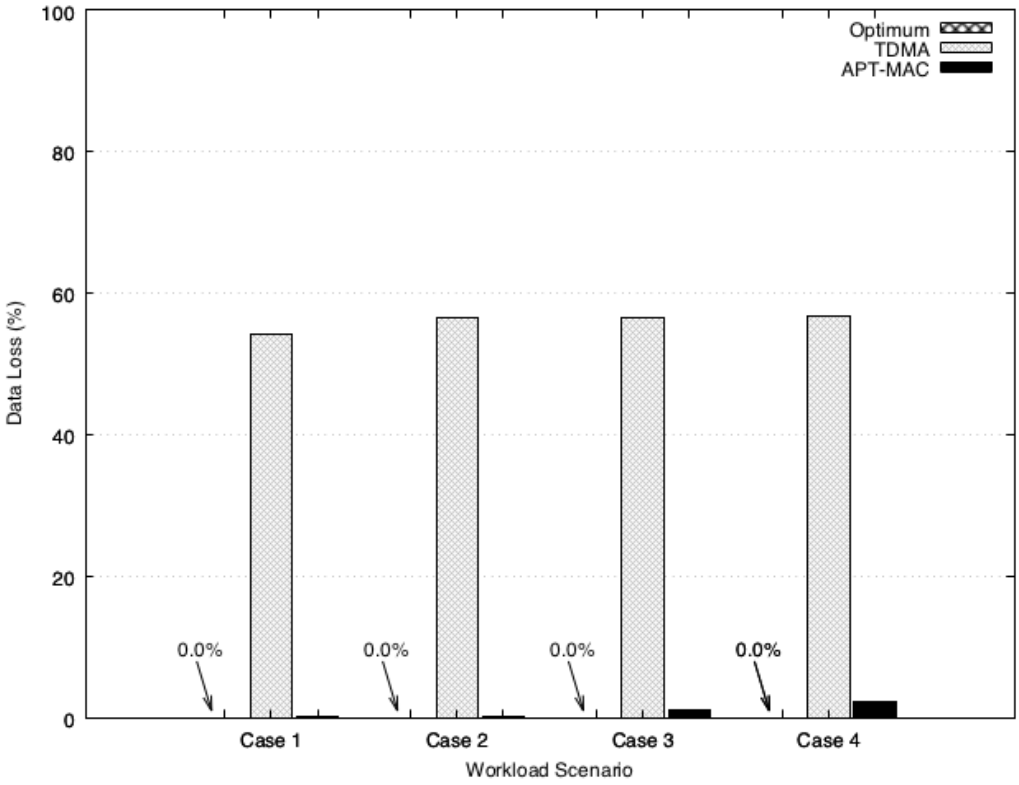


Figure 4.13. Data Loss with 40 devices.

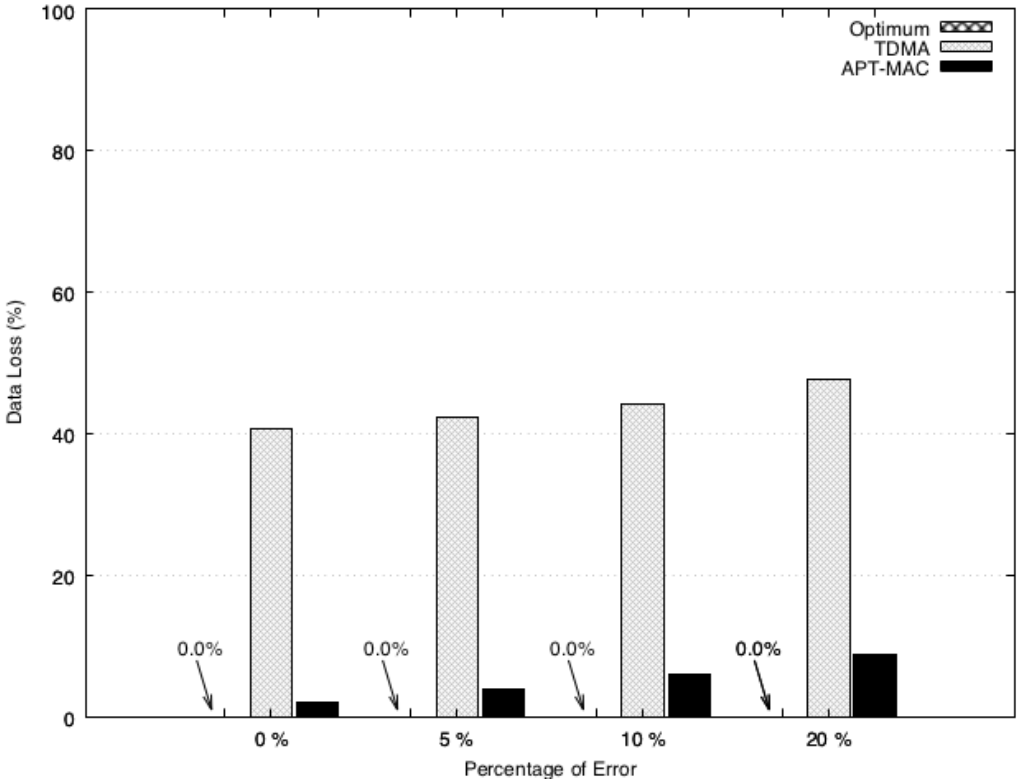


Figure 4.14. Data Loss in noisy environment.

where x_i is data loss for device i.

Results (see Fig. 4.16) confirm that APT-MAC protocol is fair independently of channel noise, presenting a fairness index equal to 1 when PER varies from 0% to 10% and equal to 99% when $PER = 20\%$. This is because the MaxQD time avoids any starvation problem, guaranteeing that even less operating devices are queried at regular intervals. In presence of channel errors this mechanism remains effective. TDMA instead decreases its fairness from 96% in absence of noise to 94.23% when $PER = 20\%$.

4.6 Related Work

We discuss related work that we have not touched upon in the previous sections.

Information Collection from Sensor-Augmented RFID Tags - A few solutions have been proposed to collect information from sensor-augmented RFID tags. A couple of recent solutions are based on Hash functions [27][99]. The idea behind these works is to make the reader send only one query. Many of the devices in the environment will answer subsequently, without the need of a query for each of them. Exploiting a hash function as an index, the reader is able to understand in which frame it will receive no answer, one answer or a collision. With this information available the reader can send a bit vector in which a 1 means that the tag with that index should transmit, otherwise it should not. In this way the reader is able to avoid

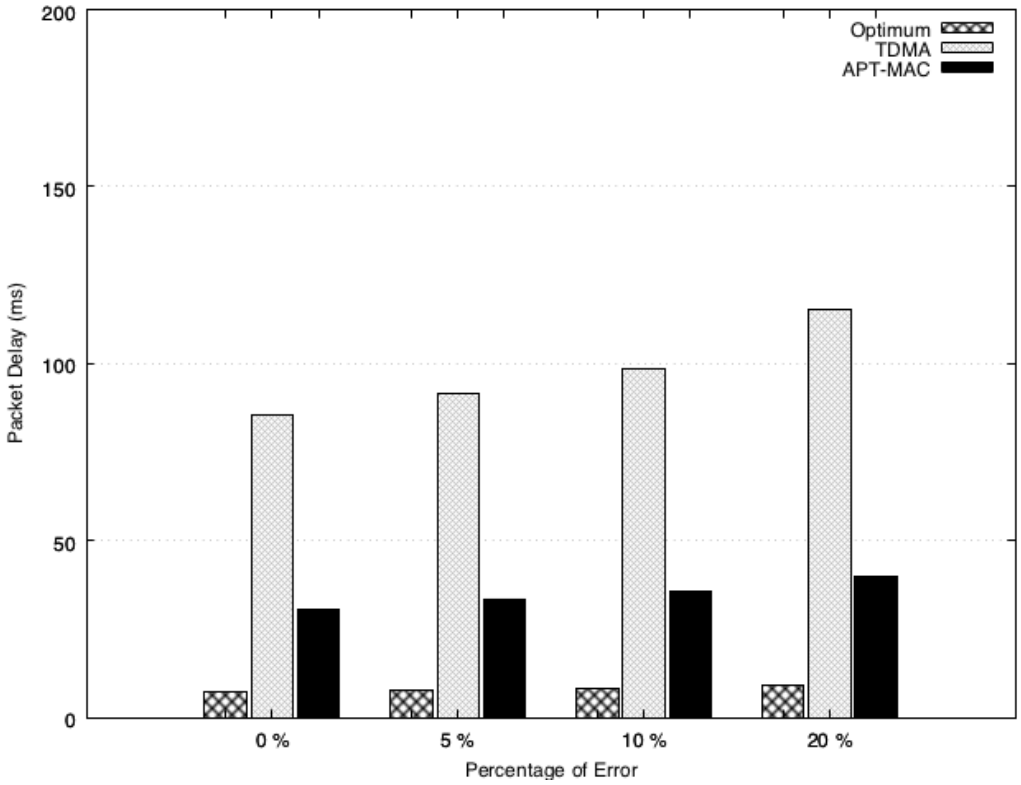


Figure 4.15. Delay in noisy environment.

collisions and optimize transmission. Differently from our system, these protocols can only reduce the number of empty slots and optimize transmissions, but they are not able to adapt to current tags needs (e.g., give priority to more demanding devices). The vector index is defined by the hash function, and can not be defined depending on the device need to transmit. Furthermore, as the reader can send only one query which contains information regarding many successive slots, these protocols are not able to dynamically adapt to burst data. The backscatter concept appears also in [50], but the paper presents mainly a modeling contribution, based on Poisson assumptions, stochastic geometry, and evaluated only via simulation.

MAC protocols for Smart Homes - Current smart home devices are mainly based on technologies such as Zigbee and Wi-Fi, or mainstream systems like Amazon Echo. At the MAC layer, these technologies operate according to a CSMA approach, that cannot be adopted in the context of this work, because RFID tags can not communicate spontaneously: they need a centralized entity (e.g., the reader) that energizes and queries them.

4.7 Technology Potential over Time

Fig. 4.17 shows the range of devices that can be handled today with RFID technology, by varying the sensing frequency and the data rate. A reader that has a transmission power of $P_t = 0.5W$ achieves less than one meter of distance between the antennas

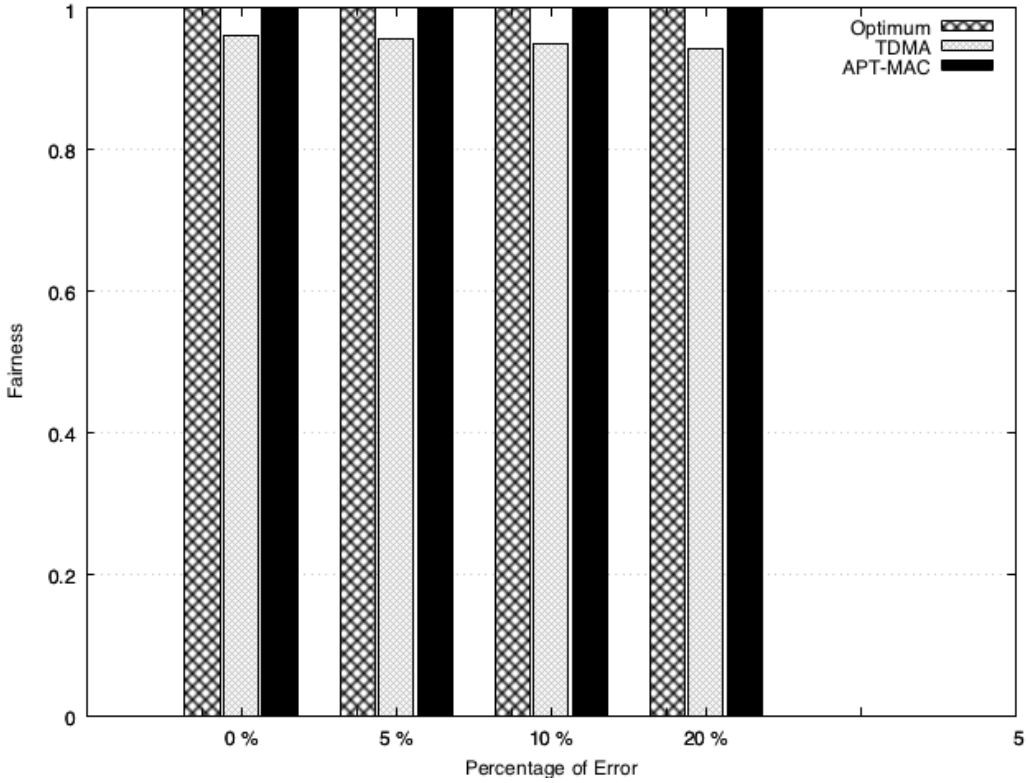


Figure 4.16. Fairness on data loss.

and the tags. With this technology we can realize several smart devices (e.g., videogame controller, light switch), but we cannot realize a videocamera. A more powerful reader (e.g., with $P_t = 1W$) allows for longer distance (up to 3 meters) between the reader's antennas and the tags, but cannot satisfy a real-time frequency. Wi-Fi technology achieves the best results, allowing for data transmission also from a videocamera, but is it not battery free.

Many constraints of our RFID based solution are related to technology, and introduced by our prototypical implementation. There are a number of improvements that may be employed in a real environment in order to achieve better results in term of bit rate, distance, and energy available.

Transmission Power In our scenario we exploit an USRP RF daughterboard modified in order to transmit 500mW of power, only half of a commercial reader. Exploiting the Friis Equation and the work [110] we can predict an operating range of 3.3m, just changing the utilized reader (e.g., with transmission power equal to $1W$) and without hardware modification to RFID devices.

Exploiting hybrid harvesting (RF + light) In [148] we can find an example of a RFID device powered from both RF harvesting and a small solar panel (3cmx3cm). In this case it was shown to reach a distance of 7 meters and a maximum bitrate of 21.7kbps. In our specific case, the smart home, we should consider the energy

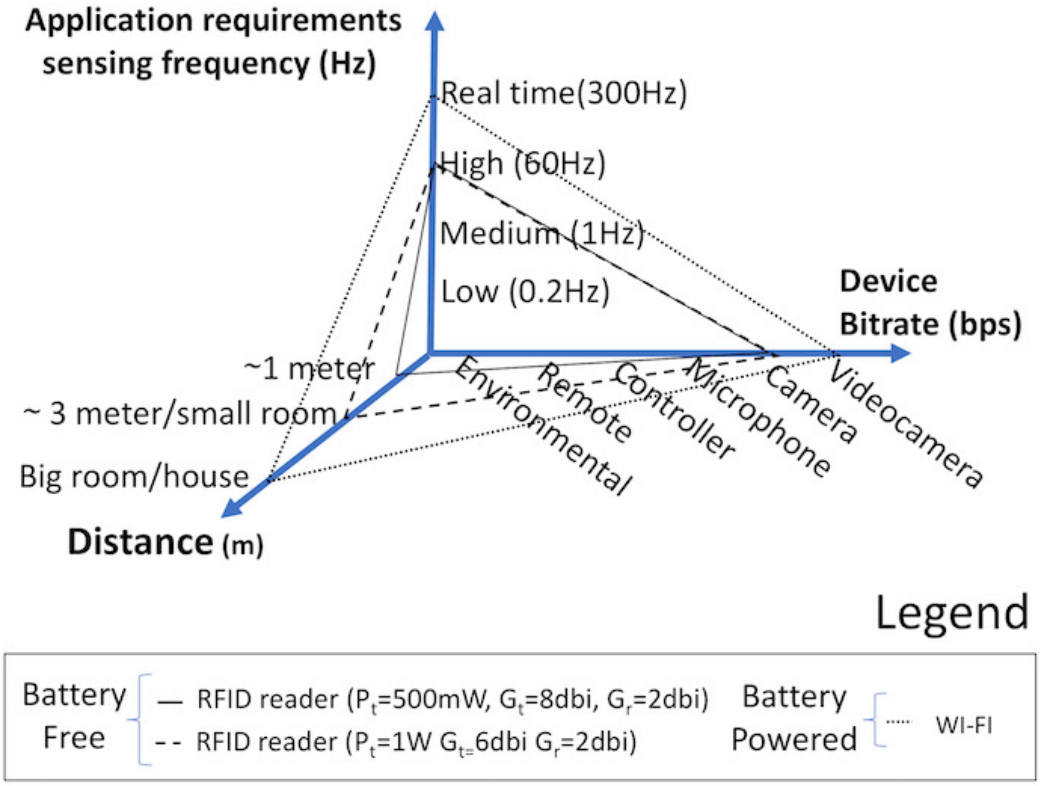


Figure 4.17. Devices that can be handled today with RFID (compared with Wi-Fi) technology depending on the distance between devices, datarate and data frequency required by applications.

harvested inside a room. The minimum amount of Lux required in this case in order to operate is 150 Lux, which is less than standard natural indoor light.

4.8 Energy Consumption and Health Issues

We now evaluate the energy consumed by a continuously transmitting reader and its impact on users' health.

With the help of an amperometer, we quantified the energy consumption of our prototypal RFID reader in 1.32A, with 6V, for a total of about 8W. In one year, our reader consumes less than 70KW/h. In the USA, the maximum price for one KW/h is always less than 20 cents, involving a cost of less than 20$ per year. We should even note that consumption of a commercial reader is expected to be less than that of our prototype.

From a health standpoint, the work in [13] studied the absorption rates in the human head and shoulder for Passive UHF RFID Systems at 915 MHz and found that in an ideal absorption environment, an RFID reader located within 10 centimeters (3.9 inches) from the human head presents a specific absorption rate above the maximum value allowed by the United States' Federal Communications Commission ($1.6W/kg$ for both the spatial-peak 1 g and 10 g cube of tissue). Consequently, really close proximity to UHF RFID readers has potential health issues, particularly when close to the eyes. To avoid any potential harm to humans, UHF RFID transmitting antennas should be set back at least 0.5 meter (1.6 feet) from anyone who might receive constant exposure. If the antenna is within legal power output limits, and is kept at least 1 meter (3.3 feet) from the human body, the incident radiation – even on the eyes – is at a level well below maximum allowable levels. In a battery free smart home there is one transmitting antenna per room, typically located on walls. A user inside a smart home rarely walks or resides next to the walls. In addition, RFID antennas can be placed far away from sofas or tables with chairs.

4.9 Open Issues and Future Work

Besides the issues related to the RFID technology (discussed in sect. 4.3), open issues related to the APT-MAC protocol are mainly related to specific use cases. The protocol has been designed for environments with heterogeneous devices: in smart homes we expect to find a set of real time devices, a set of event based devices and a set of periodic devices. In such a case, APT-MAC outperforms TDMA, as it is able to differentiate channel access based on estimated devices behavior. But when the environment is atypical | featuring only real time or periodic devices | APT-MAC loses effectiveness, converging to a TDMA-like protocol with slots equally allocated. We believe that although very rare, these situations should be properly addressed by the protocol, to guarantee high data collection performance. We leave this issue as future work.

4.10 Conclusions

With APT-MAC RFID technology can be employed to connect a variety of battery-free smart devices, performing real-time, periodic, and event based sensing. Considering a wide set of battery-free devices, we defined a system to realize a battery-free smart home, specifying the system architecture — at both the hardware and software level — and the MAC protocol — called APT-MAC, that dynamically allocates transmission slots to devices without requiring any a priori knowledge of the environment. The key aspect in APT-MAC is the use of reinforcement learning that allows to dynamically adapt to user and environmental behavior. Results clearly show the benefits of our approach. The system efficiently handles scenarios with 40 different devices including a camera shooting a new image every $30s$.
However, even if adaptive, we noticed that the APT-MAC protocol could be improved by introducing some memories of the past. APT-MAC does not memorize device's patterns, and each time a device changes transmission pattern the protocol has to learn from scratch. In the next chapter we introduce an enhanced protocol which memorizes devices transmission pattern, and can adapt the network allocation in an optimized manner.

Chapter 5

Environment-driven Communication in Battery-free Smart Buildings

5.1 Chapter Introduction

In this chapter, we propose a new communication protocol, called ReLEDF, which dynamically discovers devices in smart buildings, their active and nonactive status, and when active their current communication behavior — through a learning-based mechanism — and schedules transmission slots — through an Earliest Deadline First (EDF) based mechanism — adapting to different data transmission requirements. Combining learning and scheduling introduces a tag starvation problem, so we also propose a new mode-change scheduling approach. Extensive simulations clearly show the benefits of using ReLEDF, which successfully delivers over 95% of new data samples in a typical smart home scenario with up to 150 heterogeneous smart devices, outperforming related solutions. Real experiments are also conducted to demonstrate the applicability of ReLEDF and to validate the simulations.
Let us imagine a smart building, outfitted with a myriad of battery-less sensors and smart devices (e.g., cameras, presence sensors, smoke sensors, light sensors, thermostats, smart meters, those with real-time user interactions, etc.) that are used to reduce resource consumption and improve the quality of life. These smart systems will contain many devices with widely varying communications requirements depending on sensed events, transmission rates, number of bits per sensing sample, ON/OFF activity, and deadlines. Many of these devices will also have dynamically varying communications requirements based on their current operational mode. In fact, the nature of the sensor data generation is spontaneous; it is highly dependent on the events in the environment. Many applications are interested in changes in measured values: they want to get new data when the sensed value has changed since the previous sample, rather than at the device's sensing rate. As an example, a temperature sensor has the capability of sensing the environment every 25 ms, but a significant change in the temperature (for example of at least 1°C) since the last reading may happen after 1 second in case of a fire, or 1 hour in case of normal conditions. To pick another example, a joystick may sense no changes for hours

(while it is OFF), and then start sensing new data (while used for playing) at very different rates (from a few milliseconds to one or more seconds), depending on the game type and player's activity. Thus, the rate at which sensors detect a new value depends on environmental events. In other words, it is the environment that changes and while the sensor's sampling rates are set according to the Nyquist sampling theorem, to minimize overhead, wireless traffic, and handle large numbers of devices, it is often necessary to act only when there is a significant change in the sensed value. Consequently, in a smart building setting, *a device should transmit only when it has new/changed data to send*. This is a challenging task when there are many heterogeneous devices that have data to transmit at the same time.

MAC protocols for RFID-based devices have to follow a time slotted scheme (i.e., TDMA), because they need to be energized by the reader. However, current TDMA protocols based on a random access (i.e., the tag randomly selects a slot to transmit) perform poorly: collision time is above 50% [63]. A better solution to avoid collisions would be to poll directly each device with a new data sample. But the reader should know when a device has generated new data to query it. Hence, the need for the reader to *guess* which devices should be queried.

For many devices, especially those interacting with humans, it is also necessary to deliver data by a *deadline*. IoT environments will also evolve with devices being added and removed from the environment, with a need for zero configuration capability. Consequently, it is a significantly difficult challenge to design a communication protocol that dynamically discovers (current) environmental patterns, changes in those patterns, and required rate for reading/transmission, for devices that are added and taken away over time. In other words there is a need for a communication protocol that is able to detect, learn and monitor devices' presence and transmission needs (depending on events in the environment) and dynamically adapt to current device states and requirements, including deadline requirements for packet delivery. Meeting deadline requirements are especially challenging when device requirements dynamically change and new devices activate, which requires schedules for packet delivery to be adjusted. Maintaining performance requirements in the transition period between the current and new schedule is also very difficult. Further, in these highly dynamic environments it will not always be possible to meet all performance requirements so it is necessary to establish solutions with *fairness*.

Given the resource and computation constraints typical of backscattering-based devices, as well as the large heterogeneity among devices in a smart home or building, realizing efficient and effective communication is a big challenge.

With respect to ReLEDF, in APT-MAC the bandit algorithm provides a limited view on devices' behavior: it adapts to changes in the environment without considering previous history of changes. This may cause slow protocol adaptation.

For quicker reaction to changes in the environment, we propose a novel MAC protocol for battery free networks, called ReLEDF, which exploits a Q-learning based approach coupled with an EDF-based device interrogation mechanism. The use of Q-learning is motivated by its ability to quickly adapt to changes (by considering previous history of changes), while EDF guarantees time requirements of scheduling packet transmissions. The mix of learning and scheduling is an innovative way to schedule the network, which guarantees high performance but also brings a novel challenge: how to ensure that all devices meet deadlines without starvation. We

address this issue by proposing a new mode-change scheduling approach, called *EDF with Inherited Jobs* (EDF-IJ), which reduces delays and avoids device starvation in scheduling transmissions. In more detail, the main contributions of this work are:

- A *new dynamic and adaptive MAC protocol*, called ReLEDF, able to optimize transmission slots assignment based on device needs, exploiting the combination of Reinforcement Learning (RL) techniques and EDF real-time system policies.

- A *new computational approach to learning* current active devices and their dynamic requirements from interaction with the environment, inspired by RL. Our system introduces the concept of sub-agents — multiple entities that are influenced by actions defined by the same agent — and allows building/maintaining knowledge to make predictions on the state of multiple devices inside a network, enabling a highly dynamic transmission scheduling based on devices' behavior.

- A *no transition starvation* solution for Earliest deadline first (EDF) which is able to avoid starvation of low priority communicating devices while admitting new devices and changing schedules.

- A *new mode-change* scheduling approach based on EDF, called *EDF with Inherited Jobs* (EDF-IJ), that is able to reduce delays and avoid task starvation by immediately scheduling a new frame while keeping some information about previously scheduled jobs.

