Symbiosis of smart objects across IoT environments

688156 - symbIoTe - H2020-ICT-2015

D4.2 – SymbIoTe Middleware Implementation

The symbIoTe Consortium

Intracom SA Telecom Solutions, ICOM, Greece
Sveučiliste u Zagrebu Fakultet elektrotehnike i računarstva, UNIZG-FER, Croatia
AIT Austrian Institute of Technology GmbH, AIT, Austria
Nextworks Srl, NXW, Italy
Consorzio Nazionale Interuniversitario per le Telecomunicazioni, CNIT, Italy
ATOS Spain SA, ATOS, Spain
University of Vienna, Faculty of Computer Science, UNIVIE, Austria
Unidata S.p.A., UNIDATA, Italy
Sensing & Control System S.L., S&C, Spain
Fraunhofer IOSB, IOSB, Germany
Ubiwhere, Lda, UW, Portugal
VIPnet, d.o.o, VIP, Croatia
Instytut Chemii Bioorganicznej Polskiej Akademii Nauk, PSNC, Poland
NA.VI.GO. SCARL, NAVIGO, Italy
Universität Zürich, UZH, Switzerland

For more information on this document or the symbIoTe project, please contact:
Sergios Soursos, INTRACOM TELECOM, souse@intracom-telecom.com

Document Control

Title: SymbIoTe Middleware Implementation

Type: Public

Editor(s): Daniele Croce

E-mail: daniele.croce@cnit.it

Author(s): CNIT

Doc ID: D4.2-v1.4

Amendment History

Version	Date	Author	Description/Comments
v0.1	March 30, 2017	Daniele Croce (CNIT)	Initial Table of Contents.
v0.3	October 23, 2017	Daniele Croce (CNIT)	ToC finalized.
v0.4	November 30, 2017	Daniele Croce (CNIT), Matteo Pardi (NWX), Konrad Leszczyński (PSNC), Matteo Di Fraia (UNIDATA), Pavle Skocir (UNIZG), Vasileios Glykantzis (ICOM)	Partners' initial contributions, added Innkeeper API implementation description, Revisited ToC.
v0.5	December 14, 