- A new theorem to prove *feasibility analysis* for EDF-IJ, which formally demonstrates that the feasibility of this new approach can be verified in constant time.

- A *fair loss policy* for EDF scheduling which is able to fairly penalize tasks and reach schedule feasibility when there is overload.

- A thorough *comparative performance evaluation* of ReLEDF with APT-MAC [79], TDMA, and optimum protocols. By means of extensive simulations in scenarios involving up to 150 heterogeneous devices, we show that ReLEDF scales well as it is able to always successfully deliver over 95% of packets with negligible delay, and outperforming state of the art solutions.

- An *experimentation on real prototype devices* to validate simulation results and show the applicability of our new protocol through real experiments.

5.2 Related Work

As this work gives a twofold contribution, i.e., on communication protocols for battery-free devices and a new mode-change scheduling approach for EDF, we present the related work in both fields.

5.2.1 Communication Protocols for Battery-Free Devices

In this section, we discuss existing solutions to collect data from sensor-augmented RFID tags and their limitations. However, as a similar related work has already been presented in chapter 4, we focus on the protocol closest to our work, APT-MAC [79]. Specifically, APT-MAC exploits the bandit algorithm to select the next device to query: if the device has a new sample the reward is positive, if instead the device does not have new data since the last reading the reward is negative. The main limitation of bandit is the lack of previous history of changes, because it keeps only one state (i.e., the current one), and the reward taken under consideration is only the immediate one. Hence, bandit can be thought of as having single-state episode, meaning that the algorithm is not able to remember past devices' behaviour. Every time a device changes behavior, the bandit starts learning from the beginning.

5.2.2 EDF Mode Change

We now present a number of useful related works for each of the topics we address in the chapter (a new scheduling policy, a new feasibility test, and a policy on how to manage overload scheduling).

Deadline scheduling has been deeply studied in the literature, for example [120] offers a prospect of available algorithms in the context of real time systems. Even the issue of dealing with dynamic tasks received huge attention from the scientific community. The problem has been divided into two main areas: the first addresses dynamic workloads, in which some of the tasks to be executed dynamically change, while the second addresses systems with multiple working modes: in this case a change in the environment drives the system from an operating mode to another, asking a number of tasks to be deleted or released. In the case of EDF, a recent solution for transients (i.e., periods of time in which tasks leave and join the schedule), has been presented in [26], where an online protocol is able to manage the admission control of a dynamic workload. In [24], a scheduling frame with elastic coefficients related to each task has been proposed. In this case, the system is able to modify the period of tasks in order to adapt them to the transient, but no guarantees on preceding tasks deadlines are given, thus causing packets loss. Another solution, EDF-VD[17] (extended by [82]), is able to manage tasks arrivals assigning shorter and "virtual" deadlines to critical tasks. In [20] a partitioned scheduling scheme able to re-weight tasks has been proposed, but the paper also points out how the proposed scheme is not capable of providing fairness or real-time guarantees. The main limitations of these works are in terms of starvation risks — less demanding tasks during mode changes could be excluded from the schedule — and in terms of fairness — most demanding tasks, in absence of priority, may completely saturate the communication channel at the expense of less demanding tasks.

In the context of how to deal with mode change the literature presents several solutions [41][100][114]. In particular, the work in [114] proposes a protocol for mode change in a preemptive scheduling environment. In [122] the authors address dynamic mode change in real-time systems under both EDF and Fixed Priority scheduling policies, but, in this work during a mode change no other tasks changes are accepted. In a context more related to networking, [123] investigates the adaptive

reservation of resource provisioning for servers using a TDMA approach. Another proposal presents a protocol for multicore system scheduling under the assumption that, in a mode change, only part of the tasks change requirements [87]. While some of these proposals are able to guarantee absolute deadlines handling mode change in real-time systems, they are inappropriate for a networking context as they do not consider the same execution priorities, and they do not allow continuous and independent changes in the task set. Even those proposals more related to networking do not fit with our problem: the tasks cannot independently change requirements and there are admission delays or admission control systems. Differently, we attempt to create a new scheduling policy that is compatible with every already presented TDMA-based MAC protocol, and instead of refusing new tasks we aim at *fairly* updating the task set in order to make it feasible.

Regarding the feasibility test, it is known that when a new task enters a system, in order to avoid missing deadlines, the task has to satisfy an admission test, which typically consists in a feasibility test. Generally, in a transient situation in which tasks are leaving and entering the system, a feasibility analysis which does not consider leaving tasks is not safe [24][130]. Furthermore, in our case we want also to consider some jobs of the old scheduled frame while producing the new one. In [40] two feasibility tests for EDF with mode changes have been proposed: the first one considers a fixed change sequence known a priori, while the second one allows the system to change between a limited a set of modes. Unfortunately, both these tests are not applicable to EDF-IJ, as they do not consider jobs inherited from a number of previous schedules. Another approach to the transient problem consists in transforming temporally some periodic tasks to aperiodic. While an admission controller for tasks set composed of both periodic and aperiodic jobs has been proposed by [9], this transformation has an impact on the whole scheduled frame, while in our proposal we produce changes only in the initial part of the scheduled frame.

For the management of over scheduling, an interesting approach is presented in [24], where a resource manager is able to adapt tasks deadlines, depending on an elastic parameter, in order to make the task set schedule feasible. This approach, different from our proposal, does not take into account fairness between tasks while changing their periods.

5.3 Overview and Assumptions

Our goal is to design a protocol that dynamically discovers when devices have new data to send (different from the previous sampling value) and adapts to their different communication requirements. This protocol is practical to use in IoT environments, i.e., devices may dynamically change their transmission modes by detecting different type of events in the environment. In particular, we consider the case of battery-free devices [74], realized through sensor-augmented RFID tags, such as the Moo [147] or the Intel DL WISP [144] tags. These devices are computational RFIDs. Specifically, we use Moo tags, that are built on the prototype of WISP tags, and feature reprogrammable microcontrollers, sensors and actuators, nonvolatile memory, and small energy buffers that temporarily store harvested energy for computation

[147]. Moo tags also allow for function extension, thanks to their general-purpose I/Os, serial buses, and 12-bit ADC/DAC ports. The key feature of Moo devices is the energy provisioning: they operate through RFID Backscattering, harvesting power from signals emitted by a dedicated RFID reader, to sense and communicate data. In particular, the reader emits a *continuous wave* which is absorbed by devices to get powered and sense. The devices eventually communicate by reflecting the continuous wave, modulating messages for the reader. For a survey on capabilities and applications of computational RFIDs, as well as more details on physical layer aspects we point the reader to [74].

Importantly, these devices present specific peculiarities that significantly limit their communication ability. Backscatter communication introduces different MAC layer communication mechanisms. In particular, sensor-augmented RFID tags can communicate only with a dedicated device, i.e., the reader, and cannot practically communicate with each other (the network is single hop). In addition, they cannot perform carrier sense, nor transmit spontaneously, because they have no power source on-board. Hence, communication has to be controlled by the reader that has to query (and hence energize) devices to collect their sensed data. For this reason, we adopt a centralized approach to dynamically gather data from sensor-augmented RFID tags, as tags backscatter the signal received by the reader. In particular, in sensor-augmented RFID systems it is natural to designate the reader as the master of the communication protocol, a centralized entity that receives devices packets, and schedule device transmissions. This master also periodically runs an inventory query to handle devices entering or leaving the area.

Considering the typical characteristics of RFID technology, we make the following assumptions:

- The network has a star topology, with the reader that directly queries all devices. This approach is reasonable in a smart home because the reader can be plugged in and the rest of the home devices (potentially many) have no batteries. It has been studied that the improvement achieved on the environment by making devices battery-free is greater than the cost of powering a continuous operating reader [79]. In case of large areas with multiple or vast rooms we assume that the RFID reader is equipped with multiple antennas, one transmitting and one receiving for each room. Each antenna has a transmission range that is able to reach all devices in the same room. When the reader issues a query, all the transmitting antennas broadcast the query at the same time, reaching all the devices in the home. When tags receive a query, only the queried tag (whose ID is indicated inside the query message) backscatters the received signal to send its sensed data to the reader; all the other tags use the received signal to power on-board sensors and store the sensed data in a local buffer. At each query only one device in the home is interrogated. Multiple receiving antennas may receive its response (e.g., antennas in nearby rooms), eventually producing redundant information.

- To minimize configuration costs and handle the IoT world we assume that devices freely enter and leave the system without communicating any information about their type (e.g., temperature sensor, TV remote, etc.); they just receive an ID from the system and start operating. In other words, the

reasons for assuming no prior knowledge of devices are twofold. First, to allow devices to declare their requirements without specific interaction with the user, the system should have access to a global up to date database, containing information on all possible devices, which would be laborious and error-prone. Second, the rate at which sensors need to send data is not necessarily the rate at which they sense the environment. Sensor transmission rate requirements are given by the rate at which they detect a new value (different from the previous sampling), which depends on environmental events. Consequently, a device should transmit only when it has new/changed data to send. Thus, knowing the device sensing rate does not imply knowing the device communication needs that may be unpredictable. For example a joystick may have different transmission rates depending on its current use (e.g., on/off moments as well as different video-game requirements and player activity). For these reasons, we believe that it is simpler and easier to have an automatic and configuration/management free protocol, such that proposed in this chapter, that can adapt to any newly installed or invented device and any environment change/condition.

To achieve environment-driven communication in such a context, the primary issue that we need to deal with is learning and continuously monitoring environmental patterns and changes in those patterns, which require specific rates for reading/transmission. As sensor transmission rate requirements depend on new/changed sensed data, from now on we define *transmission rate* as the rate at which a device needs to send data regarding a new event in the environment (again, to be clear this is not the sensor sampling rate). The second important issue is to schedule channel access such that data is delivered in time.

We tackle these problems through an innovative MAC protocol that integrates two key components: a RL based mechanism and an EDF real time scheduling policy. We also address the challenges of mixing learning and scheduling, which guarantees high performance but also poses a critical issue on the way to *ensure that all devices meet deadlines without starvation.*

We propose a new MAC protocol, named ReLEDF, which quickly learns transmission rate requirements of active devices, without having any a-priori knowledge on the type of devices, and properly schedules devices transmissions. The protocol relies on two key components: 1) a RL based mechanism that builds and updates the devices' behavioral models; and 2) a channel access method based on EDF scheduling, improved to deal with schedule transitions, that maps devices behavior models into a network frame. Figure 5.1 presents the overall system. In particular, the figure shows a smart home with heating, fire alarm, light switches, cameras, joysticks and a TV remote as possibly found in a smart home. As possibly multiple users activate any of these devices by using them, the RFID reader detects this and supports the appropriate scheduling and communications. Later, a home owner may add a smart humidity control for plants, and a home security system. Such new devices are automatically handled together with the current devices if and when they are used.

The two components (i.e., RL and EDF) continuously interact and change upon transmission feedback. In the following we detail the two components.

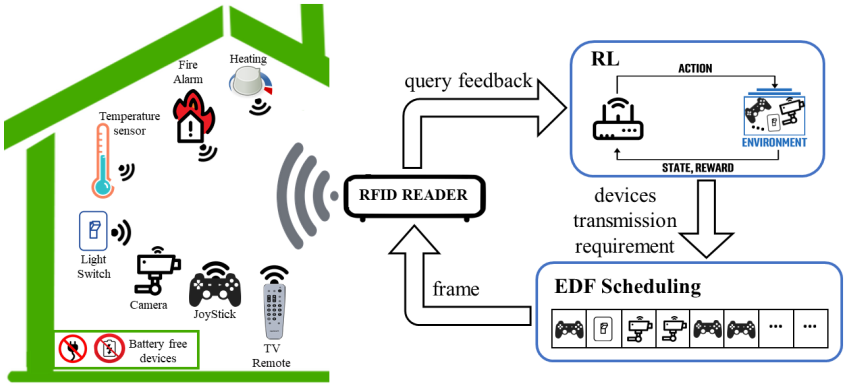


Figure 5.1. While receiving data samples, the reader learns and builds behavioral models. In particular, the RL component estimates the current transmission requirement for each device. Then the reader maps transmission requirements into a communication schedule (frame) through an EDF policy.

5.4 ReLEDF: Learning Devices Behavior

The first component of our system is the construction of a dynamic behavioral model for each network device, in order to learn required transmission rates (at the current time and in the near future). We briefly describe our idea and then present more details.

We want to model device behavior (i.e., the rate at which devices detect new events in the environment) with a graph, where nodes correspond to transmission rates and edges represent changes in transmission rate. As the system does not have prior knowledge on device's type, we consider several nodes in the graph, corresponding to a reasonable set of possible rate requirements $tr_1, tr_2, \ldots, tr_n$, where tr_1 represents the highest rate (for example transmission every $20ms$) and tr_n the lowest (for example once every $2sec$).

The lowest transmission rate represents the maximum delay between the generation by a sensor of a new data sample and the transmission of the corresponding packet to the reader. This means that also a dormant sensor, e.g. a fire alarm which is rarely enabled, in our set up is queried at least every two seconds. Notice that, we consider a discrete finite set of transmission requirements to employ Q-learning algorithms. However, device behavior involves only a subset of the nodes: devices typically move through a small subset of transmission rates. As an example, a temperature sensor mainly remains in a low state in (e.g., tr_{n-2}, tr_{n-1}) in case of normal conditions, while a joystick may pass from a stand-by state when it is OFF (e.g., tr_n state) to a burst transmission state, when used for playing (e.g., tr_1 or tr_2 states, depending on user activity and game characteristics).

How do we represent the subset of states modeling a device behavior? We add weights on edges: the value associated with the edge $tr_i \rightarrow tr_j$ represents the probability to change rate from tr_i to tr_j. Our goal is to have for each active device an instance of the graph that at each time truly represents current device communication behavior. The technique that we use to discover which state of the graph more likely reflects current device behavior is based on reinforcement learning.

A brief primer on Reinforcement Learning: In RL problems, an *agent* interacts with the environment and at each time t, is placed in a certain *state* s in the set of states S, and takes an *action* a in A. The agent decides the action to perform depending on a *policy* $\pi(a_t|s_t)$. The policy, for each action a_t and each state s_t, defines a scalar *reward* r_t and the transition to the next state s_{t+1}, with the help of the state transition probability $P(s_{t+1}|s_t, a_t)$ and the reward function $R(s, a)$. While running, the agent tries to reach a terminal state (with the highest reward). However, if the problem does not reach a terminal state, but keeps changing over time, as it is the case addressed by our work, then the problem is called non-stationary.

In the context of our work, considering a single device, the *agent* corresponds to the *interrogator* or *master* — the device dedicated to deciding which device to poll for transmission — *states* represent device's transmission *rate requirements*, *actions* concern *changes of transmission rate*, and the *reward* concerns *finding the current rate requirement* of the device (corresponding to the real rate requirement).

As we have multiple devices with heterogeneous rate requirements, and different variations over time, but only one master, we introduce *sub-agents*. A sub-agent receives the rate change from the agent and changes state. The master performs an action, i.e., selects a device to query, and the sub-agent changes state (i.e., the master polls a device and updates its transmission requirement, while the sub-agent transits eventually in a different state).

The set A of actions represents transitions between states. We have two types of actions: *mandatory* actions, in which the sub-agent has to move to another state; and *non-mandatory* actions, in which the agent asks the sub-agent to move to another state in a set of states, but the sub-agent's decision depends on the corresponding expected reward (it may happen that it remains in the same state).

Let us now see how RL can be used to learn and monitor devices' behaviors. In order to keep track of device's states and actions, we introduce a graph $G_{dv} = \{S, E, W_{dv}\}$, in which S is the set of states $\{s_i|1 \leq i \leq n\}$, E is the set of directed edges, where $e_{i\to j}$ represents a transition from state s_i to s_j, and W_{dv} is set of weights labeling edges $w_{i\to j}$. In our model, a state s_i corresponds to a transmission rate tr_i, an edge $e_{i\to j}$ corresponds to the action a_j in A, with (i, j) any pair of nodes in G, and a weight $w_{i\to j}$ is tied to the action of going from state i to state j. While the sets S and E are the same for all devices, W_{dv} depends on the specific transmission requirement of each device dv. As the transmission requirement depends on the specific device behaviour, each device is related to a different set W_{dv}. We define the reward function as in equation 5.1.

$$R(s_i, a_j) = w_{i\to j} \tag{5.1}$$

Each time a weight $w_{i\to j}$ is updated, we perform a Softmax [125] on the subset of rewards associated with the set of actions that can be taken from state s_i, so as to compress the values into the range of [0,1]. All the values of such a subset after Softmax sum to 1, meaning that the function $R(s_i, a)$ has the property that

$$\sum_{\forall a \in A} R(s_{current}, a) = 1 \tag{5.2}$$

Periodically, the master performs an inventory phase in which it assigns an ID to each device to identify it in the communication protocol. This inventory stage

supports devices being added or removed from the environment. Once the master knows all currently active devices in the system, it places them in the state with the lowest requirement (i.e., s_n). Then the agent schedules transmissions in a frame slots. Transmission scheduling is ruled by a EDF-based mechanism that is explained in sec. 5.5. When a device is polled, it replies with a packet containing the last *data sample* and the value of a *counter of changes* representing the number of new samples generated since the last query.

When the master receives a packet from a device, it observes the counter of changes. If $counter = 1$ then the device has a new data sample since the last query, it means that polling frequency is appropriate. In this condition, the master communicates to the corresponding sub-agent to remain in the current state. If $counter = 0$ (i.e., the device does not have new data since the last query) or $counter > 1$ (i.e., the device has generated multiple new data instances since the last query) then the sub-agent is scheduling transmissions with too high or too low frequency. The sub-agent should change state, requiring a different polling frequency. Then the master asks (in a non-mandatory way) the sub-agent to move to a higher or lower state (i.e., a state with a higher or a lower frequency).

In case counter=0 then the sub-agent is asked to move to a lower state belonging to a subset $S_{allowed} \subset S$ of possible states. For example, for a device dv with counter=0 and transmission rate corresponding to s_2, the master asks the sub-agent, in a non-mandatory way, to move to a state with a lower requirement, e.g., a state in the set $S_{allowed} = \{s_3, s_4, \ldots, s_n\}$. Instead, for a device dv with counter>1 and transmission rate s_4, the set of allowed states is $S_{allowed} = \{s_1, s_2, s_3\}$.

Devices can move to a state in $S_{allowed}$ or remain in the current one. How does a device choose the next state? The decision is probabilistic. The probability to move to a state $s_i \in S_{allowed}$ is given by the reward $R(s_{current}, a_i)$, where a_i is the action of moving to s_i. The outgoing edge with the highest reward has the highest probability to be selected. For each node there is also a self-loop edge, and the reward associated to this edge is given by the formula in equation 5.3.

$$R(s_{current}, a_{current}) = \sum_{\forall s_j \in (S - S_{allowed})} R(s_{current}, a_{sj}) \tag{5.3}$$

To guarantee a wide exploration of states, with probability $1 - \epsilon$ the master selects the next state based on the reward mechanism, while with probability ϵ (which typically assumes a value around 0.1 [125]) the master randomly selects the next state among the allowed states and the current state. Thus, our device modeling thus presents two main properties: 1) it explores the space of the states; 2) it rapidly moves a device to the most appropriate state, jumping less promising states, guaranteeing fast reaction to changes.

There is one last point that remains to be addressed: how are rewards updated? The idea is to repay successful actions — corresponding to edges that bring devices into states in which their counter remains unitary — and penalize those actions that bring devices in states in which their counter becomes 0 or greater than 1. To quantify this idea, let us suppose at time t device dv is in state s_i and takes action a_j (i.e., moves from state s_i to state s_j, following the edge $e_{i \to j}$ that has weight $w^t_{i \to j}$). After the device performed the action, at time $t + m$, the weight $w^{t+m}_{i \to j}$ is updated as in equation 5.4.

$$w_{i\to j}^{t+m} = w_{i\to j}^{t} + \alpha * return \tag{5.4}$$

where α is a *weighting factor* and *return* is the reward corresponding the execution of action a_j, and has been empirically evaluated as 2 if $counter = 1$, -0.2 otherwise.

5.5 ReLEDF: Scheduling Transmissions

We now describe how the reader assigns the transmission channel. Devices' access to the channel is based on a time division mechanism (TDMA) controlled by the reader. Time is divided into frames, and each frame is divided in slots. All slots have the same length, and the reader signals the beginning of a new slot by sending a query containing the ID of the device that has to answer next. The reader exploits the feedback from the learning mechanism to schedule transmissions: the aim is to assign slots to devices based on their communication requirements, i.e., how frequently they need to transmit data.

An efficient way to coordinate the polling process is through a dynamic real-time scheduling algorithm, following a single processor approach (we have a single reader controlling the system). In particular, we adopt a Earliest Deadline First (EDF) algorithm, as it allows to optimally schedule device transmissions, guaranteeing deadlines (i.e., periods). The idea is to map devices polling to tasks whose deadlines are based on devices' communication needs. This means that a device with a high transmission rate will have a short deadline, while a device with low transmission rate will have a longer deadline. Tasks are periodic and deadlines may cause a *mode change*: a joystick may need to be polled once per second while not in-use (only to check if a player has started using it) and hundreds of times per second while in use. Passing from the off state to the in use state, the joystick causes a change in the deadline of the related task, requiring a new scheduling and thus requiring a mode change.

Another aspect to be considered regards the size of data samples. Some devices need to send only a few bits, others instead need to send a large number of bits. In each slot it is possible to transmit a limited number of bits, depending on the encoding adopted. When a tag has long data to transmit to the reader, it fragments the information in multiple packets. Thus, in each slot, a single packet is transmitted. For this reason, the system can be considered as *preemptive*: even if a device needs multiple slots to send its data sample, its assigned slots do not have to be contiguous, as the reader is able to reassemble the information segmented in different packets.

5.5.1 Brief introduction to EDF

We now give a brief introduction to EDF and real time scheduling, while an extensive description can be found in [120].

EDF is a priority policy for scheduling tasks based on deadlines. A task τ_i consists in a minimum atomic executable entity of work and is characterized by a worst-case execution time C_i — the maximum amount of time within which the work has to be accomplished — and a time constraint, *deadline*, P_i — period within which the task has to be performed. Given a set of real-time tasks $\mathcal{T} = \{\tau_1, ..., \tau_n\}$

to be scheduled, at each time slot EDF will execute the task closest to its deadline. In this work a task represents a communication between two devices. To guarantee that each device will communicate at least once in a certain time, we consider only periodic tasks, i.e., their instance must execute regularly once per period. Moreover, we set their period equal to their deadline P_i. Furthermore, only synchronous tasks are considered, meaning that all their first instances are released at the same time, commonly considered time zero. Tasks executions are scheduled by EDF inside a *frame*, which is a sequence of time slots. In each slot only one task can be executed, and EDF decides which one. A task can be scheduled multiple times in a frame and each execution is called *job* of the task. Each job j is characterized by a *release time* r_j — defined as the point in time at which the job becomes ready to be executed — and a *deadline* d_j — defined as the point in time by which the job must be completed.

Table 5.1 reports the notation used in the rest of the chapter.

Table 5.1. Notation.

Notation	Description
MS	the master, a device dedicated to poll all other devices in the network.
DV	set $\{dv_1, ..., dv_n\}$ of devices which want to communicate with MS
TR	set $\{tr_1, ..., tr_m\}$ of rate requirements ordered by the highest to the lowest
L	dynamic mapping ($dv_i \in DV \rightarrow tr_i \in TR$) reflecting current device state
$\mathcal{T}$	a set $\{\tau_1, \tau_2, ..., \tau_n\}$ of current periodic tasks to be scheduled by the master.
τ_i	the i-th task
j_k^i	the k-th job of the i-th task
r_j	the release time of the job j
d_j	the deadline of the job j
$\mathcal{M}$	the set of mode changes
m_0	system initial mode
m_z	the z-th mode change
t_{m_z}	the time at which happens the z-th mode change
$\mathcal{T}^z$	set of real-time tasks after the mode change m_z
C_i^z	the execution time of the i-th task after z-th mode change
P_i^z	the period/deadline of the i-th task after z-th mode change

The set TR of rate requirements is defined as $\{tr_1 = 1/20, tr_2 = 1/50, tr_3 = 1/100, tr_4 = 1/200, tr_5 = 1/500, tr_6 = 1/2000\}$, where $1/x$ means once every x slots. This set has been defined empirically to include a reasonable set of values. However, other rates may be employed.

Regarding the mapping L, please note that at any instant of time the same requirement may represent the state of multiple devices, while other requirements may not represent any device.

A task τ_i, as described in [120], should have a name and a period. In our case there is a one to one correspondence between devices and tasks. The name of task τ_i is given by the *id* associated with the device dv_i and the period of task τ_i is the current requirement of device dv_i.

5.5.2 Mapping Learned Transmission Rates to Tasks

Each device $dv_i \in DV$ is linked in L to a specific transmission rate $tr_j \in TR$, depending on the device current behavior. For example at time t_1, three devices, $DV = \{dv_1, dv_2, dv_3\}$, may be mapped to the same transmission rate, $L = \{(dv_1 \rightarrow tr_1), (dv_2 \rightarrow tr_1), (dv_3 \rightarrow tr_1)\}$. ReLEDF performs an inventory phase at the beginning of each frame so as to detect new devices joining the network. As the behavior of the devices is not known, they are mapped to a medium transmission rate, i.e., $tr_{i/2}$, where i is the cardinality of set TR. Protocol operation will then allow for an eventually more appropriate transmission rate.

Transmission rates are used to create the related tasks that are scheduled by EDF. A task τ_i is defined by a name, that in our case is the *device id*; a period P_i; and an execution time C_i. Each device $dv_i \in DV$, with $(dv_i \rightarrow tr_j) \in L$, has an associated task τ_i with a unitary execution time ($C_i = 1$) and period $P_i = \frac{1}{tr_j}$. We consider unitary execution time as each task instance is executed in a single slot, due to the TDMA approach.

Given a set of tasks, EDF creates a schedule that satisfies all the tasks deadlines. It is demonstrated that if a feasible schedule exists, it will be found by EDF [120].

In case one or more devices change transmission requirements, the related set of tasks has to change deadlines. Let us suppose that at time t_1 the set of devices $DV = \{dv_1, dv_2, dv_3\}$ has mapping $L = \{(dv_1 \rightarrow tr_1), (dv_2 \rightarrow tr_1), (dv_3 \rightarrow tr_1)\}$, while at time t_2, the mapping changes in $L = \{(dv_1 \rightarrow tr_3), (dv_2 \rightarrow tr_1), (dv_3 \rightarrow tr_1)\}$. The device dv_1 could be a presence sensor that at time t_1 needs transmission rate corresponding to tr_1 because of the presence of many people, while it requires a lower transmission rate, e.g. tr_3, at time t_2 when few people are present in the environment. Then device dv_1 causes a mode change in the system, requiring EDF to produce a new schedule, as presented in Section 5.5.5.

5.5.3 Case Study

We show an example that serves as case study throughout the chapter. Let us consider a set $\mathcal{T} = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_8\}$ of 8 tasks with unitary execution time. For each task τ_i, EDF generates a set of jobs, each one identified by $j_k^i[r, d)$, where i represents the task, k the job instance, r the release time, and d the deadline. Figure 5.2 shows the jobs generated by EDF for each task in $\mathcal{T}$, and its corresponding period. Notice that, the jobs are generated according the Least Common Multiple (LCM) of tasks' periods, i.e., each task τ_i has exactly $\frac{LCM}{P_i}$ number of jobs in each frame. Figure 5.3 shows the scheduled frame which is composed of 20 slots according to the LCM. Time slots in the frame are indicated by t_s, where s is the index of the slot. Notice that EDF uses a priority scheduling policy based on deadlines, i.e., at any time the algorithm will execute the job closest to its deadline.

5.5.4 EDF Feasibility Analysis

We now present the main techniques to assess scheduling feasibility for a set of tasks $\mathcal{T}$, i.e., to establish whether it is possible to schedule all tasks such that they meet their deadlines. To this end we first introduce some definitions.
Given a set of real-time tasks $\mathcal{T}$, and an interval of time $[t_1, t_2)$, we define:

Figure 5.2. Jobs generated by EDF

Task	Period	Generated Jobs			
τ_1	5	j_1^1[0,5)	j_2^1[5,10)	j_3^1[10,15)	j_4^1[15,20)
τ_2	5	j_1^2[0,5)	j_2^2[5,10)	j_3^2[10,15)	j_4^2[15,20)
τ_3	5	j_1^3[0,5)	j_2^3[5,10)	j_3^3[10,15)	j_4^3[15,20)
τ_4	10	j_1^4[0,10)	j_2^4[10,20)		
τ_5	10	j_1^5[0,10)	j_2^5[10,20)		
τ_6	20	j_1^6[0,20)			
τ_7	20	j_1^7[0,20)			
τ_8	20	j_1^8[0,20)			

Figure 5.3. Initial Schedule

Time slot	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}
Job	j_1^1	j_1^2	j_1^3	j_1^4	j_1^5	j_2^1	j_2^2	j_2^3	j_1^8	j_1^6	j_3^1	j_3^2	j_3^3	j_2^4	j_2^5	j_4^1	j_4^2	j_4^3

- the *process demand*, as the computational time units required by the task instances (jobs) in that interval of time: $h[t_1, t_2) = \sum_{t_2 \leq r_k, d_k \leq t_1} C_k$.
- the *loading factor*, as the fraction of the interval needed to execute its jobs: $u_{[t_1,t_2)} = \frac{h_{[t_1,t_2)}}{t_2 - t_1}$

By means of these definitions it is possible to define the necessary feasibility condition for the scheduling algorithm. The condition requires that, for any given interval $[t_1, t_2)$ of the scheduled frame, the *loading factor* $u_{[t_1,t_2)}$ is not greater than 1 (i.e., the jobs can be executed within a fraction of the interval). We recall that for periodic tasks, the frame scheduled by EDF is composed of the minimum sequence of jobs such that it is possible to create all task instances, with release times and deadlines. Once the scheduled frame is created, the scheduling sequence continuously repeats. Thus, to verify the feasibility of a scheduled frame, we introduce the *absolute loading factor*, which is defined as the maximum of all possible intervals loading factor: $u = \sup_{0 \leq t_1 < t_2} u_{[t_1,t_2)}$.
Each set of real-time tasks is feasibly scheduled by EDF if *and* only if its *absolute loading factor* u is such that $u < 1$, according to Spuri Theorem [118].

Nevertheless, this feasibility analysis requires to check all the possible intervals $[t_1, t_2)$, requiring significant computational complexity. Liu and Layland [67] propose an optimized condition for feasibility under EDF for synchronous periodic tasks. Without loss of generality, they consider only the execution time (C_i) and period (P_i) of tasks in $\mathcal{T}$, without having to study each job in each possible period of the scheduled frame.

Corollary 1 (Liu and Layland). *Any set of n synchronous periodic tasks with processor utilization* $\mathcal{U} = \sum_{i=1}^{n} \frac{C_i}{P_i}$ *is feasibly scheduled by EDF if and only if* $\mathcal{U} \leq 1$.

Corollary (1) can be easily demonstrated as a consequence of Spuri Theorem, the formal proof is given in [120].

5.5.5 EDF with mode changes

Let us now consider the application of EDF in our dynamic environment. During the execution of a scheduled frame, it may happen that: i) a task changes its period (i.e., a device changes its transmission requirement); ii) new tasks enter $\mathcal{T}$ (i.e., a new device joins the network), and iii) one or more tasks leave $\mathcal{T}$ (i.e., devices leaving the network). All these events produce a mode change that has to be addressed.

EDF adopts one of the following two policies: 1) wait for the end of the current frame and then create a new schedule with updated tasks; 2) interrupt the current frame, create a new schedule and immediately start it. However both policies are inefficient in our context. The first one can introduce delays, i.e,. a task that has changed its period could wait long time before being executed as required. For example, if τ_i changes its period at the beginning of a schedule, from P_i^0 to P_i^1 s.t. $P_i^1 \ll P_i^0$, it could wait long before being executed with the new period P_i^1, due to other jobs left in the scheduled frame. The second policy can lead to starvation, i.e.,a task may never be executed. For example, assume that τ_i is a task placed at the end of the schedule, and there are some tasks at the beginning of the schedule which continuously change their periods, then the policy will continuously interrupt the current schedule to create a new one. Thus, as long as there are changes, τ_i will be postponed and never executed.

To overcome these limitations we propose a new scheduling approach based on EDF, able to reduce delays and avoid task starvation.

5.6 EDF-IJ: A new scheduling policy

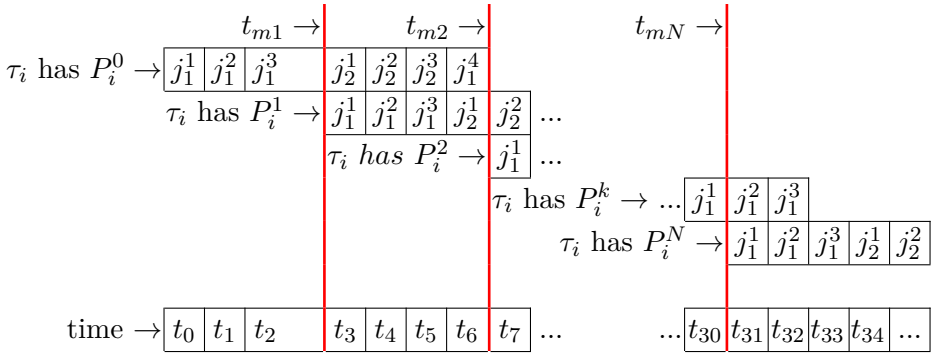
We propose a new scheduling policy, EDF with Inherited Jobs (EDF-IJ). The main idea is to immediately interrupt the current schedule when a mode change happens, and create a new schedule, keeping some information regarding the un-executed jobs of the interrupted frame. The quick creation of a new schedule allows fast changes and reduces delays, while the information about the previous one helps to avoid starvation.

Let $\mathcal{M} = \{m_0, m_1, m_2, ...\}$ be the set of possible mode changes during the execution, and let m_0 be the initial system mode. In EDF-IJ for any mode change m_i, an interrupt is generated, the current schedule is cut, and consequently a new schedule is produced.

Figure 5.4 shows an example of a frame that is interrupted, leaving some jobs un-executed; they should be rescheduled. Let t_{m_1} be the time slot in which the first mode change m_1 happens (i.e., first cut), and let t_{m_2} the time slot of the second mode change (i.e., second cut), and so on. We define P_i^0 as the period of task τ_i before the first mode change t_{m_1}, P_i^1 the period of task τ_i before t_{m_2} and after t_{m_1}, and so on.

For a mode change z, if the task τ_i changes its period then $P_i^{z-1} \neq P_i^z$. Instead, if a task τ_i does not change its period then $P_i^{z-1} = P_i^z$. In the new frame all tasks

Figure 5.4. Example of a frame that is interrupted, with un-executed tasks to be rescheduled.



should be scheduled keeping their periods, except for those which have caused the interrupt (as they have new periods).

To avoid starvation, the policy keeps some information regarding previous un-executed jobs. If these jobs are not scheduled by taking care of their old deadlines, it is possible to cause starvation: task instances placed at the end of the schedule will never be executed if they are preceded by jobs of tasks which continuously change their periods). Thus, we introduced the inherited deadlines. For each task that has not changed its period, the first un-executed instance after the mode change is inherited in the new schedule and, this instance, will have its deadline equal to the minimum between the inherited job deadline and the newly generated one. The inherited deadline is equal to the deadline of the first un-executed job (after the mode change) minus the time at which the mode change occurred plus 1.

In particular, the inherited deadlines are assigned to any first instance of $\tau_i \in \mathcal{T}$ s.t. $\exists\ j_j^i[r_j, d_j)$ not yet scheduled with $d_j > t_m \geq r_j$. Their inherited deadlines are computed using the following formula:

$$d_j = min(P_j, d_j - (t_{m_z} + 1)) \tag{5.5}$$

Only the first instances of tasks may have inherited deadlines; these jobs represent the inherited jobs from the previous schedule. For example, let us consider the beginning of a schedule and let us suppose there exists an instance of the task τ_i not yet executed at time slot t_{m_1}, the time at which a task changes its period and generates a cut in the schedule. Let P_i be the task period of τ_i, with $P_i^0 = P_i^1$. Thus, the *new schedule* will have as first job instance of τ_i (j_1^i), i.e.,*the inherited job*, with a release time 0 and deadline equal to (P_i - ($t_{m_1} + 1$)). This deadline is the minimum of the new generated deadline, P_i, and the inherited one, P_i - ($t_{m_1} + 1$). The next instances of τ_i in the *new schedule* will be normally scheduled according to P_i (i.e.,$[P_i, 2 \cdot P_i)$, $[2 \cdot P_i, 3 \cdot P_i)$ and so on).

The EDF-IJ scheduling policy is shown in Algorithm 2. Every time a mode change happens, the current schedule is interrupted and the algorithm provides a new one. Practically, it computes the job instances for all the tasks; checks the feasibility of the schedule (section 5.7) and, in case it is not feasible, it uses the DL-resolution algorithm (section 5.8) to solve the overload schedule problem; finally, it computes the scheduled frame by using a similar EDF policy.

Algorithm 2 EDF-IJ

Input: The current schedule, a set of real-time tasks $\mathcal{T}$, a mode change m;
Output: A feasible schedule
set $\mathcal{T}' \leftarrow \mathcal{T}$ after the mode change m
set $\mathcal{J} \leftarrow$ jobs set to schedule
for $\tau_i \in \mathcal{T}'$ **do**
    if $\tau_i \notin \mathcal{T} \vee P_i^m \neq P_i^{m-1}$ **then**
        $\mathcal{J} += \{j_1^i[0, P_i^m),\ j_2^i[P_i^m), 2P_i^m), ...\}$
    else
        if $\exists j_z^i[r_z, d_z)$ *not scheduled s.t.* $d_z > t_m > r_z$ **then**
            $d_z = min(P_z, d_z - t_m - 1)$
        else
            $d_z = P_z$
        end if
        $\mathcal{J} += \{j_1^i[0, d_z),\ j_2^i[P_i^m, 2P_i^m), ...\}$
    end if
end for
if not feasibility$(\mathcal{T}', \mathcal{J})$ **then** DL-resolution$(\mathcal{T}', Ld)$
end if
return EDF$(\mathcal{J})$

5.6.1 Example of the New Scheduling Policy

Let us consider the set of tasks $\mathcal{T}$ depicted in Figure 5.2, the schedule of Figure 5.3, and the occurrence of a mode change m_1 at the end of time slot t_{12}, induced by task τ_8 which changed period from 20 to 10. Then, EDF-IJ creates a table (see Table 5.5) where:

1. The first column contains the names of tasks in $\mathcal{T}$.

2. The second column indicates the new period after m_1. In this case, all jobs have the same period depicted in Figure 5.2, apart from task τ_8 which just changed.

3. The third column contains for each task the "inherited deadline". If there exists, a task "inherited deadline" is given by the deadline of the first un-executed job of such task (after m_1) minus the time at which m_1 occurred plus 1 (namely the time of the slot after the cut, t_{12+1}). Looking at Figure 5.3, after t_{12} the jobs we consider are: $j_4^1, j_4^2, j_3^3, j_2^4, j_2^5, j_1^6$. For example, τ_5 has a deadline of 20, m_1 occurs at t_{12}, so its inherited deadline is 20-(12+1)=7.

4. The fourth column shows the minimum of the second and the third column. In case the third column is empty, it considers the second column value.

Now EDF is able to generate the new jobs, as presented in Table 5.6. In particular, for the first instance of each task, the deadline will be the time reported in column "First Job Deadline" in Table 5.5. The new schedule is depicted in Table 5.7.

Figure 5.5. Inherited jobs table

Task	New Period	Inherited Deadline	First Job Deadline
τ_1	5	20-13=7	5
τ_2	5	20-13=7	5
τ_3	5	15-13=2	**2**
τ_4	10	20-13=7	**7**
τ_5	10	20-13=7	**7**
τ_6	20	//	20
τ_7	20	20-13=7	**7**
τ_8	**10**	//	**10**

Figure 5.6. Jobs generated by EDF after cut0

Task	Period	Generated	Jobs		
τ_1	5	j_1^1[0,5)	j_2^1[5,10)	j_3^1[10,15)	j_4^1[15,20)
τ_2	5	j_1^2[0,5)	j_2^2[5,10)	j_3^2[10,15)	j_4^2[15,20)
τ_3	5	j_1^3**[0,2)**	j_2^3[5,10)	j_3^3[10,15)	j_4^3[15,20)
τ_4	10	j_1^4**[0,7)**	j_2^4[10,20)		
τ_5	10	j_1^5**[0,7)**	j_2^5[10,20)		
τ_6	20	j_1^6[0,20)			
τ_7	20	j_1^7**[0,7)**			
τ_8	10	j_1^8**[0,7)**	j_2^8**[10,20)**		

Figure 5.7. Initial Schedule

Time slot	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}
Job	j_1^3	j_1^1	j_1^2	j_1^4	j_1^5	j_1^7	j_1^8	j_2^1	j_2^2	j_2^3	j_3^1	j_3^2	j_3^3	j_1^6	j_2^8	j_4^1	j_4^2	j_4^3	j_2^4	j_2^5

Whenever a task changes its periods, and before starting a new schedule, the policy must assert the feasibility (or un-feasibility) of the new tasks set $\mathcal{T}$ under EDF and, moreover, it must be aware of inherited jobs and deadlines.

5.7 Feasibility Analysis

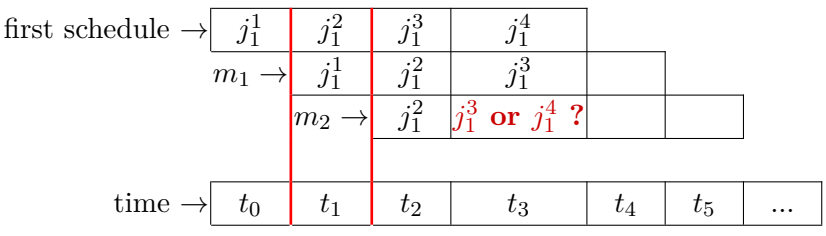
The new scheduling policy requires a feasibility analysis after each mode change. As tasks can change period, it may happen that the new periods do not allow the creation of a feasible schedule. For example, let $\mathcal{T} = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ be the task set s.t. all of them have an execution time 1 and a period P=4. If τ_1 reduces its period $4 \Rightarrow 3$ then $\mathcal{U} = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{3} = \frac{13}{12} > 1$, and thus no feasible schedule exists according to Corollary 1.

Furthermore, as presented in section 5.6, the new proposed scheduling policy considers also inherited deadlines from the previous schedule. Even if these deadlines affect only the first instance of some tasks they could make the feasibility analysis (Corollary 5) for EDF not applicable in our case.

Figure 5.8. Example in which Corollary 1 is not applicable

Task	**First schedule**		**1° mode change**		**2° mode change**	
	Period	Jobs	Period	Jobs	Period	Jobs
τ_1	4	$j_1^1[0,4)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
τ_2	4	$j_1^2[0,4)$	4	$j_1^2[0,\mathbf{3})$	**2**	$j_1^2[0,2)$, $j_2^2[2,4)$
τ_3	4	$j_1^3[0,4)$	4	$j_1^3[0,\mathbf{3})$	4	$j_1^3[0,\mathbf{2})$
τ_4	4	$j_1^4[0,4)$	4	$j_1^4[0,\mathbf{3})$	4	$j_1^4[0,\mathbf{2})$
$\mathcal{U}$	**1**		**3/4**		**1**	

Figure 5.9. Example in which Corollary 1 is not applicable



5.7.1 In-applicability of EDF Feasibility Analysis

In Figure 5.8 and Figure 5.9 we show an example in which Corollary 1 is not sufficient to provide schedule feasibility in presence of inherited deadlines. In this example $\mathcal{T}$ is composed of four periodic synchronous tasks: τ_1, τ_2, τ_3, τ_4, all of them with a period $P = 4$ and execution time 1.

After time t_0, in which j_1^1 is executed, the task τ_1 is excluded from $\mathcal{T}$ as it is not required anymore. Notice that, even if a task leaves $\mathcal{T}$ it is useful to spawn a mode change and generate a new schedule as it is possible to optimize the execution of tasks, especially in relation with the new overload scheduling policy presented in section 5.8.

After time t_1, in which j_1^2 is executed, the task τ_2 changes its period to a more demanding one ($4 \Rightarrow 2$). Thus, a second mode change (m_2) is spawned. The feasibility analysis, carried out using Corollary 1, shows that the task set $\mathcal{T}$ is feasible, as $\mathcal{U} = \frac{1}{4} + \frac{1}{4} + \frac{1}{2} = 1$. Instead, as the scheduling policy inherits also deadlines from the previous schedule for tasks τ_2 and τ_3, jobs j_1^2, j_1^3, and j_1^4 will have the same deadlines equal to time slot t_2, and thus the schedule will be infeasible. From this example it is possible to conclude that Corollary 1 is not enough in order to test the schedule feasibility for our proposal, and hence a new feasibility test for the proposed schedule policy is needed.

5.7.2 A New Feasibility Analysis for Inherited Jobs

As mentioned in the previous section, Corollary 1 is not enough to test the schedule feasibility when a mode change happens and we have some inherited jobs and deadlines. Thus, we first define the *initial process demand* and *initial loading factor* with inherited jobs and we propose a new feasibility test. We recall that, the job set IJ is composed of all the inherited jobs, i.e.,the first task instances in the new schedule with inherited deadlines.

Definition 2. Given a set of real-time tasks $\mathcal{T}$, a set IJ of inherited jobs, and an interval of time $[0, t_1)$, the *initial process demand with inherited jobs* is defined as the computational units required, by the task instances and inherited jobs, in the first interval of time up to t_1.

$$h_{IJ}[0,t_1) = \sum_{\substack{\tau_i \in \mathcal{T} \ s.t. \\ \nexists \ j_k^i \in IJ}} \left\lfloor \frac{C_i}{P_i} \cdot t_1 \right\rfloor + \sum_{\substack{\tau_i \in \mathcal{T} \ s.t. \\ \exists \ j_k^i \in IJ \ \wedge \ d_j < t_1}} C_i + \max\left(0, \left\lfloor \frac{C_i}{P_i} \cdot t_1 - C_i \right\rfloor\right) \quad (5.6)$$

Equation 5.6 is composed of two summations.

- The first summation regards only tasks without instances in IJ. For each of these tasks, the summation returns how many jobs they have with deadlines less than t_1. The floor function is used to comply with the equation on process demand, i.e., the fractional part of the sum argument represents a job that has a release time less than t_1 but a deadline greater than t_1.
- The second summation regards only tasks with an instance in IJ with deadline less than t_1. These instances have always deadlines less or equal their original

task deadlines, as they are inherited. For each of these tasks, we are counting one job (C_i) as the one from IJ with inherited deadlines, plus the number of their normal instances in this interval, minus the one inherited. Again, the floor function is used to comply with the process demand equation. Notice also that, the "minus C_i" is relevant only when $\frac{t_1}{P_i} \geq 1$.

Definition 3. Given a set of real-time tasks $\mathcal{T}$, a set IJ of inherited jobs, and an interval of $[0,t_1)$, the *initial loading factor with inherited jobs* is defined as the fraction of the interval needed to execute its jobs plus the inherited ones.

$$u_{IJ}[0, t_1) = \frac{h_{IJ}[0, t_1)}{t_1} \tag{5.7}$$

In Theorem 4 we propose a new feasibility analysis for sets of real-time tasks plus inherited jobs.

Theorem 4. *Each set of real-time tasks $\mathcal{T}$ with a set of inherited jobs, IJ, is feasibly scheduled by EDF-IJ if and only if:*

- *Corollary 1 is verified for $\mathcal{T}$, i.e., $\mathcal{U} \leq 1$*
- *$\forall j \in IJ$ the loading factor $u_{IJ}[0, d_j) \leq 1$.*

Proof. We prove that our Theorem satisfies the necessary feasibility condition for scheduling real-time tasks (Spuri Theorem 5.5.4).

The first condition guarantees that, in absence of inherited jobs IJ, all the intervals $[t_1, t_2)$ of the schedule have a loading factor $u_{[t1,t2)} \leq 1$, as direct consequence of Corollary 1 and Theorem 5.5.4. This is a necessary condition for the feasibility test also with inherited jobs. However, the inherited jobs could still overload some intervals and make the first condition not sufficient.

In particular, they can overload only intervals of the form $[0, d_j)$, where $j \in IJ$: as they can be only the first instances of tasks and, according to the definition of inherited deadlines, they can only be anticipated respect to the normal execution, they can overload only the intervals at the beginning of the schedule, up to their inherited deadlines.

Therefore, we show that the second condition can check only these intervals to complete the feasibility analysis for inherited jobs. In particular:

- It is not necessary to check any interval $[t_i, d_j)$, $i > 0$, because if there exists an overload in $[t_i, d_j)$ then also $[0, d_j)$ must be overloaded: as we are working with periodic synchronous tasks, i.e., they start together at time 0, by Theorem 3.14 of [120] it is not possible to have an idle time slot before an overload, thus any overload must involve also the slot 0.

- It is not necessary to check any interval $[0, d_j - i)$, $0 < i < d_j$, because an overload on it can be caused only by a job not in IJ, which would be already revealed by Corollary 1, or by another job in IJ with deadline different from d_j, which would be revealed by our feasibility analysis.

In summary: the first theorem condition will guarantee that, for all pairs $[d_j, t_k)$, s.t. $k > d \wedge j \in IJ$, the loading factor will be less than 1, as the inherited jobs will not affect them; the second condition of the feasibility analysis will guarantee that the loading factor in the intervals $[0, d_j)$, where $j \in IJ$, will be less than 1, by definition of initial loading factor with inherited jobs, equation 5.7.

□

5.7.3 Example of Application of the Feasibility Test on an EDF-IJ Schedule

Let us take again $\mathcal{T} = \{\tau_1, \tau_2, \tau_3, \tau_4\}$, all with the same period 4 and execution time 1. An example is depicted in Figures 5.8 and 5.9. At time t_0, IJ is empty. In this case, the EDF-IJ policy works exactly as EDF, as no instance should be "anticipated". The feasibility test is performed, following theorem 5.7.2, in the following way:

1. Verify Corollary 1 on $\mathcal{T}$. This job returns: $\mathcal{U} = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} = 1$.
2. As IJ is empty, no further actions are required.

A feasible schedule generated from $\mathcal{T}$ is shown in Figure 5.9, indicated as "First schedule".

After $t = 0$, task τ_1 leaves $\mathcal{T}$: a mode change (m_1) interrupts the current schedule. $\mathcal{T}^1 = \{\tau_2, \tau_3, \tau_4\}$ and in this case $IJ = \{j_1^2[0,3), j_1^3[0,3), j_1^4[0,3)\}$. As before, a feasibility test is required. The first condition of the test is easily verified as $\mathcal{U} = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} = \frac{3}{4} \leq 1$. In this case, as IJ is not empty, we should also verify that, for each deadline d_j, $j \in IJ$, $u_{IJ}[0, d_j) \leq 1$, as shown in Equation 5.8. Notice that all tasks in $\mathcal{T}^1$ have an inherited job in IJ.

$$u_{IJ}[t_0, t_3) = \frac{1 + max(0, \left\lfloor \frac{3}{4} - 1 \right\rfloor) + 1 + max(0, \left\lfloor \frac{3}{4} - 1 \right\rfloor) + 1 + max(0, \left\lfloor \frac{3}{4} - 1 \right\rfloor)}{3} \leq 1 \tag{5.8}$$

As both conditions are verified, there exists a schedule on $\mathcal{T}^1$ that is feasible, and this schedule is presented in Figure 5.9 indicated with m_1.

After t_1, task τ_2 changes its period, from 4 to 2. A new mode change (m_2) happens, $\mathcal{T}^1$ is updated becoming $\mathcal{T}^2$ and a new schedule should be created. As before, the feasibility test on $\mathcal{T}^2$ is performed. The first condition is verified, as $\mathcal{U} = \frac{1}{2} + \frac{1}{4} + \frac{1}{4} = 1$. As demonstrated in section 5.7.1 this test is not enough to guarantee the feasibility of the schedule with inherited jobs. Now $IJ = \{j_1^3[0,2), j_1^4[0,2)\}$. Notice also that as task τ_2 changed period, now two jobs are generated: $j_1^2[0,2)$ and $j_2^2[2,4)$. The second condition of the feasibility test must now be executed on current deadlines d_j, $j \in IJ$, as shown in Equation 5.9.

$$u_{IJ}[0, t_2) = \frac{\frac{2}{2} + 1 + max(0, \left\lfloor \frac{2}{4} - 1 \right\rfloor) + 1 + max(0, \left\lfloor \frac{2}{4} - 1 \right\rfloor)}{2} = \frac{3}{2} > 1 \tag{5.9}$$

As $u_{IJ}[0, t_2)$ is larger than 1, the feasibility test indicates that a feasible schedule does not exist under EDF. Effectively, as shown in Figures 5.8 and 5.9, in m_2 at t_3 both τ_3 and τ_4 instances have a deadline of 2, and must be executed, but there is space just for one job. This condition introduces an overload scheduling problem.

5.8 A New Overload Scheduling Policy

The literature proposes the following solutions to handle the overload scheduling condition [120]:

1. Best effort - each new task is added to $\mathcal{T}$, even if the schedule is not feasible: only tasks with shorter deadlines are executed.

2. Admission control scheme - if the new task makes $\mathcal{T}$ not feasible, the new task is refused.

3. Robust scheme - when a new task enters the set $\mathcal{T}$, some tasks are removed from $\mathcal{T}$ until it becomes feasible depending on another policy different from EDF (e.g. priority policy).

These solutions perform well for real-time systems. In case of overload, they attempt to carry out as much as possible critical jobs and dismiss those with higher period (according to the first approach), the new ones in the second approach, or those with lowest priorities (according to some priority policy). Instead, in a networking context, these solutions may have severe drawbacks, bringing an unfair channel allocation, in which some devices never communicate. While for safety critical jobs missing a deadline is not acceptable, for IoT systems such as a smart home would be preferable to miss a deadline rather than never communicate (missing a deadline introduces delay and data loss but does not compromise the system).

To solve this issue, we propose a new overload scheduling policy that accepts any change in the task set $\mathcal{T}$, and, if the new schedule is not feasible, it fairly rules the task periods to re-establish the feasibility without removing any task. As overload scheduling often arises when some tasks require to be executed more frequently, (i.e., require shorter periods but there are no enough free slots), the policy stretches the task periods to resolve the overload. When applied in a network context, such policy makes possible to communicate with all the devices.

Notice that each task period is closely related to the bandwidth allocated to the corresponding device. If a task has period P_i, the associated device can transmit up to $\frac{1}{P_i}$ rate, thus, as the task period increases the available bandwidth for the device decreases as well, causing data loss. In particular, if an algorithm for overload scheduling increases each period by 1, meaning that each task will miss its deadline by at most one slot, the subsequent data loss could be unfair. For example, given two tasks τ_1 and τ_2, with periods $P_1 = 2$ and $P_2 = 7$, which are supposed to be increased by 1 unit, the data loss of τ_1 will be around 33% while data loss of τ_2 will be around 13%. To overcome this problem the increment of task periods is done according to the possible data loss. Thus, our proposal increases periods while it guarantees a fair distributed loss among tasks. Notice that, the execution time of tasks cannot be reduced to resolve an overload scheduling because it is the minimum amount of time required to have a complete and meaningful execution of a task.

5.8.1 DL-resolution Algorithm

The overload scheduling policy is implemented in the **D**istributed **L**oss resolution algorithm (*DL-resolution*), shown in Algorithm 3, which takes as input a set of

tasks $\mathcal{T}$ and a parameter Ld, and, until the set of tasks is not feasible, increases the periods of some tasks. Only the periods of those tasks that maintain their data loss below a predefined threshold, namely *lossThreshold*, are increased. This threshold is updated at each iteration of the while loop by means of the input parameter Ld, which controls the *fairness* of the algorithm and its applicability in network context. By setting Ld to a small enough value it is possible to iteratively increase the task periods such that the tasks lose the same percentage of data. On the other hand, by setting Ld to 1, it is possible to increase all the tasks periods by 1, such that all of them may fairly miss deadlines by one slot, regardless the potential data loss in a network context.

Algorithm 3 DL-resolution

1: **Input:** A set of real-time tasks $\mathcal{T}$ and a loss data step $Ld \in (0, 1]$;
2: **Output:** A feasible set of real-time tasks $\mathcal{T}$
3: Set $\mathcal{T}' \leftarrow \mathcal{T}$
4: Map $\mathcal{I}_{\mathcal{P}} \leftarrow \{i : 0\}_{0 \leq i < |\mathcal{T}|}$
5: float lossThreshold $\leftarrow 0$
6: **while** $\mathcal{T}'$ is not schedulable **do** lossThreshold += Ld
7: **for** $\tau_i \in \mathcal{T}$ **do** $P'_i \leftarrow \mathcal{I}_{\mathcal{P}}[i] + 1$ expectedLoss $= 1 - \frac{P_i}{P'_i}$
8: **if** expectedLoss $\leq$ lossThreshold **then** $\mathcal{I}_{\mathcal{P}}[i]$ += 1 $\mathcal{T}'.update(\tau_i, P'_i)$
9: **end if**
10: **end for**
11: **end while**
12: $\mathcal{T} \leftarrow \mathcal{T}'$

In summary, the proposed overload scheduling policy always allows scheduling all input tasks by increasing their periods, and, by means of the input parameter Ld, which controls the data loss threshold, it is possible to apply either a data loss fair policy, or a fair deadline miss policy.

5.9 ReLEDF Algorithm

The ReLEDF algorithm combines the learning mechanism with the EDF based scheduling and device polling.

The main idea is that the master keeps polling devices until a change happens in device transmission requirements. Each time a device is polled, its state in the behavioral graph G is updated depending on its counter value. After such update, if the device remains in the same state, the master keeps polling following the same schedule. Otherwise, the device model changes state allowing for a new transmission rate, and a new schedule has to be generated. The generation of a new schedule requires the creation of the *transition schedule* and, in case of overscheduling, the application of the *distributed loss* algorithm. After defining and executing the transition schedule, the master generates a schedule from L by following the standard EDF policy. The pseudocode presenting the most significant steps is given in Algorithm 4.

At the beginning, the algorithm declares the master MS, the set of devices DV,

Algorithm 4 ReLEDF Algorithm

```
1: Input: Variables names refers to Sec. 5.5 and Sec. 5.9;
2: Master MS;
3: Set DV;
4: Schedule S_current;
5: Map L;
6: DV = MS.obtainDevices()
7: for dv ∈ DV do
8:     L.add(dv,new DevModel())
9: end for
10: S_current = MS.schedule(L)
11: Bool changes
12: while true do
13:     queriedDevice,ChangesCounter = MS.query()
14:     changes = L[queriedDevice].update(ChangesCounter)
15:     if changes then
16:         Set IJ = MS.getStarvJobs()
17:         overSched, S_current = MS.schedule(L,IJ)
18:         while overSched do
19:             distributeLoss(L)
20:             overSched, S_current = MS.schedule(L,IJ)
21:         end while
22:     end if
23: end while
```

Table 5.2. Scenario descriptions: each scenario includes a different number of sensors of different types.

	Experiments				Simulations							
Case / Sensor	E1	E2	E3	E4	S1	S2	S3	S4	S5	S6	S7	S8
Environmental	2	4	10	12	37	35	34	32	32	32	80	120
Remote	1	2	3	4	2	3	3	4	4	4	8	18
JoyStick	1	1	2	3	1	2	3	4	4	4	8	8
Camera	0	1	1	1	0	0	0	0	1	2	4	4
Total	4	8	16	20	40	40	40	40	41	42	100	150

the running schedule $S_{current}$, and the mapping L. Then the algorithm, through the *MS.obtainDevices* method, starts collecting the IDs of connected devices, assigned through a preliminary inventory process. For each device, the master creates the corresponding sub-agent, to which will be assigned the lowest transmission rate state (hence the first frame will have slots equally assigned to each device). After setup, the algorithm generates a new schedule (*MS.schedule(L)*) and starts polling active devices, through the *MS.query* method. For each poll, the algorithm asks the sub-agent corresponding to the polled device to update its state, depending on the device counter value. If the sub-agent remains in the current state, the algorithm polls the next device in the schedule. If the sub-agent changes state, the algorithm creates a new schedule following the *Transition without Starvation Scheduling*, described in sec. 5.6. The method *MS.getStarvTasks* returns, for each device, the first unexecuted task instance in the schedule, if there exists. If the schedule that is generated by the updated L is feasible, the algorithm updates $S_{current}$ and starts polling. Otherwise, it applies the Distributes Loss policy, described in sec. 5.8, until it does not reach a feasible schedule. In the pseudocode we present only the most important instructions, omitting operations like the rescheduling after a "Transition without Starvation Schedule".

5.10 Performance Evaluation

In this section we evaluate the performance of our protocol ReLEDF through both simulations and experiments. First, we evaluate the effectiveness of the proposed EDF-IJ scheduling, comparing it with EDF and evaluating also the overload scheduling approach against other solutions. Then, we evaluate the overall proposed protocol, namely ReLEDF, in different settings. Finally, we present the results of real experiments.

5.10.1 Scenarios

Simulated scenarios represent a typical smart home, involving a number of smart devices that vary between 4 and 20 for the experiments and 40 and 150 for the simulations. We remark that the value 40 has been used also in [79]. Devices are heterogeneous, e.g., temperature, presence, remotes and cameras [79], and are shown in Table 5.2, where E1 to E4 indicate the experimental scenarios, while S1 to S8 indicate the simulated ones.

Devices are simulated through Markov chains, as described in [79]. Devices have a set of states (representing possible transmission rates of devices), and a set of transitions between states, where transition probabilities reflect the device probabilities of producing new data (e.g., sense a new value). If a device changes state, a new packet is generated and sent to the destination device. Let us consider the case of a temperature sensor, which generates a new data sample each time a significant change in the temperature is supposed to be sensed. We imagine that the temperature changes on average 4 times per hour (this value can be set as preferred), therefore the probability to change state should encode around 4 changes per hour. The traffic generator flips a made-up coin every 10ms and decides whether to remain in the same state, expressing no change in sensed data, or to change state, sending the new temperature packet. Thus the probability to pass from a value X to a value Y is defined as $P(change) = 4/(3600 * 100)$, where the value $3600 * 100$ is the number of flips per hour.

More complex devices, which present multiple outgoing edges from the same state, follow a similar logic. However, transitions between states are more frequent. Consider the case of devices that interact with the user, such as TV remotes or joysticks. They have variable behavior, as they produce a traffic burst when they are used but remain almost inactive when they are not used.

5.10.2 Metrics

We measure the following metrics:

- **Data Delay**, defined as the time between the generation of a new packet on a device and its collection by the master. In case a tag reads new data multiple times before being polled, the data delay is measured since the first undelivered data collection.
- **Packet Delivery Ratio**, calculated as the number of packets containing new samples that are received by the master over the total number of new samples packets generated by all devices.
- **Fairness**, expressed as the Jain's fairness index [53] calculated on the Packet Delivery Ratio.

5.10.3 EDF vs EDF-IJ: Simulation Results

EDF-IJ Effectiveness

We now present simulation results on the effectiveness of EDF-IJ. For comparison we consider two versions of EDF. The first one is based on a pure implementation of EDF, where in case of changes in the task set, the new schedule is issued only at the end of the current frame. The second one creates a new schedule at each task change, without considering any information regarding the interrupted schedule jobs. We indicate this second version of EDF as *EDF-SR*.

Figure 5.10 shows the results for data delay in a typical smart home scenario (about 40 smart devices), including a camera continuously streaming data (scenario S5). EDF-SR and EDF-IJ are comparable, resulting around 65% faster than EDF.

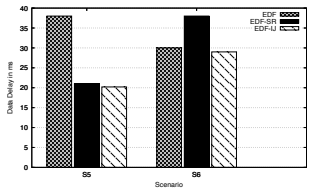


Figure 5.10. Data delay comparison between EDF, EDF with interruption (EDF-SR) and EDF-IJ.

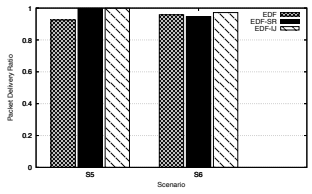


Figure 5.11. Packet delivery ratio comparison between EDF, EDF with interruption (EDF-SR) and EDF-IJ.

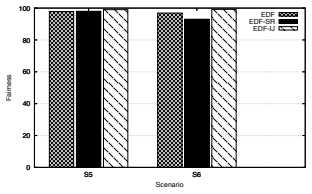


Figure 5.12. Fairness comparison between EDF, EDF with interruption (EDF-SR) and EDF-IJ.

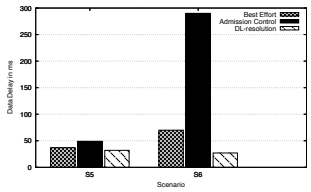


Figure 5.13. Data delay of three mode-change approaches under EDF.

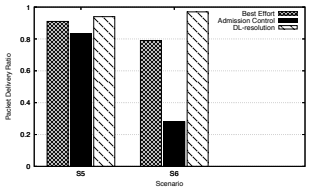


Figure 5.14. Packet delivery ratio of three mode-change approaches under EDF.

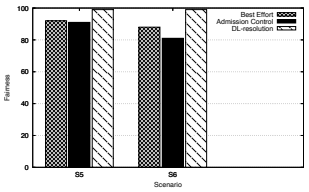


Figure 5.15. Fairness of three mode-change approaches under EDF.

This is because EDF is not very reactive to changes, and it is slower to schedule a new frame compared to the other two approaches. Increasing the number of cameras that continuously stream data (scenario S6) requires massive access to the communication channel, and introduces a significant number of overload scheduling events. In this case EDF-IJ is more efficient than EDF-SR. The superiority of EDF-IJ becomes more evident when looking at packet delivery ratio (see Fig. 5.11) and fairness (see Fig. 5.12): not only does EDF-IJ guarantee high delivery ratio in both cases (above 97%), but it is also more fair (0.99) than the other policies (0.98 and 0.92 respectively for EDF and EDF-SR in Case S6). Thus, our first set of results clearly show the benefits of EDF-IJ.

Resiliency to Overload Scheduling

We now investigate the resiliency to overload scheduling of DL-resolution (it equally distributes loss of data), against common approaches. As explained in section 5.8 there are three approaches to address overload scheduling: Best Effort, Admission Control, and a Robust scheme. For comparison we consider the first two, because the third one is not applicable to our context, as we do not have a priority list. The goal of the evaluation is to investigate the impact of DL-resolution of protocol performance in terms of data delay, data loss and fairness (we are interested in communicating with all the devices in an optimal but fair way) than the baselines.

Results are depicted in Figures 5.13, 5.14 and 5.15. For all metrics DL-resolution performs significantly better than the other two approaches, increasing its effectiveness in the more complex scenario. S6 represents a "saturation scenario", as the number of cameras continuously streaming data introduces a significant number of

overload scheduling events. DL-resolution not only makes device communication faster (in S6 it takes on average only 28ms, while the delay for the Admission Control is nearly 300ms), but also reduces data loss (0.04 against 0.21 and 0,71 of Admission Control and Best Effort in S6). In any case, DL-resolution results in more fairness than the other two approaches (see Fig. 5.15), as it always achieves 0.99 of fairness independently of the scenario. Thus this second set of results clearly show the benefits of using our DL-resolution to handle overload scheduling.

5.10.4 ReLEDF: Simulation Evaluation

We now turn to a thorough simulation evaluation of ReLEDF. We compare our protocol with APT-MAC [79], which is the closest to our work. To the best of our knowledge APT-MAC is the only solution so far proposed to address communication for battery-free devices. We also compare to the optimum and TDMA. The optimal solution, called OPT, is abstract as it always knows the best action to perform (i.e., which device to query next to avoid any data loss). TDMA is a baseline strategy that assigns slots in a round-robin TDMA schedule.

For workloads we consider again typical smart-home scenarios, involving around 40 heterogeneous smart devices (temperature, presence, remote, joystick, camera) [79]. However to investigate scalability we consider also larger networks (up to N=150 devices).

Furthermore, we consider a wide range of possible uses of devices (S1 to S8). We have performed actual experiments with real-devices. The simulations use parameter values such as frame rates of the cameras that match these real devices. The simulations are also validated by these experiments. Further, Table 5.2 shows for each of the seven simulations how many environmental, remotes, joysticks and cameras are simulated. We also chose to simulate 40 devices to represent a reasonable current generation smart home and 150 devices to demonstrate scaling for larger buildings. Complexity increases from S1 to case S8 as the number of burst devices increases.

Learning Time

The evaluation first looks at the effectiveness of our learning components, studying how fast ReLEDF learns and reacts to device behavior. Fig 5.16 shows the results on the time taken by ReLEDF to learn the required transmission rate for a joystick since it is switched on. For simplicity we indicate the relative time on the graph, meaning that the joystick activation occurs at relative time 0. When the joystick is activated for the first time, it takes the learning components a bit more than $1s$ to understand that the joystick is on, and around $4.5s$ to learn its communication requirement (i.e., tr_6), that is a reasonable time at system start up. At the second activation instead the system achieves the proper requirement much faster, as it takes only $0.5s$ to reach state tr_6, that is short enough for practical usage (see Fig. 5.17). Thus our first results clearly show the benefits of using a RL based approach, which quickly learns device behavior.

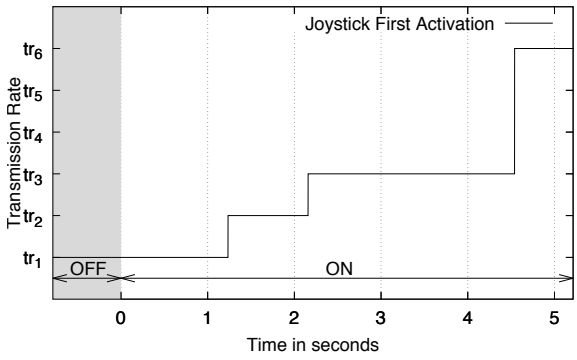


Figure 5.16. First activation of a joystick.

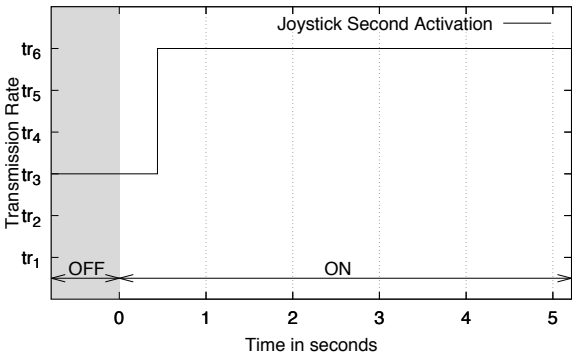


Figure 5.17. Second activation of a joystick.

System Efficiency

The second set of simulations investigates system efficiency. In terms of speed, Fig. 5.18 shows data delay for a variable number of devices. Our ReLEDF outperforms APT-MAC taking only from $26,87ms$ (case 1) to $39,8ms$ (case 8), to deliver a new data sample, resulting respectively 2 and over 3.5 times faster than APT-MAC in cases 1 and 8. ReLEDF is very close to the OPT data delivery, performing only from 2.3 times slower in case 7 to 3.9 times in case 2. This performance trend is confirmed also when looking at the packet delivery ratio. As shown in Fig. 5.19, ReLEDF delivers 99% of newly generated packets independently of the simulated case, being very close to the OPT that always achieve 100%, while APT-MAC loses 5% of new data in case 4 and up to 53% in case 8. APT-MAC clearly suffers when cameras are present in the environment (cases 5 to 8). TDMA loses more than 50% of new sample data, successfully delivering only $9,92\%$ of new data in case 8 and around 45% in the other cases. Increasing the number of devices (up to 150) does not have a significant impact on ReLEDF performance. Packet delay remains close to the optimum, reducing up to 6 times with respect to APT-MAC. In case of 150 devices, ReLEDF spends only $52ms$ to deliver new data, against $316ms$ for APT-MAC. Packet delivery ratio also remains high, resulting over 95% for up to 150 devices. In contrast, APT-MAC packet delivery ratio drops to only 47% in the case of 150 devices, while it is below to 10% for TDMA. Thus ReLEDF has almost optimal efficiency in assigning channel access for heterogeneous devices. These results are important in that they show that even ReLEDF's centralized solution is quite scalable.

5.10.5 ReLEDF: Experimental Evaluation

The goals of our experiments are twofold: i) to investigate the real applicability of ReLEDF; and ii) compare performance of ReLEDF with that of baseline TDMA and state of art protocols like APT-MAC [79]. To this end we implemented ReLEDF, APT-MAC, TDMA, and OPT on prototype devices and run experiments. Before presenting experimental results we give a brief description of our testbed.

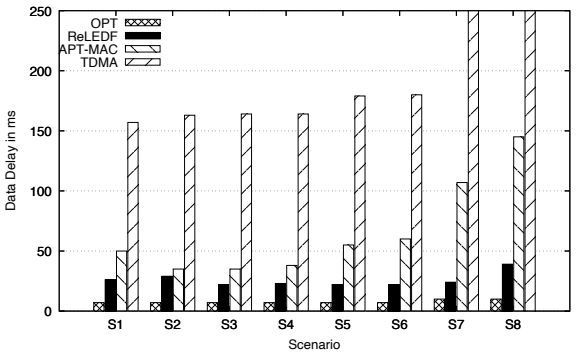


Figure 5.18. Packet delay for 40 devices by varying types of devices.

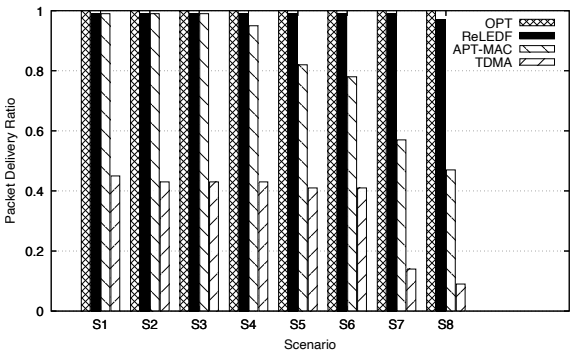


Figure 5.19. Packet Delivery Ratio by varying types of devices.

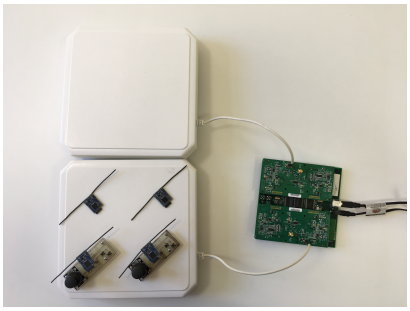


Figure 5.20. Implementation devices: four Moo Tags, two 860-910 MHz antennas, a USRP Reader and two Flex 900 daughterboard.

Testbed

Figure 5.20 shows our testbed, which is composed of: i) an USRP RFID reader, running RFID Gen2 Reader protocol developed by Buettner [23], and equipped with two 860-910 MHz antennas; and ii) four smart devices — a temperature sensor, a presence sensor, a remote, a joystick — realized leveraging sensor augmented RFID MOO tags [147]. Moo platforms are equipped with an ultra low power MSP processor, a temperature sensor and a series of input peripherals, like buttons and joysticks, and are powered only by the energy harvested from the RFID reader. The communication protocols (ReLEDF, APT-MAC, TDMA, and OPT) have been implemented introducing new primitives to the EPC Gen 2 [12], which is the standard protocol for RFID identification. EPC issues frames of slots, whose length has been empirically estimated, observing the effect of communication errors or interference (all packets are validated through a CRC). During experiments we noticed that the packet error rate changes depending on the slot length: with $6ms$ slots, the system reaches a $18,9\%$ of packet error rate, with $15ms$ a $8,1\%$ while with $25ms$ gets lower than 2%. Although slots of reduced size have high packet error rates, the redundancy introduced by multiple queries makes ReLEDF work better with reduced size slots, reaching an optimum at $6ms$ per slot. For this reason we fixed the slot length at $6ms$.

Each MOO tag emulates multiple smart devices so that we can perform experi-

ments with a number of devices ranging from 4 to 20. Specifically, we use the traffic generators defined for the simulation environment to emulate the different devices. The Moo tags transmit data according to device models. In this way we can collect data on different environments, e.g., with 4, 8, 16 or 20 smart devices.

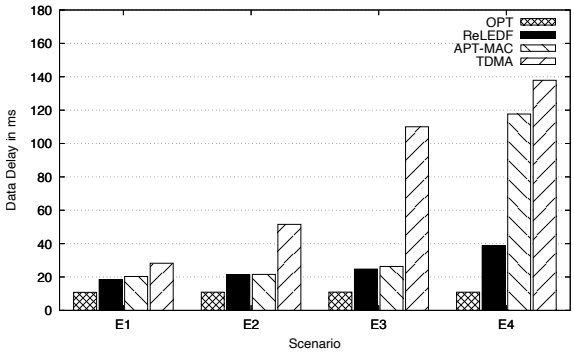


Figure 5.21. Experimental Packet delay by varying MAC protocol.

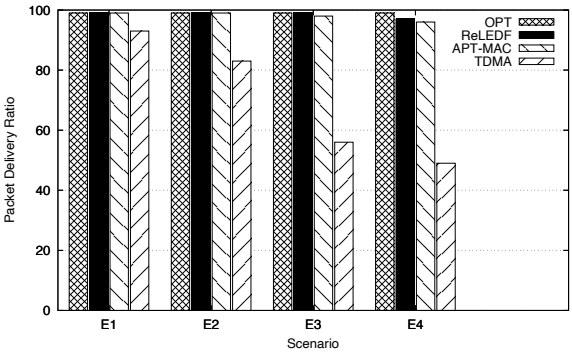


Figure 5.22. Experimental Packet Delivery Ratio by varying MAC protocol.

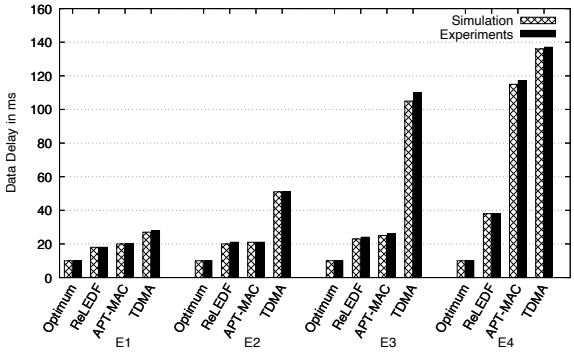


Figure 5.23. Data Delay comparison between MAC protocols (Optimum, ReLEDF, APT-MAC, TDMA) by varying the number of devices both in terms of Simulation and Experiments.

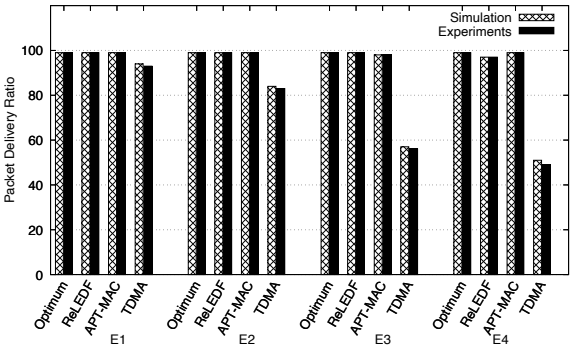


Figure 5.24. Packet Delivery Ratio comparison between MAC protocols (Optimum, ReLEDF, APT-MAC, TDMA) by varying the number of devices both in terms of Simulation and Experiments.

Results

We evaluated Data Delay and Packet Delivery Ratio while changing the number and the type of connected devices (see E1 to E4 scenarios in Table 5.2). Results are averaged over 100 repetitions.

Figure 5.21 shows that all protocols are very fast in delivering data, taking less than $40ms$ in scenario E1, when there are few burst devices (only one remote and one joystick). Introducing a camera and increasing the number of burst devices (1 joystick and 2 remotes in E2, and 2 joysticks and 3 remotes in E3) ReLEDF and APT-MAC keep low the data delay (below $25ms$), while TDMA is more than 4

times slower, taking on average $110ms$ to deliver data in E3. The advantage of using ReLEDF becomes evident when the number of burst devices further increases (i.e., 1 camera, 3 joysticks and 4 remotes in scenario E4), as it takes ReLEDF on average only $39ms$ to deliver data. Indeed, this time is only one third of the time taken by APT-MAC in the same scenario.

The good performance of ReLEDF is confirmed by results on packet delivery ratio (see Figure 5.22), as it is able to deliver above 97% of new data, independently of the scenario. Thus ReLEDF happens to be practical for smart homes with up to 200 devices.

Validation

We validated our simulations with experiments in the following way. We simulate the experimented scenarios, i.e., E1 to E4. Figure 5.23 shows the data delay for the four protocols. It is clear that the two evaluation techniques show comparable results, as simulation produces results that deviate by about only 4.7%, independently of the scenario. The experimental results of Fig. 5.23 are the same as Fig. 5.21. This outcome is confirmed by Figure 5.24, where performance difference between experiments and simulations in packet delivery ratio is below 4.8%.

Energy Evaluation

We now evaluate the energy impact of ReLEDF. As the whole protocol runs on the reader, the protocol does not have any effect on the energy consumption of battery-free devices. Hence we estimate the energy consumption of the reader, which is a wired-powered device. We measured the energy consumed by our reader, and quantified its consumption in a year. The reader employed in the experimentation has a power consumption of around 1.3A at 6V, for around 8W. Publicly available data specify that the maximum cost for energy in the USA is less than 20 cents per KW/h. Given this data we can estimate the system consumption in around 70KW/h per year, for a total of less than 20$ per year, which is low. We also notice that, from an environment-friendly point of view, the reader represents the most convenient solution, as we don't have any waste of batteries, nor any pollution impact on the environment.

Finally we notice that as stated in [107] RFID technology should be deployed as part of a building's physical infrastructure, just like running water, lights, and heat. Hence, exploiting such readers, our proposed devices can work with the emitted energy, without requiring batteries, or other sources of energy. Hence, it is true that our system requires a continuously operating RFID reader but any smart environment involves the deployment of an RFID reader [85].

5.11 Conclusions

In this chapter we presented ReLEDF, a new communication protocol that enables the coexistence of multiple battery-free smart devices in the same smart building or home. ReLEDF is able to learn and monitor the current and dynamic behavior (depending on changes in the environment that involves what devices are currently

being used and what their current communication requirements are) of a variety of battery-free devices, and properly coordinate channel access, so as to satisfy heterogeneous device requirements. Our protocol exploits RL to build and update behavioral device models that represent device communication needs based on events detected in the environment, and a channel access method based on EDF scheduling, modified to deal with schedule transitions.

The mix of learning and scheduling is an innovative way to schedule the network, which gives high performance ensuring all devices meet deadlines without starvation. For this reason we propose a new mode-change scheduling approach (EDF-IJ), which reduces delays and avoids device starvation in scheduling transmissions. Our simulation results show the benefits of using ReLEDF: it outperforms state-of-the-art solutions both in data collection efficiency and effectiveness, allowing a high number of smart devices to communicate with an almost optimal packet delivery ratio. Finally, real experiments demonstrate the applicability of our protocol. The overall value of the solution is to support many IoT devices in a smart space without needing batteries.

After this exploration of the link layer, in the next chapter we present a new tool for smart buildings network emulation.

Chapter 6

SMARTEEX: a software tool for SMART Environment EXperiments

6.1 Chapter Introduction

The connections required to run a smart building are double: a network one and a logical one. While two devices are connected through a wireless or wired link, they should be also connected in the logic *sensor-actuator*, meaning that in some way it should be defined what a sensor should provoke inside the building.
The research in logic connections, due to the emerging demand for smart building systems, has significantly stimulated the creation of management platforms for sensor and actuator networks. In the last few years, several solutions have been proposed for these kind of networks, spanning from research to commercial systems. However, single devices operate based on proprietary protocols, making interoperability a significant issue. To achieve a thorough understanding of the performance of these systems in field experiments are needed.
To tackle these issues we propose SMARTEEX, a software tool to test smart building solutions. SMARTEEX is the first proposal that provides a virtual network to test different architectures for smart environments, running experiments with real and/or simulated smart devices. Our performance evaluation demonstrates that using SMARTEEX it is possible to compare for example a cloud-based with an SDN-based architecture.

6.2 Related Work

Only recently, makers of building control systems have begun to adopt open protocols (e.g., LonWorks[69]) that allow all systems to communicate in a common protocol language. Nevertheless, system coexistence is still at an early stage, and it is not clear how smart devices can perform when operating in an IoT network: most of them are developed as stand-alone systems, without considering the impact of other devices — operating and coexisting in the same environment — on their performance.

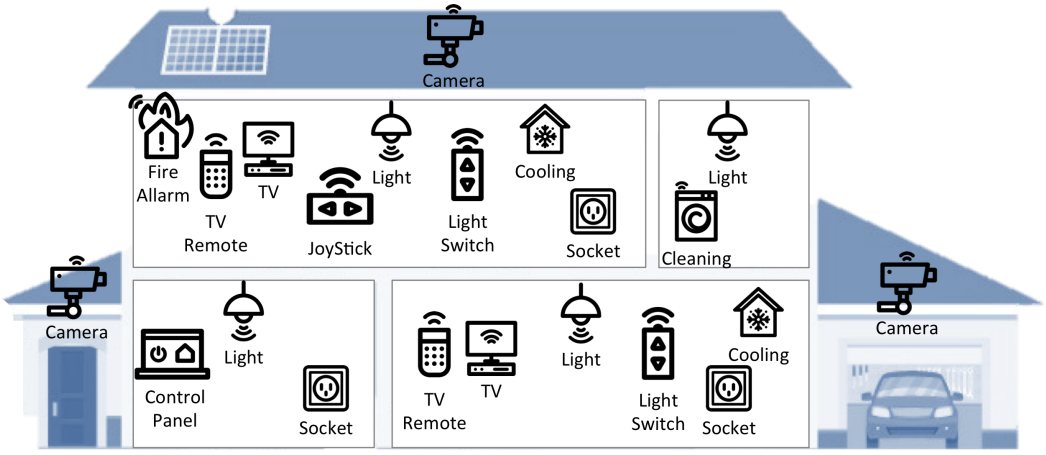


Figure 6.1. Example of smart home devices.

Current network simulators, like NS3 [25], are often too complex for testing single devices, and require significant work to simulate an entire building. The need for a simple and easy to use evaluation tool for smart building appears clear. The reason is twofold: first, to understand how a new device would operate inside a number of different complex smart building environments, and second to evaluate new network paradigms or protocols before their development.

In this chapter we propose SMARTEEX [81], a software tools that allows to test smart building solutions. Current systems presents different architectures, that may be centralized [56], cloud-based [47], SDN-based [75], etc. Using SMARTEEX, anyone willing to test its own architecture can easily set up a network and run experiments with real and/or simulated devices. Even new network architectures can be tested with our tool.

6.3 Smart Buildings

We consider the case of smart homes, which are outfitted with a myriad of devices including sensors -| buttons, switches, remotes, joysticks, presence, temperature, humidity, wet, smoke, and light sensors -| and acuators -| lights, shutters, cooking appliances, TV, air conditioners, heating systems, game consoles, alarms, music devices, and cleaning devices. See Figure 6.1 for a comprehensive set of smart home devices.

Smart home devices include sensors, such as buttons, switches, remotes, joysticks, presence, temperature, humidity, wet, smoke, and light sensors, and actuators, like lights, shutters, cooking appliances, TV, air conditioners, heating systems, game consoles, alarms, music devices, cleaning devices.

The main issue in smart environments is system management, especially in terms of device operation, communication, and interoperability. Let us consider the case of a smart room with a smart light switch (wireless), a gateway, and a smart bulb. What happens at the routing level when the user presses the switch? Data routing

depends on the system architecture. There are mainly four different approaches.

The first includes *traditional* home-automation platforms (e.g., Kblue [56]), where a local gateway, deployed inside the home, acts as (centralized) server and controls all devices. All the management logic is local to the home. In this case, the light switch sends a packet to the gateway, which in turn transmits the packet to the bulb. Once it has received the packet, the bulb performs the required action (i.e., switch on/off). This approach, frequent when internet connections were expensive and not reliable, is now in disuse.

The second approach concerns *cloud-based* platforms that implement the management logic inside the cloud and use a home local forwarding gateway to interface home devices with the remote server. Examples of popular commercial cloud-based systems are speakers like Google Home [47] or Amazon Echo [8]. These devices act as: 1) sensors, as they have an onboard microphone able to detect a user's voice; 2) gateways, as they can connect a number of devices to the cloud (e.g., Amazon Echo supports Wi-Fi and Zigbee), and 3) actuators, as they have onboard speakers able to communicate with users. In this case, the light switch sends the packet to the gateway, which in turn sends received data to the cloud. The cloud interprets the corresponding actuator command and sends it back to the proper gateway inside the home. The destination gateway will then forward the command to the specific actuator. Although this approach is widely adopted in recent commercial products, it introduces a number of critical issues, like significant dependence on the cloud -|making the system unusable in case of connectivity interruption -| and data privacy -| as data is processed outside the smart home.

The third approach combines the benefits of the previous two by proposing a smart gateway that is able to decide if received data has to be sent to the cloud or managed locally. Commercial systems currently available are connected bulbs [96], connected thermostat [88], connected cameras [89], connected washing machines [111], etc.

The fourth approach, besides combining cloud-based with local control, exploits Software Defined Networking (SDN) [43][140]. The controller that generates the logic is on the cloud and the rules modeling the logic are distributed through SDN routers inside the smart building. In this case, the first time a light switch is pressed, a new rule will be installed by the SDN controller on local SDN routers. Then, at the subsequent switch pressing, local routers already know how to route the switch information directly to the corresponding light actuator.

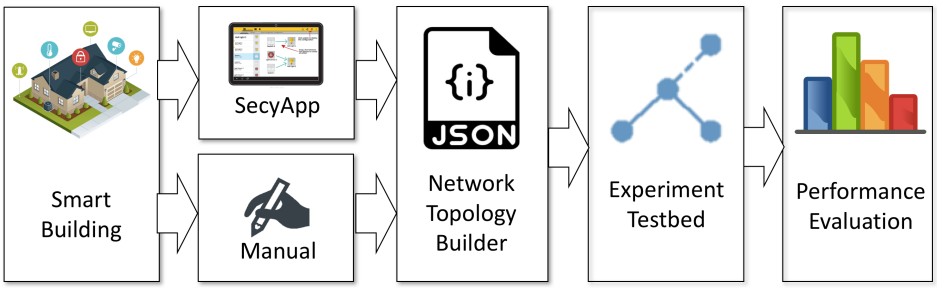


Figure 6.2. SMARTEEX architecture.

6.4 SMARTEEX

SMARTEEX [81] is a novel solution developed to test smart building solutions. SMARTEEX is the first proposal that provides a virtual network to test different architectures for smart environments. Using SMARTEEX, anyone willing to test its own architecture (cloud-based, SDN-based, combination of local and cloud-based, etc.) can easily set up a network and run experiments with real and/or simulated smart devices. Since we use Mininet [58] as a virtual network for experimenting with SDN systems, we now give a short introduction to it.

A brief primer on Mininet: Mininet is a tool able to create a virtual network of hosts on a single machine. Each virtual host runs its own real kernel and applications, making possible to emulate a network of devices in order to run experiments. Applications which run on Mininet hosts can send packets like on a real Ethernet interface, with a given link speed and delay. It allows for full network topology customization, both in terms of connection type and quality (e.g., throughput, packet loss, etc.). It is mainly used for emulations based on Software Defined Networks. Also, as the whole network is software simulated, it is easy to share and replicate experiments.

6.4.1 Architecture

The SMARTEEX architecture, shown in Figure 6.2, is composed of a main module, called the Network Topology Builder, which takes as input a smart environment scenario and produces as output a virtual network.

Starting from a smart building scenario, the user can define the set of smart devices manually or through applications like SECY [75], whose goal is to help in configuring and managing software-defined smart homes. The output of this first step consists of a list of network components — sensors, actuators, and routes — available in the smart environment, and their location within the environment — rooms, floors, etc.

The list of network components is then fed to the Network Topology Builder to create the *Smart Schema*, a file encoding the network topology of the smart environment.

Devices can be of three types: 1) routers, e.g., standard Wi-Fi routers connected to the Internet or sinks for other technologies; 2) sensors, i.e., devices which collect information from users or from the environment; and 3) actuators, devices which accomplish actions on the environment. Sensors and actuators can be real or simulated.

Algorithm 5 Device representation within the Network Topology Builder

```
1: {"cam1":
2:     { "type": "sensor",
3:       "ip":"10.0.0.1",
4:       "format": "simulated",
5:       "script": "java -jar trafficmodel.jar operation device dest time"
6:     }
7: }
```

Algorithm 6 Connection representation within the network Topology Builder

```
{
    "src": "f2.r4.rou4",
    "dst": "f2.r6.pre3",
    "pckt loss": "2%",
    "type": "wifi",
    "rate": "100mb",
    "latency": "10ms"
}
```

The Snippet in Algorithm 5 shows the representation of a simulated sensor — a videocamera — within the Network Topology Builder. In case the user wants to test real devices, he/she just has to set the `format` parameter to `real` inside the Smart Schema. In this case SMARTEEX produces a transparent host for the device to connect it to the Mininet network that acts as a transparent proxy to the generated network. If instead the user wants to evaluate a simulated device, he/she can use SMARTEEX's device simulator (see Sect. 6.4.2). Thus, each device is characterized by at least two parameters: `type`, indicating if the device is a sensor, an actuator, or router; `format`, specifying if the device is real or simulated. As additional optional parameters we also have: `ip`, in order to define a static IP for the device; `script`, to run a script on the host which represents the device.

The Network Topology Builder also represents connections between devices. Source and destination devices are addressed in the form `floorname.roomname.devicename`. The Snippet in Algorithm 6 shows the connection between a router located in the second floor, room 4 (named `f2.r4.rou4`) and a presence sensor located at the same floor in room 6 (named `f2.r6.pre3`). Each connection is characterized by a set of features such as source, destination, rate and packet loss.

With the structure obtained by the Network Topology Builder, SMARTEEX generates a set of hosts and switches, and connections between them. Figure 6.3 shows how the software works: from the network topology defined by the user and the set of real devices, it creates the virtual network that contains all the simulated devices and switches to which real devices can be attached. The script defined in the Network Topology Builder is executed on each of these devices, e.g., sensors start generating traffic and actuators start receiving and logging this traffic. While the network is running, a user is able to use all the Mininet client commands, e.g., access the shell of a specific host or run tests like "ping-all".

6.4.2 Device simulators

SMARTEEX provides a traffic generator useful to simulate a large variety of smart devices. This software, written in Java, can be executed by running the command:
`java -jar trafficmodel.jar operation device destination time`
where `operation` can assume the values: "generator", to generate traffic; "receiver", to log received packets only; and "forwarder", to receive packets and forward them to another address. `destination` is the ip address at which the traffic generator has

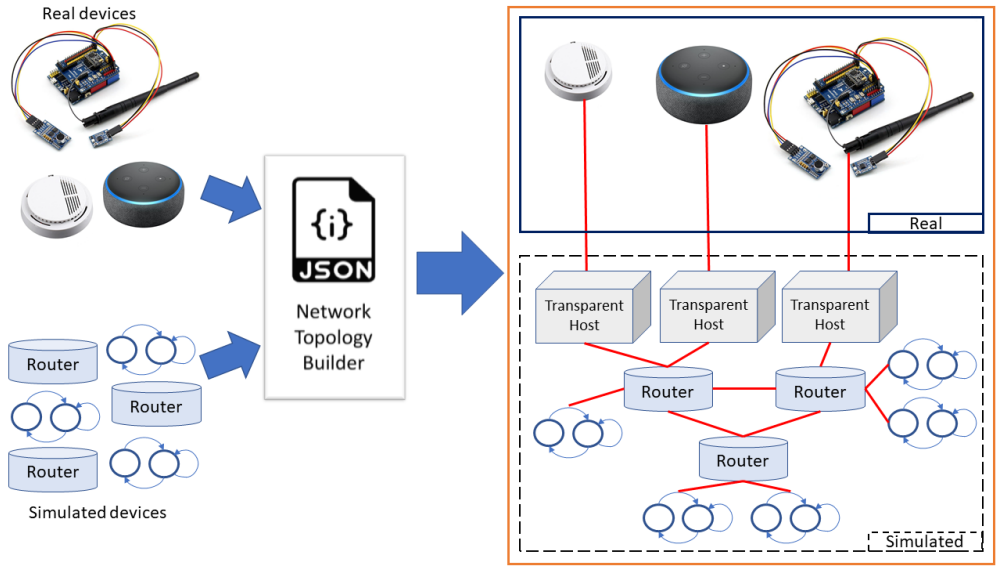


Figure 6.3. SMARTEEX generates a virtual network, to which it is possible to attach real devices, with the specification defined by the user with the Network Topology Builder.

to send the traffic; `time` is the test time in seconds; `device` represents the simulated device (e.g., camera, joystick, temperature, presence, remote).

Device simulators are defined as Markov chains, following the intuition described in chapter 4. They have a set of states (representing possible transmission rates of devices), and a set of transitions between states, where transition probabilities reflect the device frequency of producing new data (e.g., sense a new value). At each instant of time, SMARTEEX runs all device simulators, which may or not change state. If a device changes state, a new packet is generated and sent to the destination device.

Let us consider the case of a temperature sensor, which can be defined as represented in figure 6.4. This model generates a new data sample each time a significant change in the temperature is sensed. We imagine that the temperature changes on average 4 times per hour (this value can be set as preferred), therefore

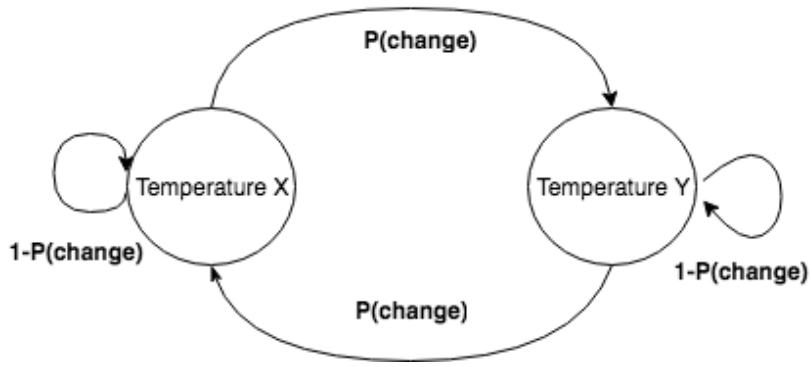


Figure 6.4. Temperature sensor model.

Table 6.1. Temperature Sensor Simulator - Transition Probabilities

	Temperature X	Temperature Y
Temperature X	1-4/(3600*100)	4/(3600*100)
Temperature Y	4/(3600*100)	1-4/(3600*100)

Table 6.2. Presence Sensor Simulator - Transition Probabilities

	No Presence	Short Presence	Long Presence
No Presence	0	10/(24*3600*100)	4/(24*3600*100)
Short Presence	1/(10*100)	0	0
Long Presence	1/(600*100)	0	0

the probability to change state has to encode around 4 changes per hour. Every 10ms the traffic generator flips a made-up coin and decides whether to remain in the same state, expressing no change in sensed data, or to change state, sending the new temperature packet. Thus the probability to pass from a value X to a value Y is defined as $P(change) = 4/(3600 * 100)$, where the value $3600 * 100$ is the number of flips per hour. Transition probabilities can be represented in a table as shown in Table 6.1 for the temperature sensor.

In more complex devices, where there are multiple outgoing edges from the same state, the traffic generator follows a similar logic, moving to other states with probability $P(otherState)$ and remaining in the same state with probability $P = 1 - (SumP(otherState))$.

SMARTEEX provides a simulator for different devices, such as presence sensors, TV remotes, joysticks, cameras. The probabilities related to the respective chains are reported in tables 6.2, 6.3 and 6.4.

6.5 Experiments

The purpose of this section is to show that the platform allows evaluating the performance of different architectures (e.g., cloud and SDN based). Goal of this comparison is the evaluation of the delay between the time at which an event occurs, e.g., a change in the temperature or a button pressure, and the corresponding action of the actuator on the environment, calculated as the time from the instant at

Table 6.3. Remote Simulator - Transition Probabilities
(T.D. = 24*3600*100)

	Not in Use	Conf. A	Conf. B	Not Press.
Not in Use	0	4/T.D.	4/T.D.	0
Conf. A	0	1/2	0	1/2
Conf. B	0	0	1/2	1/2
Not Press.	1/(5*100)	4/(5*100)	4/(5*100)	0

Table 6.4. Joystick Simulator - Transition Probabilities (T.D. = 24*3600*100)

	No Play	Playing	Conf.A	Conf. B	No Press.
No Play	1-3/(T.D.)	2/(T.D.)	0	0	0
Playing	0	0	1/2	1/2	0
Config A	0	0	1/3	1/3	1/3
Config B	0	0	1/3	1/3	1/3
No Press.	1/(T.D.)	0	0	0	1-1/(T.D.)

which a sensor sends a packet containing the request to the instant at which the corresponding actuator receives the packet. We compare a cloud-based system, in which all packets generated by sensors are first transmitted to a remote server that elaborates them and sends the relative commands to the proper actuators, and a SDN-based system, in which a remote controller installs rules on local routers, reducing the amount of traffic directed to the cloud (as proposed in [75]). In this approach, when a sensor generates a packet the first time, it is sent to a remote controller on the cloud. The remote controller, if possible, installs a routing rule on the local router, such that the next time the local router receives a similar packet from the same sensor, it already knows to which actuator inside the smart home the packet has to be sent.

6.5.1 Simulated Environment

The simulated environment, composed of two floors and three rooms per floor is depicted in Fig. 6.5, and the corresponding network topology is represented in Fig. 6.6. On floor 1 there is a kitchen (room1), which is even the home entrance; a family room (room2) with a sofa and a TV, and a bathroom (room3). On the second floor we have a corridor (room6); a bedroom (room4) and another bathroom (room5). In rooms 1, 2, 4 and 5 there are routers: router1 is connected to the Internet and to router 2 and 4, while router 5 is connected to router 4. All devices are connected to their room router, apart from devices in room3 which are connected to router 2. In each room there are lights, which in rooms 1, 3, 5 are controlled by buttons, while in rooms 2, 4 and 6 are controlled by three different presence sensors. Outside room1, Button2 commands a buzzer in the same room, and Camera1 streams video directly to TV1 in room2. Additionally, in Room2 there are Joystick1 and Joystick2 that are connected to TV1. In room4 the heating system heating1 is connected and managed by the temperature sensor temperature1, and in room5 speaker1 streams the music from an online service.

Cloud Configuration In order to obtain this configuration, we defined a server outside the smart home and with a delay of 150ms, as with an aDSL connection. All sensors inside the smart home send their packets to the cloud server, which then forward packets to corresponding actuators.

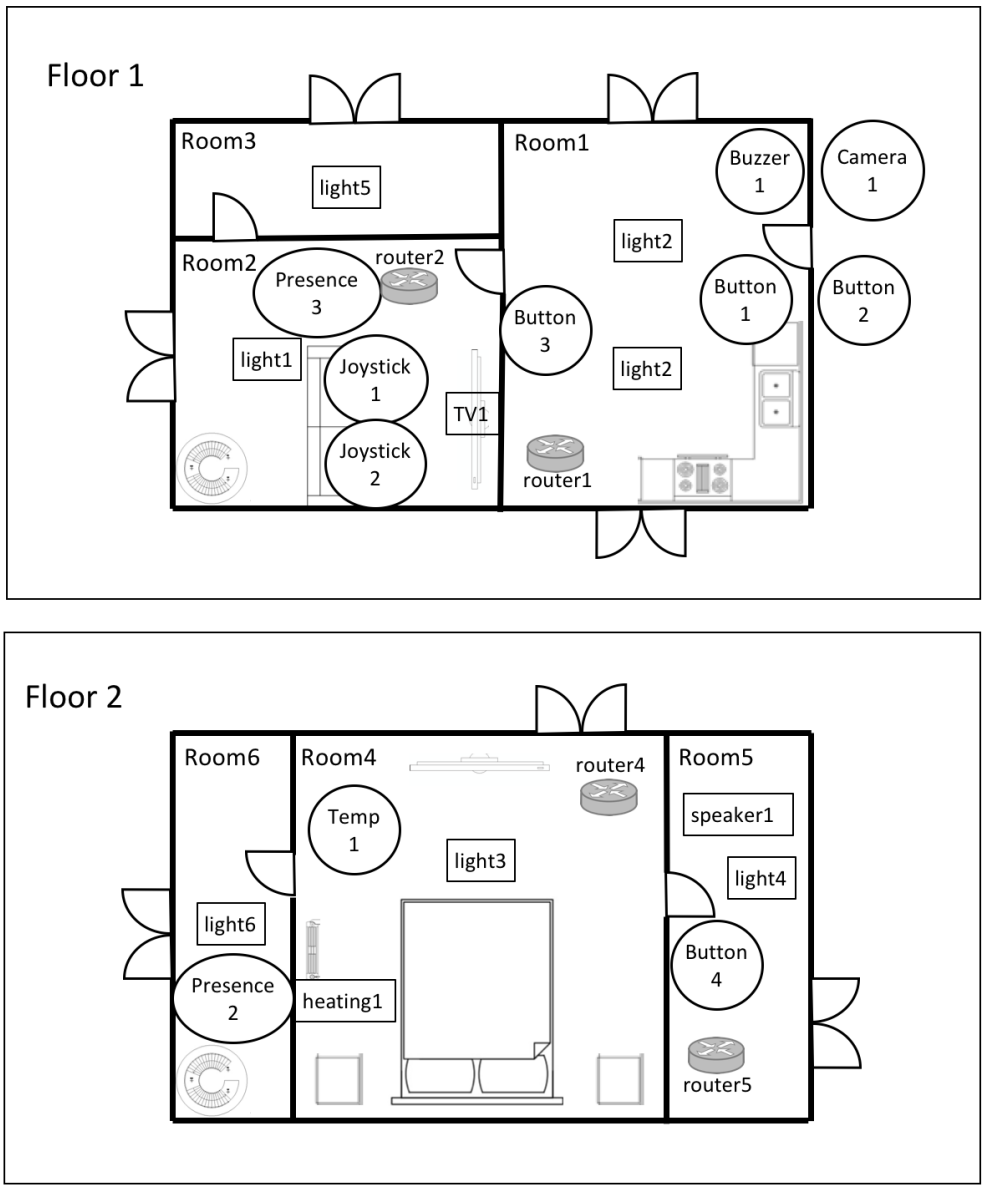


Figure 6.5. Map of the simulated smart building.

SDN Configuration A sensor with new data sends its packets to the local SDN router. If it does not have applicable rules, it sends the packet to a remote controller, which installs a new rule on the SDN router, depending on how the user has configured the smart building. The SDN router then is able to directly forward all the subsequent packets from the same sensor directly to the next hop inside the smart building, without sending them to the cloud.

Metrics SMARTEEX allows to measure a wide number of network characteristics, as it is possible to directly inspect every point of the simulated network. In this comparison, we evaluate the *delay*, defined as the amount of time between the

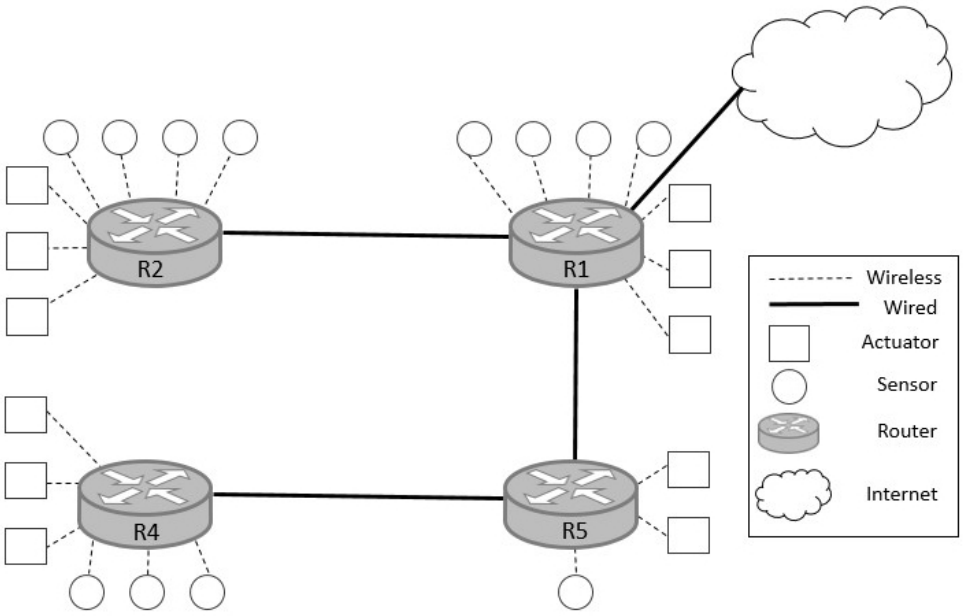


Figure 6.6. Topology of the simulated smart building.

generation of a packet containing a sensor data and its arrival at the proper actuator, and the *number of hops* traversed by a packet travelling from a sensor to the related actuator.

Workload In order to create the workload we exploit the device simulators presented in section 6.4.2. The devices simulated are those represented in figure 6.5. All the configurations regarding sensors, actuators, models and connections are available at SMARTEEX page [81]. All traffic generators have been running for 6 hours.

Results Figure 6.7 shows the delay incurred by applying the two different approaches (cloud-based vs. SDN-based). Regardless of the type of device, the delay incurred with the SDN-based approach is much shorter than that experienced with a cloud-based architecture. In particular, the SDN-based approach is able to perform nearly 10 times faster than the cloud-based solution: for example a packet that flows from a camera to a TV employs around 160ms with the SDN and 1885ms with the cloud-based solution. This result may appear obvious, because with the SDN-based approach all packets remain inside the home, but SMARTEEX allows to measure the difference between the two approaches by considering different physical aspects: in some geographical areas the cloud-approach may result comparable with the SDN based approach, while in others the gap may become significant, due to less powerful Internet connection.

The advantage of using a SDN-based approach is evident also when looking at the average number of hops traversed by packets when leaving from a sensor and arriving to the corresponding actuator (see Figure 6.8). In this case, the SDN approach improves local routing by using the best path for connections between two

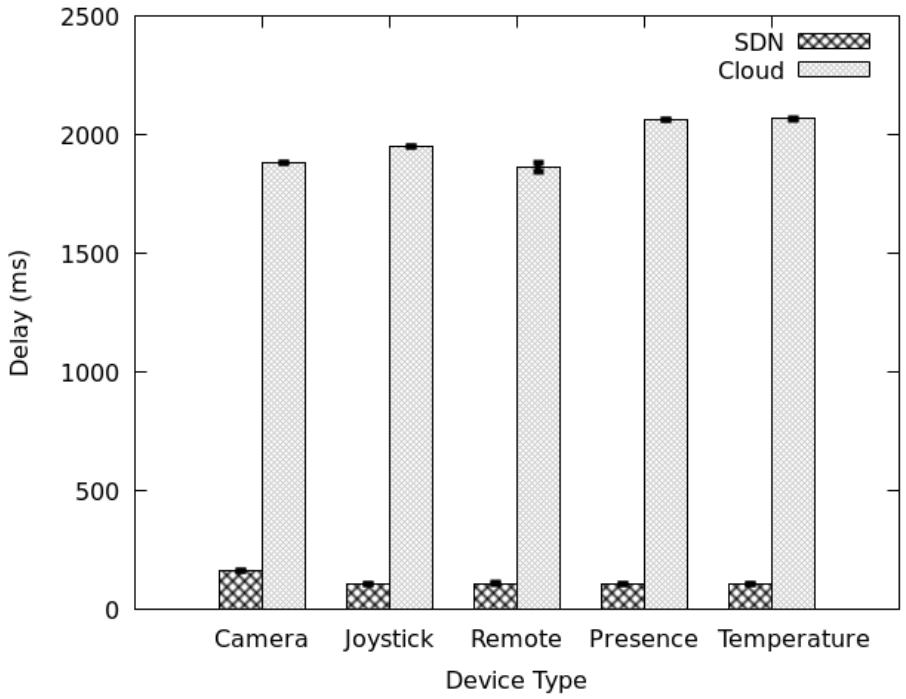


Figure 6.7. Average delay per device model on a SDN-based and a cloud-based environment.

devices.

6.6 Conclusion

In this chapter we presented SMARTEEX, a software tool for smart building network emulation. We also investigated the performance of current smart building network architectures, showing the pros and cons of a cloud based smart building logic control system. While a cloud system allows for powerful improvements and services, it exposes the smart building to risks related to an attacker taking control of the cloud system. The smart building owner, or the inhabitant, as of today has no defenses against this type attack, and must completely trust the cloud system provider. In the next chapter we will present a blockchain based framework which empowers users, defending them against cloud system attacks.

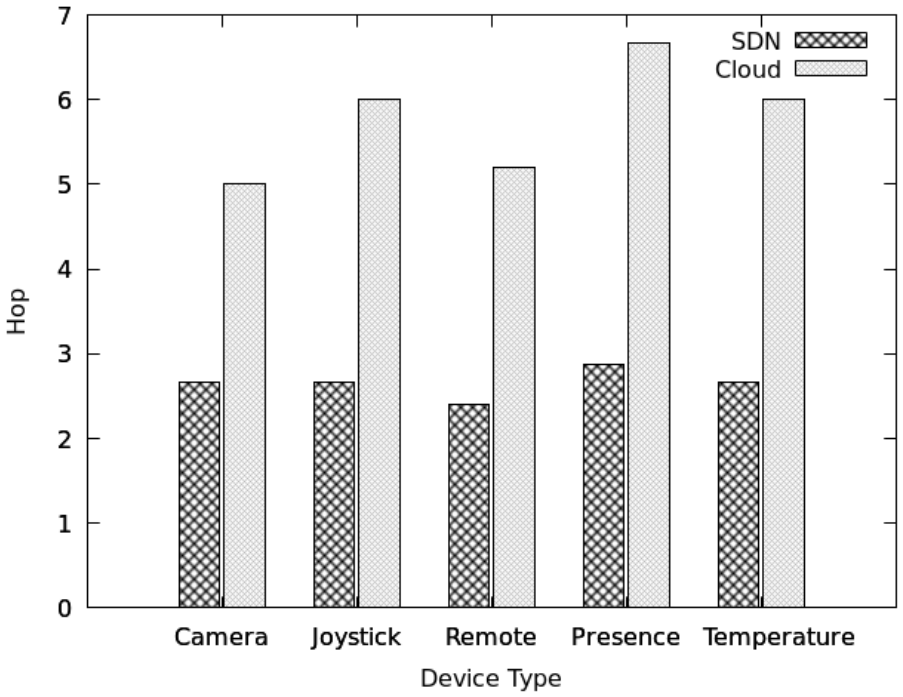


Figure 6.8. Average number of hops per device model inside a SDN-based and a Cloud-based environment.

Chapter 7

HyBloSE: Hybrid Blockchain for Secure-by-Design Smart Environments

7.1 Chapter Introduction

Thanks to the Internet of Things (IoT), a fast-growing number of buildings is becoming *smarter*, connecting to the Internet an expected number of 55+ billion kitchen appliances, televisions, smartphones, utility meters, intra-body sensors, thermostats, etc. by 2025 [131, 137, 127, 152]. As homes and offices become more autonomous, we ask the following question: *can a smart building be dangerous like a smart vehicle?* Unfortunately, the security issues related to the development of a smart building are often undervalued [42]. These issues are further exacerbated by the centralized nature of the IoT. However, delegating the full control of an autonomous place to devices and cloud services leads to security vulnerabilities [35].

Figure 7.1 shows how the location of IoT devices can raise security and privacy concerns. For example, a smart plug may attack other devices or steal passwords [66], a widespread closed-source communication protocol may be exploited to allow attacks [15], or an authority may trespass into smart devices to spy on people [14]. The key issue is that centralized systems controlling smart environments are susceptible to attacks due to incorrect behavior of employees or authorities [61]. Thus, it is essential to limit and control the capabilities of remote centralized control systems, introducing a *decentralized and trust-less system*, only manageable by the smart environment owners.

Existing work has tackled smart environment security through blockchain [104, 62, 7, 31, 6, 29]. However, speed of execution, privacy and cost issues have limited the adoption of existing techniques in widespread commercial devices. The main limitations are (i) the need for additional hardware to be installed inside the smart environment; (ii) the need for IoT devices with high energy demand; (iii) the need to deal with blockchain related issues, like slowness, costs and privacy; (iv) the usage of private blockchain, which may be modified by the owner producing multiple forks

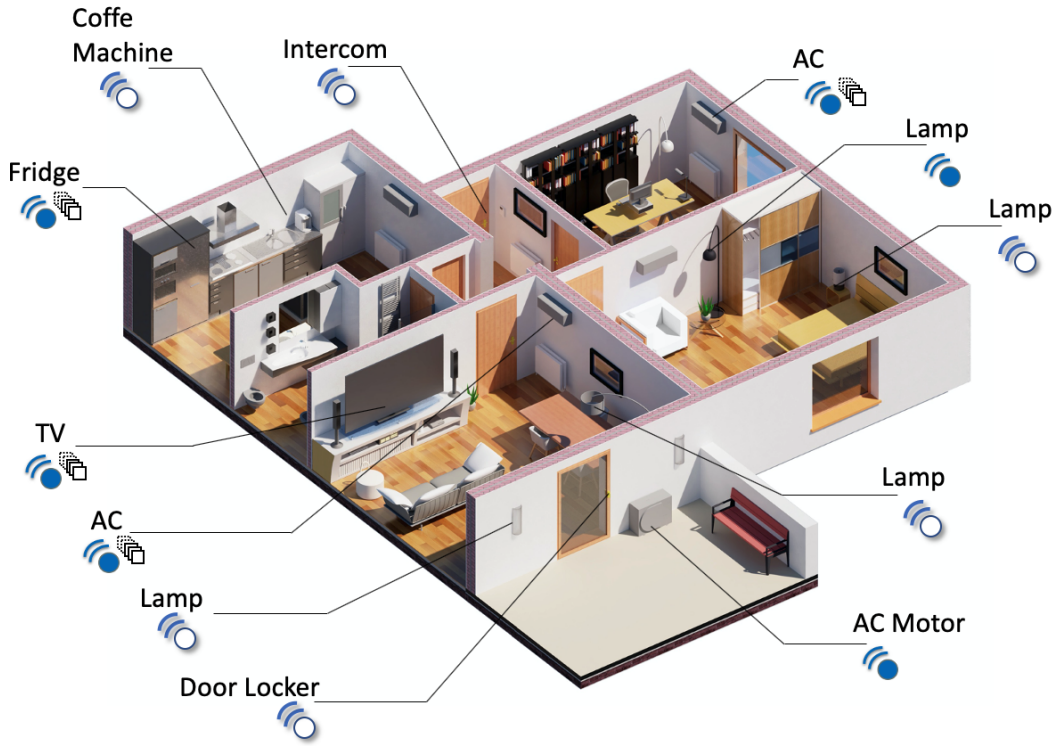


Figure 7.1. IoT Devices inside a Smart Building

[62].

In this chapter we address the current research gap by proposing HyBloSE. Our goal is to make the use of blockchain practical for smart environments, with a lightweight solution able to run on low power devices without requiring additional hardware. HyBloSE is a *secure-by-design* blockchain framework based on a Delegated Proof of Authority and a Moving Window Blockchain.

To tackle these limitations, HyBloSE introduces several novel features for smart buildings:

• HyBloSE makes a smart environment secure-by-design by splitting up security from application logic, and introducing an additional layer of security. This approach maintains the benefits of cloud-based logic systems (*e.g.*, speed) limiting the drawbacks introduced by blockchain;

• HyBloSE introduces a *Public Smart Building Contract*, which clarifies and defines the functionalities of each device or system of a smart environment, restricting its capabilities even in case of exploitation. HyBloSE is based on a Delegated Proof of Authority. In this way HyBloSE does not involve any significant overhead, and differently form other blockchain based proposals, it does not require any additional hardware (e.g., powerful server inside homes);

• Finally, to allow traceability and be compliant with privacy regulations, HyBloSE presents a new approach to hybrid blockchain, introducing a moving window of signed contents, which are periodically deleted.

The remainder of this chapter is structured as follows. Section 7.2 presents a

description of the challenges which are still to be tackled to implement a blockchain system inside a smart building. Section 7.3 presents HyBloSE, and how these challenges are addressed. Section 7.4 assesses the effectiveness and the efficiency of the proposed architecture through emulations and real experiments, Section 7.5 presents the related work and Section 7.6 draws some conclusions on HyBloSe.

7.2 Challenges

The objective of this work is to make the use of blockchain practical for smart buildings. This involves several challenges, as we explain.

Speed: Blockchain is slow. How can we make it fast enough to meet expectations of smart building users? Well-known public blockchains are very slow for smart building applications. Bitcoin accepts up to 10 transaction per second, with a delay between each block of 15 minutes [61]. The faster Ethereum accepts up to 30 transactions per second, with a delay between each block of 20 seconds [138]. This means that when a user issues a new transaction, *e.g.*, a transfer of funds or the execution of a function on the blockchain, she has to wait for at least 20 seconds for the request confirmation. However, users expect fast response, *e.g.*, a light starts emitting light after touching a switch. A way to reduce this delay with the Proof of Work (PoW) consensus is to reduce the puzzle complexity [138]. However, this approach is faulty as the reduction makes the system open to attacks based on controlling the majority of the nodes.

Cost: Writing on a public blockchain has an economic cost. How can we make this cost low and independent of the building size? A public blockchain, like Bitcoin or Ethereum, is a distributed network of systems all running the same protocol and all owning a copy of the contents of the blockchain. The spread of new data through all the members of a blockchain requires time, thus public blockchains are slow in confirming transactions. Furthermore, writing on a public blockchain costs monetary resources, then used to refund the users that participate in the blockchain. On Ethereum, this fee increases depending on how much we want to write, and is around $0.10 per kilobyte [16]. Also, for each operation we want to perform there is a minimum fee of around $0.05. This means that for each proposal based on the Ethereum public blockchain, users are expected to pay at least $0.05 for each operation they perform.

Low power devices: How can we connect low power devices to a blockchain? There are multiple ways for a system to connect to a public blockchain. The first one involves a full node that connects itself to the other nodes running the public blockchain, obtains from them the entire blockchain, verifies it, and continuously receives new blocks from other nodes. A full node is also able to mine. However, this node must have a large amount of space available, *e.g.* more than 140 gigabytes for Ethereum. Also, if the system wants to mine it must be provided with at least a 64 bit processor, and with a powerful GPU. It is not realistic to expect such power on many IoT devices devices. The second way is through a light node: instead of

maintaining the full blockchain, it stores only the block headers, reducing the space required. While this node can run on modest devices, it is still not able to run on low power devices, as it should be always connected to the peer to peer network. Another way to reach a public blockchain is through a gateway, like Infura [32]. These systems are free and publicly available, and offer classical REST API which devices can contact to interact with the public blockchain. These systems have full nodes running, and translate REST requests to blockchain transactions. However, the adoption of such systems creates a new single point of centralization.

Integrity: How can we guarantee that a private blockchain has not been altered? A way to solve the cost and speed issues related to the use of a public blockchain is the adoption of a private blockchain. Instead of having a public peer-to-peer network of miners, we have a close set of nodes connected together which operate on a blockchain disconnected from the public one. The owner of the nodes in this case can modify the underlying protocol, for example changing the way in which the Proof of Work puzzle complexity is defined, reducing or increasing it. However, as the owner can control all the nodes, she may easily change their content. For example, in case of an inappropriate event, the owner may induce some nodes to produce a fork, maintaining two copies of the blockchain, one with the event and the other one without.

How can we combine the blockchain immutability with the privacy right to be forgotten? In the last years the need for users' privacy protection has become evident [30]. Two of the most discussed principles of privacy are the temporality of data and the right to be forgotten. The first principle defines that collected data must be stored for a limited amount of time, depending on its specific utilization. The second defines that each user must be able to ask for the removal of data that regards itself. Both these principles are not obtainable on a public blockchain, and require specific changes in a private one.

7.3 The HyBloSE System

We propose HyBloSE (see Figure 7.2), a practical solution that enhances smart environment security and maintains the advantages of a centralized orchestration, involving a public and a private moving blockchain. HyBloSE addresses the challenges as follows. To address speed HyBloSE introduces a Delegated Proof of Authority (DPoA) and a Private Moving Blockchain. To reduce cost HyBloSE uses a combination of the Ethereum public Blockchain and a private moving Blockchain. To integrate low-power devices HyBloSE exploits powerful nodes as a gateway for low-power ones, in addition to the DPoA. To ensure integrity and to guarantee privacy, HyBloSE exploits the characteristics of the private moving window blockchain.

Delegated Proof of Authority The Proof of Authority is a consensus mechanism where only a set of authorized nodes can mine. In a smart building, the owner can indicate in a Smart Contract the authorized nodes. To limit power consumption,

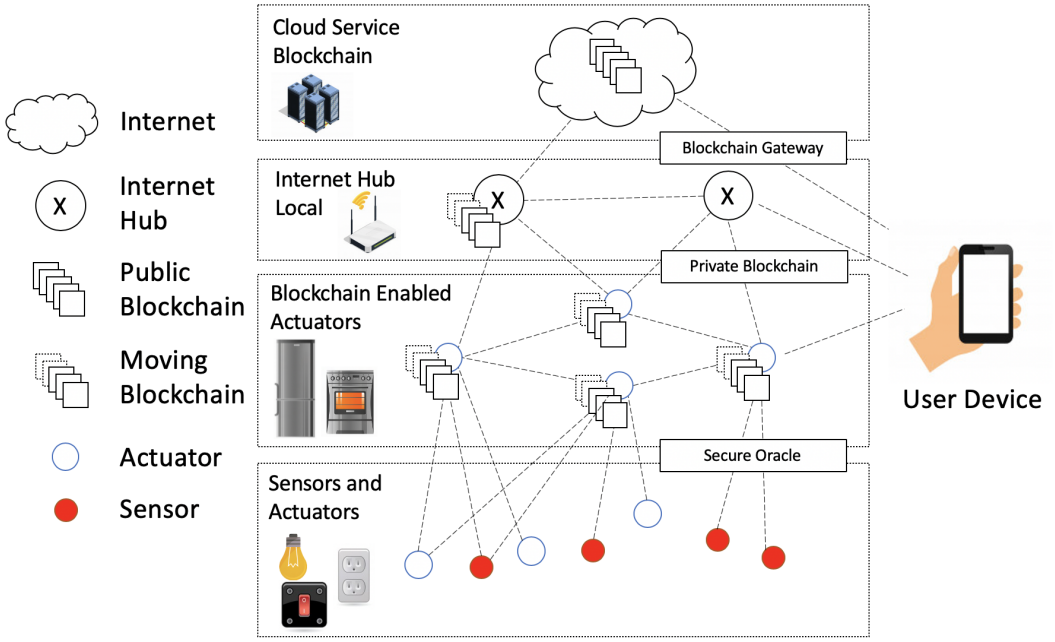


Figure 7.2. HyBloSE System Overview.

and to let only part of the nodes mine, in the private moving blockchain a *Delegated Smart Contract* elects two miners to mine for the next hour.

At the end of each hour elected miners invoke the Delegate Smart Contract, and elects the two miners which will be active for the next hour. This election is performed by calculating the number of operations that happened in the last hour, excluding the ones produced by the active miners, and with a modulo operation on the set of potential authorized nodes. The two miners that are currently mining can not mine in the successive hour. The number of operations is required because current nodes may influence a random number, becoming able to deterministically define the next nodes that will mine.

The number of operations is hard to be forged by the miners, as the only way they have would be to refuse some device requests. However, refusing requests automatically activates the other miners which were not mining, activating a security procedure, in which the other nodes produce a new fork and the exclusion of the misbehaving node. Non elected nodes perform two actions: i) verify the operation of elected nodes, and in case of misbehavior start mining; ii) verify that the elected nodes include in the private blockchain all the requests from other nodes; otherwise, in case elected nodes are excluding some device, not elected nodes start mining. In this way only two nodes per time have high energy consumption, but an attacker still has to exploit the majority of all the nodes with mining capabilities to control the building. Differently from the PoW, the DPoA does not requires the ethash DAG computation. This operation requires a huge amount of memory and a 64bit processor. Removing this operation, the DPoA can be executed even on low power nodes with reduced capabilities.

Private Moving Blockchain Each device with a fixed source of power and with modest computation capabilities, for reference a Raspberry Pi Zero is enough, can run the Private Moving Blockchain. This blockchain follows the Ethereum protocols, while differing for three reasons. First, it implements the Delegated Proof of Authority consensus, meaning that only authorized devices, the ones contained in the set M of the *Public Smart Building Contract*, can mine and create new blocks. This blockchain manages the *Delegated Smart Contract* and a perfect copy of the Public Smart Building Contract, that is continuously updated. Second, periodically the hash of some blocks is written to the public blockchain. Third, periodically the oldest part of the private blockchain is removed. The remaining part always starts from the header of a node whose hash has been recorded on the public blockchain. In this way HyBloSE is able to satisfy the privacy requirements of data temporarily and right to be forgotten. The privacy is also increased as the private moving blockchain is only local to the smart building, and not accessible from outside.

Secure Oracle Low power nodes cannot run a Blockchain node. In order to interact with the moving blockchain Blockchain-Enable actuators act as gateways for low power nodes. However, low power nodes sign the requests they send to the gateways with their ethereum account. In this way the Blockchain-Enable actuators are not able to impersonate other nodes, but can only receive their requests and write them on the moving blockchain. The entire system, as depicted in Fig. 7.2, is based on a mesh network. This avoids disconnections of low power nodes due to a single Blockchain-Enable actuators failure.

Blockchain-Enabled Actuators All smart building actuators connected to a fixed source of power can be miners. The computational capability required to contribute to the Moving Window Blockchain is not high, as it is based on the Proof of Authority consensus. These actuators receive information, i.e., sensed data, from sensors or remote services. For example, when a switch is enabled by an user, the switch sends the information to at least one Blockchain-Enabled Actuator. Active miners will take the information and insert it in the first available block. Depending on the sensed information, an actuator may be required by an external logic system to perform an action. This request is written to the Moving Blockchain. Once that the actuator reads the request, it verifies the request signatures and the request maker's authorizations.

Public Smart Building Contract The Smart Contract is deployed on the public Ethereum blockchain, and contains:

- $U = \{u_1, ..., u_n\}$ a set containing the Ethereum addresses of the users allowed to interact with the contract;
- $M = \{m_1, ..., m_n\}$ a set containing the Ethereum addresses of the authorized miners inside a smart building;
- $D = \{d_1, ..., d_n\}$ a set containing the Ethereum addresses of the entities, devices and remote services, related to the smart building;

- $A = \{a_1, ..., a_n\}$ a set of encoded strings defining all the actions which can be performed by actuators;
- $R(d_s, d_a, a) - > 0, 1$ a function which takes as input the addresses of two devices and an action, and defines if the device d_s can trigger the action a of device d_a.

Each time a user wants to define a new building, he deploys a new Public Smart Building Contract. For each new device, the user updates the smart contract with identities and rules.

Users In order to start HyBloSE a user has to perform the following actions: i) she deploys the Public Smart Building Contract to the Ethereum public blockchain; ii) she sets the Public Smart Building Contract in the private moving window blockchain; iii) the private moving window creates a copy of the Public Smart Building Contract and asks to the user to set the list of authorized miners, the list of smart building devices and related authorizations. With HyBloSE the owner of the smart building is able to define the rules for his autonomous place, and even in case of a cloud service or device attack, actuators would not be able to perform actions not already allowed, making the autonomous place secure by design. Through the use of blockchain, HyBloSE guarantees that a cloud service or an IoT device, even if compromised, cannot perform unforeseen actions. Also, with HyBloSE it is always possible to track who or what triggered an action, both for statistics and accountability.

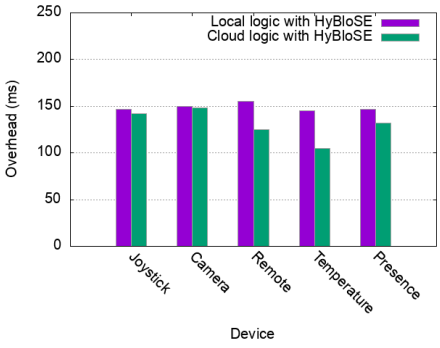


Figure 7.3. Emulation results on HyBloSE's overhead introduced on local and cloud logic architectures (40 devices).

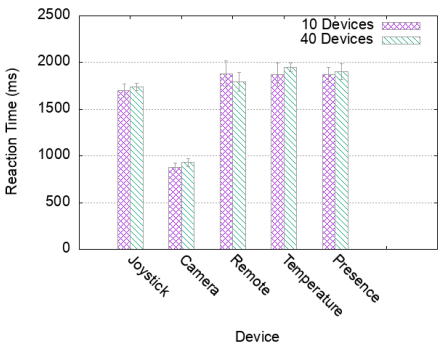


Figure 7.4. Emulation results on HyBloSE's reaction time for different network sizes (10 vs 40 devices) by varying the types of devices.

7.4 Performance Evaluation

To study the effectiveness of HyBloSE we perform a qualitative analysis, while to assess efficiency we carry out a quantitative analysis that includes both emulation and real experiments. We consider two system architectures. The first one, named cloud logic, considers a cloud-based management logic in which all sensor packets are routed to actuators through the cloud service. The second architecture, named local

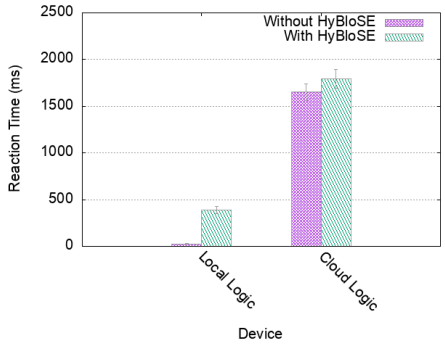


Figure 7.5. Experimental results on the impact of HyBloSE on reaction time (10 devices).

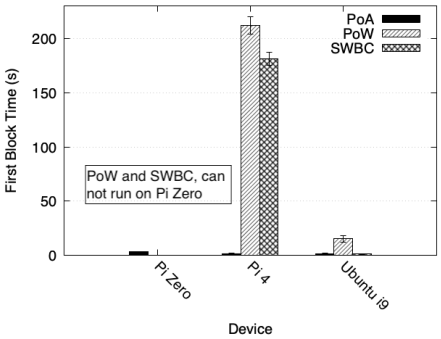


Figure 7.6. First block generation time of DPoA (HyBloSE), PoW and SWBC (based on PoW).

logic, keeps all decisions local and sensor packets are routed directly to actuators, according to a local management system. These architectures are considered in a standalone fashion as well as integrated with the HyBloSE system, so that we can evaluate the impact of introducing the cryptographic operations.

7.4.1 Effectiveness

We now look at the effectiveness of HyBloSE through a qualitative security analysis, which follows the approach described in [61, 133]. Table 7.1 shows how HyBloSE is resilient to different types of threats in each layer of the blockchain. The application layer is mainly focused on issues related to the centralization points of a system. HyBloSE tackles most of them with DPoA, as only authorized nodes can take part in the system, and through its mesh network. The mesh structure makes also HyBloSE robust to network layer attacks. The Smart Contract layer examines the application source code. As HyBloSE Smart Contracts have the same source for every building, the surface of potentially vulnerable code is reduced. The third layer refers to the consensus mechanism, and the Delegated Proof of Authority is able to contrast all the reported security risks. The fourth investigates the network, and the last investigate the Data layer, where the private moving blockchain is fundamental, as able to guarantee confidentiality, as the data are local to the smart building, and privacy, as data are periodically removed.

7.4.2 Efficiency: Emulation

We evaluate HyBloSE by using SMARTEEX [78], a software tool to emulate complex network scenarios and collecting statistics. We consider a smart home scenario with up to 40 devices. We measure the *overhead* — the time incurred by HyBloSE operations— and the *reaction time* — the time from when a sensor detects a new event to when the actuator takes the corresponding action.

Figure 7.3 shows the results on overhead by varying the type of devices. The additional time incurred by applying HyBloSE respectively to local and cloud logic

architectures, is very short (between 100ms and 150 ms), and is negligible from a user perspective. Furthermore, the overhead is quite constant independently of the type of device. We then evaluate the impact of diminishing the number of devices, as a measure of scalability. We consider two different settings: (i) a simple case including 10 devices (5 sensors and 5 actuators), and (ii) a more complex case, with 40 devices (20 sensors and 20 actuators). In both settings, the number of devices does not have an impact on reaction time. Results with 10 and 40 devices are comparable, independently of the type of devices. Thus, our first set of results clearly show that HyBloSE is lightweight (i.e., contained overhead) and scalable.

7.4.3 Efficiency: Experiments

We now present the results of our experimental evaluation with 10 devices: 3 Raspberry Pi Zero W, 3 Raspberry Pi 4, 1 Raspberry Pi 3b+, 1 Raspberry Pi 2 and 2 computers, with i7 and i9 processor family. Experimental results (see Fig. 7.5) show that in both architectures the overhead introduced by HyBloSE is limited, i.e., $370ms$ for the local logic architecture and $140ms$ for the cloud logic counterpart. These values confirm the outcome of the emulation analysis. We also investigate how much time the private blockchain uses to create the first block, i.e., the time required between the system startup and the first action in the smart building. We compare it with SWBC [61], the most recent state of the art blockchain framework for smart building (based on a modification of PoW), and a classical private blockchain based on PoW. We found that the production of the first block in PoW systems has a high computational cost due to the generation of the Ethash DAG, requisite for mining [138], that requires a long time, $220seconds$ like shown in Figure 7.6, even on a Pi 4. This operation is impossible on 32 bit processors, due to RAM address space limits. This makes PoW based systems not work 32 bit and low power devices, like the Pi Zero. The Prof of Authority instead does not involve a DAG, making it suitable even for less powerful devices. Thus, HyBloSE can run on low power and 32 bit devices, requiring always less than 2 seconds to start, being more than $100times$ faster then solutions based on PoW, as shown in Figure 7.6.

Economic feasibility. We report an effective cost analysis of the operations related to the public blockchain, presented in Table 7.2. This test has been performed with the real contract on the Ethereum main net, performing operations. We defined different gas costs depending on each operation, e.g. we want to be as fast as possible while deactivating rules but we accept a delay while removing a device whose rules have already been deactivated. A lower amount of gas paid produces higher delay in operation execution. The most *expensive* operation is the deployment of the Smart Building Contract, which requires 4.60$, but which is executed only once per building. Other operations have a really low cost, always less then 0.20$. These operations are executed rarely, only when a user introduces or removes devices from the smart building. The snapshot is instead performed once per day, with a monthly cost of about 3$. We recall that all the other operations are performed on the private moving blockchain, and do not involve any cost.

7.5 Related work

The dilemma of how to make IoT secure has raised significant interest from the research community over the last years [102, 121, 142, 39]. Several survey has promoted the adoption of blockchain for IoT security [104, 62, 7, 31, 6, 29]. However, current approaches are still far from being adopted because of several practical issues.

Dorri et al. [37] propose the use of local private blockchain for smart-home, but they require an additional high power system inside the smart-home, which creates a central private blockchain. Approaches with only a private blockchain fail in the traceability feature, as losing the distributed ledger allows modification of the blockchain contents. HyBloSE instead relies on both public and private blockchain, guaranteeing security and transparency. The need for additional hardware is also present in [90], where authors propose to use the blockchain for an access control system based on IoT devices. In this case an Ethereum Smart Contract contains the list of managers, i.e., people able to define specific access control permissions for devices. Adler et al. tackled the problem of low power IoT devices proposing an oracle management hub [1], an hub that translates messages from IoT devices, encoded in CoAP, into JSON-RPC messages understandable by blockchain nodes. As recognized by the authors, this proposal has low applicability in IoT scenarios due to public blockchain transaction costs and delays. HyBloSE instead exploits a private blockchain for permissions management, reducing blockchain related costs and decreasing delays.

The access management problem has also been tackled by Zhang et al. [151], which extended the problem to generic rule-based resources management, introducing one Ethereum Smart Contract for each rule. Interactions between entities, i.e., access to resources, are checked by another smart contract called judge that controls and penalizes devices having unfair behaviour. Even in this work the authors acknowledge that the proposal is hard to be applied: the average time for each operation is around 30 seconds (unacceptable for the user), while costs related to blockchain are not analyzed. Also Pinno et al. [97] worked on permission management through blockchain, presenting ControlChain, a framework composed of four blockchains which supports access management models like RBAC, OrBAC, ABAC, UCON, CapBAC. However, ControlChain has not been publicly deployed, and the proposal does not evaluate costs and delays of the new blockchain. HyBloSE instead is deployed on a public, trustworthy and active blockchain. This allowed us to evaluate both the HyBloSE costs and induced delays, demonstrating the negligible impact of our proposal.

7.6 Conclusion

In this chapter, we have presented HyBloSE, a system able to secure smart buildings even in case of attack to the cloud management system. Our approach is based on a Delegated Proof of Authority consensus and a Moving Window Blockchain. HyBloSE decouples the security aspects from the application logic of IoT, saving the centralized logic and distributing the security management. HyBloSE overcomes

related work limitations, especially in terms of practicality: HyBloSE is able to run on low power devices, and without requiring any additional hardware. Results demonstrate that HyBloSE induces negligible overhead (below $0.4ms$), being fully compatible with already deployed hardware.

Table 7.1. Qualitative Analysis of HyBloSE

Layer	Threat	Analysis
Application	Exchange Unauthorized Access	Only nodes authorized by the smart building owner can access the private moving blockchain. If the intruder controls less than half of the nodes running the private blockchain it cannot perform attacks.
	Exchange DDoS	The private network is based on a mesh, and has no single point of failure. In case of misbehaving nodes, these nodes are excluded from mining or from the entire system.
	Employees Host Security	No personal data is collected in the public or the private blockchain. The actions which happen inside the smart building are only temporarily recorded on the private blockchain.
	Malicious Program Infection	A malicious agent must compromise at least the majority of the mining nodes, otherwise the other nodes can exclude compromised ones automatically.
	Mining Pool Attack	The PoA is robust to the mining pool attack, as only authorized nodes can mine.
Smart Contract	Reentrancy Attack	Each device to device authorization is verified as the first step of each action execution, making the reentrancy attack not feasible.
	Unauthorized Access Attack	Since each authorized miner is directly recorded in the private blockchain and managed by voting, an attacker in order to become dangerous must first control more than half of the mining nodes.
	Solidity Development Security	The entire system is based on a set of standard smart contracts.
Consensus	Bribe attack	Validators do not follow an economic logic. This makes the bribe attack not applicable.
	Long range attack	The private key of the authorized miners is known only to the miners. For this reason an attacker should first exploit the nodes before making the attack.
	Precomputing Attack	If a node performs a Precomputing Attack, other sleeping powerful nodes would reveal it. These nodes would then start mining and mitigate the attack.
	Sybil Attack	As the network of nodes is a mesh, it is hard to disconnect it or to hide parts of it. However, all systems which provide a private blockchain are vulnerable in case of network disjunction.
Network	Eclipse Attack	The mesh network limits this attack feasibility. Furthermore, a node performing dos or behaving inconsistently may be excluded by the others with a vote.
Data	Block Data	The private moving blockchain may only collect structured data coming from authorized nodes.
	Signature and Encryption Method	Now, the SHA256 algorithm is used, but it can be updated with a new building smart contract.

Table 7.2. Cost analysis of HyBloSE

Operation	GAS	GWei	Time(s)	$
Smart Building Contract Deployment	760022	25	226	4.60$
Add a new Device	21800	29	64	0.15$
Remove a Device	21000	18	6598	0.09$
Create a new Rule	21204	29	64	0.14$
Deactivate a Rule	21000	32	27	0.16$
Private Blockchain Snapshot	21800	18	6598	0.10$

Chapter 8

Conclusions

It is possible to develop smart homes and offices which have a sustainable impact on the environment?
In this thesis we proposed a set of technological solutions to tackle the sustainability problem born from the massive development of IoT devices. One of the key issues of this army is the disposal of the batteries they need to operate. While it is possible to develop low power devices that need battery changes rarely, we tried to radically solve the issue, by removing batteries entirely.
In the first chapter we showed how with backscattering it is possible to achieve it, guaranteeing both wireless communication and power to devices from a fixed source. We collected a set of already realised devices, starting from a temperature sensor, up to a phone and a camera.
While these devices are characterized by an impressive lifetime, they are limited by the small amount of energy they can collect from radio waves. This limit translates to computation capabilities, and the impossibility to use strong cryptography over RFID devices. We tried to tackle this limit in chapter 3, where we proposed a radio fingerprint framework for RFID devices based on convolutional neural networks. We collected raw signals from 200+ RFID, even simulating implantable devices, and trained a neural network able to detect non-original devices.
After developing and protecting RFID devices, we faced the problem of how to make these devices communicate. While there exist a pletora of MAC protocols, to the best of our knowledge none has been developed to manage the communication of a set of battery-free devices. For this reason, in chapter 4, we introduce APT-MAC, an adaptive communication MAC protocol for battery free devices, which exploits reinforcement learning to maximize the performance of the device's network. Leveraging on the reinforcement learning algorithm, the protocol is able to predict which devices will have to communicate more or less, respectively assigning more or less bandwidth. We developed a set of device analytical models based on Markov Chains, and we evaluated the performance of APT-MAC in a simulated scenario.While APT-MAC is a good protocol for stable scenarios, it presented some limitations in continuously changing ones. Based on a state-less reinforcement learning algorithm, the protocol is not able to remember patterns once they changed.
To learn these changes we introduce the concept of a *mode* in chapter 5, where we present another protocol for battery free devices called ReLEDF, which is based

on a combination of Q-Learning, a reinforcement learning algorithm, and Earliest Deadline First, an algorithm for task scheduling. Analisyng the network pattern of a set of IoT devices inside a smart building we noticed that they are strongly influenced by periods of time. For example, a presence sensor inside an office will be triggered hundreds of times during the day, and rarely during the night. This led to the introduction of variable scenarios, and leveraging Q-Learning we are able to immediately recognize a scenario and to optimize the network to what it is happening. However, as the number of scenarios may explode, we introduced the concept of *sub-agents*, meaning that in a single Q-Learning problem we have a set of agents instead of a single one. The coordination of these agents has been assigned to the EDF algorithm, which translates network requirements into slot assignments. We showed how ReLEDF can satisfy the requirements of a smart building, detecting in few seconds changes in the device utilization scenarios.
After realizing battery free devices, making them secure and communicating, we investigated the coexistence of battery-free and classical devices. For this reason we developed SMARTEEX, a network simulator based on Mininet specific for buildings, which is presented in chapter 6. This tool, with a short configuration, allows for the emulation of complex scenarios with multiple rooms and devices per room. It also allows for mixed scenarios, with real and simulated devices. With SMARTEEX we have been able to evaluate the network performance of a smart building, showing that with an SDN based approach it would be possible to improve the overall performances, both in terms of speed and network resilience. Classical IoT devices are usually connected through remote servers, even if inside the same local network. This remote connection, other then being the cause of service interruption in case of network disconnection, can lead to risks to a smart building. As of today, in case of an attack to the remote server which manages IoT devices, the owner of the smart building has no defences. This means that a smart home may be controlled by an attacker which can control the cloud service connected to the IoT devices inside, or by an unfaithful employee, or even worse by governments. To tackle this issue in chapter 7 we propose HyBloSe, an hybrid blockchain able to act as security controller for smart buildings. While the logic of the building is left to cloud services, HyBloSe can locally check that the operations required by the cloud service are authentic and that have really been authorized by the smart building owner.

In these thesis we presented an approach to the development of smart building, performing for each proposal numerical and qualitative analysis. While some of these approaches are still not ready for a widespread use, we tried to show that step by step we can design a new smart building network infrastructure based on a reduced environmental footprint and security.
Each step closer to the sustainability of our environment, for the health of our life.

Bibliography

[1] Adler, J., Berryhill, R., Veneris, A., Poulos, Z., Veira, N., and Kastania, A. Astraea: A decentralized blockchain oracle. In *2018 IEEE iThings and IEEE GreenCom and IEEE CPSCom and IEEE SmartData*. IEEE (2018).

[2] Ahson, S. A. and Ilyas, M. *RFID handbook: applications, technology, security, and privacy*. CRC press (2017).

[3] Ai, X., Chen, H., Lin, K., Wang, Z., and Yu, J. Nowhere to hide: Efficiently identifying probabilistic cloning attacks in large-scale rfid systems. *IEEE Trans. on Information Forensics and Security*, (2020).

[4] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, **17** (2015), 2347.

[5] Al-Shawabka, A., et al. Exposing the Fingerprint: Dissecting the Impact of the Wireless Channel on Radio Fingerprinting. *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, (2020).

[6] Ali, M. S., Vecchio, M., Pincheira, M., Dolui, K., Antonelli, F., and Rehmani, M. H. Applications of blockchains in the internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, **21** (2018), 1676.

[7] Alotaibi, B. Utilizing blockchain to overcome cyber security concerns in the internet of things: A review. *IEEE Sensors Journal*, **19** (2019).

[8] Amazon. Amazon Echo. `http://www.amazon.com/echo` (2020).

[9] Andersson, B. and Ekelin, C. Exact admission-control for integrated aperiodic and periodic tasks. *Journal of Computer and System Sciences*, **73** (2007), 225 . Special Issue: Real-time and Embedded Systems.

[10] Angell, I. and Kietzmann, J. RFID and the End of Cash? *Communications of the ACM*, **49** (2006), 90–96. Available from: `https://doi.org/10.1145/1183236.1183237`, `doi:10.1145/1183236.1183237`.

[11] Annalee Newitz (Wired). The RFID Hacking Underground. `https://www.wired.com/2006/05/rfid-2/` (2006).

[12] Anonymous. Epc uhf gen2 air interface protocol - epc/rfid (2015). Available from: `https://www.gs1.org/standards/epc-rfid/uhf-air-interface-protocol`.

[13] Arumugam, D. and Engels, D. Specific absorption rates in the human head and shoulder for passive uhf rfid systems. *International Journal of Radio Frequency Identification Technology and Applications*, **2** (2009), 1.

[14] Assange, J. Vault 7: Cia hacking tools revealed. *WikiLeaks.(Mar. 2017). Retrieved Mar*, **7** (2017), 2017.

[15] Badenhop, C. W., Graham, S. R., Ramsey, B. W., Mullins, B. E., and Mailloux, L. O. The z-wave routing protocol and its security implications. *Computers & Security*, **68** (2017), 112.

[16] Bartolini, N., Coletta, A., Maselli, G., and Piva, M. Druber: A trustable decentralized drone-based delivery system. In *ACM DroNet 2020* (2020).

[17] Baruah, S., Bonifaci, V., D'angelo, G., Li, H., Marchetti-Spaccamela, A., Van Der Ster, S., and Stougie, L. Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems. *Journal of the ACM (JACM)*, **62** (2015), 14.

[18] Benedetti, D., Maselli, G., Petrioli, C., and Piva, M. The impact of external interference on rfid anti-collision protocols. *IEEE Networking Letters*, **1** (2019), 76.

[19] Bertoncini, C., Rudd, K., Nousain, B., and Hinders, M. Wavelet Fingerprinting of Radio-frequency Identification (RFID) Tags. *IEEE Trans. on Industrial Electronics*, **59** (2011), 4843.

[20] Block, A. and Anderson, J. Accuracy versus migration overhead in real-time multiprocessor reweighting algorithms. In *12th International Conference on Parallel and Distributed Systems - (ICPADS'06)*, vol. 1 (2006).

[21] Brik, V., Banerjee, S., Gruteser, M., and Oh, S. Wireless Device Identification with Radiometric Signatures. In *Proc. of the 14th ACM international conference on Mobile Computing and Networking (MobiCom)*, pp. 116–127. ACM (2008).

[22] Bu, K., Weng, M., Zheng, Y., Xiao, B., and Liu, X. You Can Clone But You Cannot Hide: A Survey of Clone Prevention and Detection for RFID. *IEEE Communications Surveys & Tutorials*, **19** (2017), 1682.

[23] Buettner, M. and Wetherall, D. A gen 2 rfid monitor based on the usrp. *ACM SIGCOMM Computer Communication Review 2010*, **40** (2010).

[24] Buttazzo, G. C., Lipari, G., Caccamo, M., and Abeni, L. Elastic scheduling for flexible workload management. *IEEE Transactions on Computers*, (2002), 289.

[25] Carneiro, G. Ns-3: Network simulator 3. In *UTM Lab Meeting April*, vol. 20, pp. 4–5 (2010).

[26] Casini, D., Biondi, A., and Buttazzo, G. C. Handling transients of dynamic real-time workload under edf scheduling. *IEEE Transactions on Computers*, (2018).

[27] Chen, S., Zhang, M., and Xiao, B. Efficient information collection protocols for sensor-augmented rfid networks. In *2011 Proceedings IEEE INFOCOM*, pp. 3101–3109. IEEE (2011).

[28] Chen, X., Liu, J., Wang, X., Zhang, X., Wang, Y., and Chen, L. Combating tag cloning with cots rfid devices. In *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9. IEEE (2018).

[29] Christidis, K. and Devetsikiotis, M. Blockchains and smart contracts for the internet of things. *Ieee Access*, **4** (2016), 2292.

[30] Coletta, A., Maselli, G., Piva, M., Silvestri, D., and Restuccia, F. My sim is leaking my data: Exposing self-login privacy breaches in smartphones. *arXiv preprint arXiv:2003.08458*, (2020).

[31] Conoscenti, M., Vetro, A., and De Martin, J. C. Blockchain for the internet of things: A systematic literature review. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1–6 (2016).

[32] Consensys. Scalable blockchain infrastructure (2020). Available from: `https://infura.io/`.

[33] Danev, B., Capkun, S., Jayaram Masti, R., and Benjamin, T. S. Towards Practical Identification of HF RFID Devices. *ACM Trans. on Information and System Security (TISSEC)*, **15** (2012), 1.

[34] Danev, B., Heydt-Benjamin, T. S., and Capkun, S. Physical-layer Identification of RFID Devices. In *Proc. of the USENIX Security Symposium*, pp. 199–214 (2009).

[35] Deogirikar, J. and Vidhate, A. Security attacks in iot: A survey. In *2017 International Conference on I-SMAC*, pp. 32–37. IEEE (2017).

[36] Dierk, C., Nicholas, M. J. P., and Paulos, E. Alterwear: battery-free wearable displays for opportunistic interactions. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–11 (2018).

[37] Dorri, A., Kanhere, S. S., Jurdak, R., and Gauravaram, P. Blockchain for iot security and privacy: The case study of a smart home. In *2017 IEEE PerCom workshops*, pp. 618–623. IEEE (2017).

[38] Farsens. Battery free rfid sensors (2020). Available from: `http://www.farsens.com/en/products/battery-free-rfid-sensors/`.

[39] Fernandes, E., Jung, J., and Prakash, A. Security analysis of emerging smart home applications. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE (2016).

[40] Fisher, N. and Ahmed, M. Tractable real-time schedulability analysis for mode changes under temporal isolation. In *Embedded Systems for Real-Time Multimedia (ESTIMedia), 2011 9th IEEE Symposium on*, pp. 130–139. IEEE (2011).

[41] Fohler, G. Realizing changes of operational modes with a pre run-time scheduled hard real-time system. In *Responsive Computer Systems*, pp. 287–300. Springer (1993).

[42] Frustaci, M., Pace, P., Aloi, G., and Fortino, G. Evaluating critical security issues of the iot world: Present and future challenges. *IEEE Internet of Things Journal*, **5** (2018), 2483.

[43] Galluccio, L., Milardo, S., Morabito, G., and Palazzo, S. Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 513–521. IEEE (2015).

[44] Global, E. EPC Radio-frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz–960 MHz. *Version*, **1** (2008), 23.

[45] Gollakota, S., Reynolds, M. S., Smith, J. R., and Wetherall, D. J. The emergence of rf-powered computing. *Computer*, **47** (2013), 32.

[46] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, vol. 1. MIT press Cambridge (2016).

[47] Google. Google Home. `https://store.google.com/product/google_home` (2020).

[48] Gummeson, J., Mccann, J., Yang, C., Ranasinghe, D., Hudson, S., and Sample, A. Rfid light bulb: Enabling ubiquitous deployment of interactive rfid systems. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **1** (2017), 1.

[49] Han, J., Qian, C., Yang, P., Ma, D., Jiang, Z., Xi, W., and Zhao, J. GenePrint: Generic and Accurate Physical-layer Identification for UHF RFID tags. *IEEE/ACM Trans. on Networking*, **24** (2015), 846.

[50] Han, K. and Huang, K. Wirelessly powered backscatter communication networks: Modeling, coverage, and capacity. *IEEE Transactions on Wireless Communications*, **16** (2017), 2548.

[51] Jagannath, J., Polosky, N., Jagannath, A., Restuccia, F., and Melodia, T. Machine Learning for Wireless Communications in the Internet of Things: A Comprehensive Survey. *Ad Hoc Networks*, **93** (2019). `doi:https://doi.org/10.1016/j.adhoc.2019.101913`.

[52] Jain, R. *The art of computer systems performance analysis*. john wiley & sons (2008).

[53] Jain, R., Durresi, A., and Babic, G. Throughput fairness index: An explanation. (1999).

[54] Johnson, E. Physical Limitations on Frequency and Power Parameters of Transistors. In *1958 IRE International Convention Record*, vol. 13, pp. 27–34. IEEE (1966).

[55] Juels, A. RFID Security and Privacy: A Research Survey. *IEEE Journal on Selected Areas in Communications*, **24** (2006), 381.

[56] K-blue. K-blue. `https://www.kblue.eu/?lang=en` (2020).

[57] Kantareddy, S. N. R., Mathews, I., Bhattacharyya, R., Peters, I. M., Buonassisi, T., and Sarma, S. E. Long range battery-less pv-powered rfid tag sensors. *IEEE Internet of Things Journal*, **6** (2019), 6989.

[58] Kaur, K., Singh, J., and Ghumman, N. S. Mininet as software defined networking testing platform. In *International Conference on Communication, Computing & Systems (ICCCS)*, pp. 139–42 (2014).

[59] Kellogg, B., Parks, A., Gollakota, S., Smith, J. R., and Wetherall, D. Wi-fi backscatter: Internet connectivity for rf-powered devices. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 607–618 (2014).

[60] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, (2014).

[61] Koshy, P., Babu, S., and Manoj, B. Sliding window blockchain architecture for internet of things. *IEEE Internet of Things Journal*, **7** (2020).

[62] Kshetri, N. Can blockchain strengthen the internet of things? *IT professional*, **19** (2017), 68.

[63] La Porta, T. F., Maselli, G., and Petrioli, C. Anticollision protocols for single-reader rfid systems: Temporal analysis and optimization. *IEEE Transactions on Mobile Computing*, **10** (2010), 267.

[64] Laird Connectivity. S902 Series RFID Antenna. `https://www.lairdconnect.com/rf-antennas/rfid-antennas/s902-series-rfid-antenna` (2020).

[65] LeCun, Y. et al. Generalization and network design strategies. *Connectionism in perspective*, (1989), 143.

[66] Ling, Z., Luo, J., Xu, Y., Gao, C., Wu, K., and Fu, X. Security vulnerabilities of internet of things: A case study of the smart plug system. *IEEE Internet of Things Journal*, **4** (2017), 1899.

[67] Liu, C. L. and Layland, J. W. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, **20** (1973), 46.

[68] Liu, V., Parks, A., Talla, V., Gollakota, S., Wetherall, D., and Smith, J. R. Ambient backscatter: Wireless communication out of thin air. *ACM SIGCOMM Computer Communication Review*, **43** (2013), 39.

[69] Loy, D., Dietrich, D., and Schweinzer, H.-J. *Open control networks: LonWorks/EIA 709 technology*. Springer Science & Business Media (2012).

[70] Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y., and Kim, D. I. Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. *IEEE Communications Surveys Tutorials*, **21** (2019), 3133. doi:10.1109/COMST.2019.2916583.

[71] M. Piva, G. Maselli, and F. Restuccia. RFID2020Dataset. mauropv.github.io (2020).

[72] Maleki, H., Rahaeimehr, R., Jin, C., and Van Dijk, M. New clone-detection approach for rfid-based supply chains. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 122–127. IEEE (2017).

[73] Mansoor, K., Ghani, A., Chaudhry, S. A., Shamshirband, S., Ghayyur, S. A. K., and Mosavi, A. Securing iot-based rfid systems: a robust authentication protocol using symmetric cryptography. *Sensors*, **19** (2019), 4752.

[74] Maselli, G., Pietrogiacomi, M., Piva, M., and Stankovic, J. A. Battery-free smart objects based on rfid backscattering. *IEEE Internet of Things Magazine*, **2** (2019), 32.

[75] Maselli, G. and Piva, M. Secy app: Self configuration and easy management for software defined smart homes. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pp. 398–399 (2018).

[76] Maselli, G., Piva, M., Ramponi, G., and Ganesan, D. Joytag: a battery-less videogame controller exploiting rfid backscattering. In *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking*, pp. 515–516 (2016).

[77] Maselli, G., Piva, M., and Restuccia, F. Hyblose: hybrid blockchain for secure-by-design smart environments. In *Proceedings of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pp. 23–28 (2020).

[78] Maselli, G., Piva, M., and Silvestri, D. SMARTEEX: a software tool for SMART environment EXperiments. In *2019 IEEE INFOCOM WKSHPS: CNERT 2019*. Paris, France (2019).

[79] Maselli, G., Piva, M., and Stankovic, J. A. Adaptive communication for battery-free devices in smart homes. *IEEE Internet of Things Journal*, **6** (2019), 6977.

[80] Maselli, G. and Salierno, G. Performance evaluation of a battery-free videogame controller. In *Proceedings of the 14th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, pp. 71–75 (2017).

[81] Maselli, Gaia and Piva, Mauro and Silvestri, Domenicomichele . Smarteex repository. `http://reti.dsi.uniroma1.it/eng/maselli/code/smarteex` (2020).

[82] Masrur, A., Muller, D., and Werner, M. Bi-level deadline scaling for admission control in mixed-criticality systems. In *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 100–109. IEEE (2015).

[83] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics*, pp. 1273–1282 (2017).

[84] Mikołajczyk, A. and Grochowski, M. Data Augmentation for Improving Deep Learning in Image Classification Problem. In *2018 international interdisciplinary PhD workshop (IIPhDW)*, pp. 117–122. IEEE (2018).

[85] Miorandi, D., Sicari, S., De Pellegrini, F., and Chlamtac, I. Internet of things: Vision, applications and research challenges. *Ad hoc networks*, **10** (2012), 1497.

[86] Naderiparizi, S., Parks, A. N., Kapetanovic, Z., Ransford, B., and Smith, J. R. Wispcam: A battery-free rfid camera. In *2015 IEEE International Conference on RFID (RFID)*, pp. 166–173. IEEE (2015).

[87] Nelis, V., Andersson, B., Marinho, J., and Petters, S. M. Global-edf scheduling of multimode real-time systems considering mode independent tasks. In *2011 23rd Euromicro Conference on Real-Time Systems*, pp. 205–214. IEEE (2011).

[88] NEST. NEST. `https://nest.com` (2020).

[89] NEST. NEST Camera. `https://nest.com/cameras/nest-cam-indoor/overview` (2020).

[90] Novo, O. Blockchain meets iot: An architecture for scalable access management in iot. *IEEE Internet of Things Journal*, **5** (2018), 1184.

[91] O'Shea, T. J., Roy, T., and Clancy, T. C. Over-the-air deep learning based radio signal classification. *IEEE Journal of Selected Topics in Signal Processing*, **12** (2018), 168. `doi:10.1109/JSTSP.2018.2797022`.

[92] Parks, A. N., Sample, A. P., Zhao, Y., and Smith, J. R. A wireless sensing platform utilizing ambient rf energy. In *2013 IEEE Topical Conference on Biomedical Wireless Technologies, Networks, and Sensing Systems*, pp. 154–156. IEEE (2013).

[93] Peng, L., Hu, A., Zhang, J., Jiang, Y., Yu, J., and Yan, Y. Design of a Hybrid RF Fingerprint Extraction and Device Classification Scheme. *IEEE Internet of Things Journal*, **6** (2019), 349. doi:10.1109/JIOT.2018.2838071.

[94] Perez, L. and Wang, J. The Effectiveness of Data Augmentation in Image Classification Using Deep Learning. *arXiv preprint arXiv:1712.04621*, (2017).

[95] Periaswamy, S. C. G., Thompson, D. R., and Di, J. Fingerprinting RFID Tags. *IEEE Trans. on Dependable and Secure Computing*, **8** (2010), 938.

[96] Philips. HUE. https://www.meethue.com (2020).

[97] Pinno, O. A., Gregio, A. A., and De Bona, L. C. Controlchain: Blockchain as a central enabler for access control authorizations in the iot. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6. IEEE (2017).

[98] Piva, M., Maselli, G., Coletta, A., and D., S. Analysis of new and existing methods of reducing intercarrier interference due to carrier frequency offset in OFDM. *ACM Transaction on Internet of Things*.

[99] Qiao, Y., Chen, S., Li, T., and Chen, S. Tag-ordering polling protocols in rfid systems. *IEEE/ACM Transactions on Networking*, **24** (2015), 1548.

[100] Real, J. and Crespo, A. Mode change protocols for real-time systems: A survey and a new proposal. *Real-time systems*, **26** (2004).

[101] Restuccia, F., D'Oro, S., Al-Shawabka, A., Belgiovine, M., Angioloni, L., Ioannidis, S., Chowdhury, K., and Melodia, T. DeepRadioID: Real-Time Channel-Resilient Optimization of Deep Learning-based Radio Fingerprinting Algorithms. In *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM MobiHoc)* (2019).

[102] Restuccia, F., DÓro, S., and Melodia, T. Securing the internet of things in the age of machine learning and software-defined networking. *IEEE Internet of Things Journal*, **5** (2018), 4829.

[103] Restuccia, F. and Melodia, T. Deep learning at the physical layer: System challenges and applications to 5g and beyond. *IEEE Communications Magazine*, **58** (2020), 58. doi:10.1109/MCOM.001.2000243.

[104] Reyna, A., Martín, C., Chen, J., Soler, E., and Díaz, M. On blockchain and its integration with iot. challenges and opportunities. *Future generation computer systems*, **88** (2018), 173.

[105] Riyaz, S., Sankhe, K., Ioannidis, S., and Chowdhury, K. Deep learning convolutional neural networks for radio identification. *IEEE Communications Magazine*, **56** (2018), 146. doi:10.1109/MCOM.2018.1800153.

[106] Riyaz, S., Sankhe, K., Ioannidis, S., and Chowdhury, K. Deep learning convolutional neural networks for radio identification. *IEEE Communications Magazine*, **56** (2018), 146.

[107] Roberti, M. Rfid needs to be part of the building. In *IEEE RFID Journal*. IEEE (2018).

[108] Romero, H. P., Remley, K. A., Williams, D. F., and Wang, C.-M. Electromagnetic Measurements for Counterfeit Detection of Radio Frequency Identification Cards. *IEEE Trans. on Microwave Theory and Techniques*, **57** (2009), 1383.

[109] Safkhani, M., Peris-Lopez, P., Hernandez-Castro, J. C., and Bagheri, N. Cryptanalysis of the cho et al. protocol: A hash-based rfid tag mutual authentication protocol. *Journal of Computational and Applied Mathematics*, **259** (2014), 571.

[110] Sample, A. P., Yeager, D. J., Powledge, P. S., and Smith, J. R. Design of a passively-powered, programmable sensing platform for uhf rfid systems. In *2007 IEEE international Conference on RFID*, pp. 149–156. IEEE (2007).

[111] Samsung. Connected Washing Machine. `https://www.samsung.com/uk/laundry/washing-machine/?washing-machine` (2020).

[112] Sankhe, K., Belgiovine, M., Zhou, F., Angioloni, L., Restuccia, F., D'Oro, S., Melodia, T., Ioannidis, S., and Chowdhury, K. No radio left behind: Radio fingerprinting through deep learning of physical-layer hardware impairments. *IEEE Trans. on Cognitive Communications and Networking*, **6** (2019), 165.

[113] Sankhe, K., Belgiovine, M., Zhou, F., Riyaz, S., Ioannidis, S., and Chowdhury, K. ORACLE: Optimized Radio clAssification through Convolutional neuraL nEtworks. In *Proc. of IEEE Conference on Computer Communications (INFOCOM)* (2019).

[114] Sha, L., Rajkumar, R., Lehoczky, J., and Ramamritham, K. Mode change protocols for priority-driven preemptive scheduling. *Real-Time Systems*, **1** (1989), 243.

[115] Shorten, C. and Khoshgoftaar, T. M. A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, **6** (2019), 60.

[116] Soltani, N., Sankhe, K., Dy, J., Ioannidis, S., and Chowdhury, K. More is better: Data augmentation for channel-resilient rf fingerprinting. *IEEE Communications Magazine*, **58** (2020), 66.

[117] Song, X., Hu, S., Chen, D., and Zhu, B. Estimation of waste battery generation and analysis of the waste battery recycling system in china. *Journal of Industrial Ecology*, **21** (2017), 57.

[118] Spuri, M. Earliest deadline scheduling in real-time systems. *Doctorate Dissertation, Scuola Superiore S. Anna, Pisa*, **34** (1995).

[119] Stankovic, J. A. Research directions for the internet of things. *IEEE Internet of Things Journal*, **1** (2014), 3.

[120] Stankovic, J. A., Spuri, M., Ramamritham, K., and Buttazzo, G. C. *Deadline scheduling for real-time systems: EDF and related algorithms*, vol. 460. Springer Science & Business Media (2012).

[121] Stellios, I., Kotzanikolaou, P., Psarakis, M., Alcaraz, C., and Lopez, J. A survey of iot-enabled cyberattacks: Assessing attack paths to critical infrastructures and services. *IEEE Communications Surveys & Tutorials*, **20** (2018), 3453.

[122] Stoimenov, N., Perathoner, S., and Thiele, L. Reliable mode changes in real-time systems with fixed priority or edf scheduling. In *proceedings of the Conference on Design, Automation and Test in Europe*, pp. 99–104. European Design and Automation Association (2009).

[123] Stoimenov, N., Thiele, L., Santinelli, L., and Buttazzo, G. Resource adaptations with servers for hard real-time systems. In *Proceedings of the tenth ACM international conference on Embedded software*, pp. 269–278. ACM (2010).

[124] Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press (1998).

[125] Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press (2018).

[126] Sutton, R. S., Barto, A. G., et al. *Introduction to Reinforcement Learning*, vol. 135. MIT press Cambridge (1998).

[127] Swan, M. *Blockchain: Blueprint for a new economy*. " O'Reilly Media." (2015).

[128] Talla, V., Kellogg, B., Gollakota, S., and Smith, J. R. Battery-free cellphone. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **1** (2017), 1.

[129] Tan, C. C., Sheng, B., and Li, Q. Secure and serverless rfid authentication and search protocols. *IEEE Trans. on Wireless Communications*, **7** (2008), 1400.

[130] Tindell, K. W., Burns, A., and Wellings, A. J. Mode changes in priority preemptively scheduled systems. In *Real-Time Systems Symp.* IEEE (1992).

[131] Union, E. European union report (2020). Available from: `https://ec.europa.eu/knowledge4policy/foresight/`

topic/accelerating-technological-change-hyperconnectivity/hyperconnectivity-iot-digitalisation_en.

[132] Vo-Huu, T. D., Vo-Huu, T. D., and Noubir, G. Fingerprinting Wi-Fi Devices Using Software Defined Radios. In *Proc. of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pp. 3–14. ACM (2016).

[133] Wang, H., Wang, Y., Cao, Z., Li, Z., and Xiong, G. An overview of blockchain security analysis. In *Cyber Security*, pp. 55–72. Springer, Singapore (2019). ISBN 978-981-13-6621-5.

[134] Wang, J., Abari, O., and Keshav, S. Challenge: Rfid hacking for fun and profit. In *Proc. of the 24th Annual International Conference on Mobile Computing and Networking*, pp. 461–470 (2018).

[135] Want, R. Enabling ubiquitous sensing with rfid. *Computer*, **37** (2004), 84.

[136] Welbourne, E., Battle, L., Cole, G., Gould, K., Rector, K., Raymer, S., Balazinska, M., and Borriello, G. Building the Internet of Things using RFID: the RFID Ecosystem Experience. *IEEE Internet computing*, **13** (2009), 48.

[137] Whitmore, A., Agarwal, A., and Da Xu, L. The internet of things—a survey of topics and trends. *Information Systems Frontiers*, **17** (2015).

[138] Wood, G. et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, **151** (2014), 1.

[139] Xie, F., Wen, H., Wu, J., Hou, W., Song, H., Zhang, T., Liao, R., and Jiang, Y. Data Augmentation for Radio Frequency Fingerprinting via Pseudo-Random Integration. *IEEE Transactions on Emerging Topics in Computational Intelligence*, **4** (2020), 276. doi:10.1109/TETCI.2019.2907740.

[140] Xu, K., Wang, X., Wei, W., Song, H., and Mao, B. Toward software defined smart home. *IEEE Communications Magazine*, **54** (2016), 116.

[141] Xu, Q., Zheng, R., Saad, W., and Han, Z. Device Fingerprinting in Wireless Networks: Challenges and Opportunities. *IEEE Communications Surveys & Tutorials*, **18** (2016), 94.

[142] Yang, Y., Wu, L., Yin, G., Li, L., and Zhao, H. A survey on security and privacy issues in internet-of-things. *IEEE Internet of Things Journal*, **4** (2017), 1250.

[143] YaronTech. AZ 9662 Alien H3 73.5x21.2mm UHF tag RFID Adhesive Tag Inlay RFID Label (2020).

[144] Yeager, D. J., Sample, A. P., and Smith, J. R. Wisp: A passively powered uhf rfid tag with sensing and computation. In *RFID Handbook*, pp. 261–276. CRC Press (2017).

[145] Zanetti, D., Danev, B., and Capkun, S. Physical-layer Identification of UHF RFID tags. In *Proc. of the ACM International Conference on Mobile Computing and Networking (ACM Mobicom)*, pp. 353–364 (2010).

[146] Zanetti, D., Sachs, P., and Capkun, S. On the Practicality of UHF RFID Fingerprinting: How Real is the RFID Tracking Problem? In *International Symposium on Privacy Enhancing Technologies Symposium*, pp. 97–116. Springer (2011).

[147] Zhang, H., Gummeson, J., Ransford, B., and Fu, K. Moo: A batteryless computational rfid and sensing platform. *University of Massachusetts Computer Science Technical Report UM-CS-2011-020*, (2011).

[148] Zhang, P. and Ganesan, D. Enabling bit-by-bit backscatter communication in severe energy harvesting environments. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, pp. 345–357 (2014).

[149] Zhang, P., Gummeson, J., and Ganesan, D. Blink: A high throughput link layer for backscatter communication. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 99–112 (2012).

[150] Zhang, T., Becker, N., Wang, Y., Zhou, Y., and Shi, Y. Bitid: Easily add battery-free wireless sensors to everyday objects. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 1–8. IEEE (2017).

[151] Zhang, Y., Kasahara, S., Shen, Y., Jiang, X., and Wan, J. Smart contract-based access control for the internet of things. *IEEE Internet of Things Journal*, **6** (2018), 1594.

[152] Zheng, Z., Xie, S., Dai, H., and Wang, H. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE International Congress on Big Data (BigData Congress)*. IEEE (2017).

[153] Zhou, S., Zhang, Z., Luo, Z., and Wong, E. C. A lightweight anti-desynchronization rfid authentication protocol. *Information Systems Frontiers*, **12** (2010), 521.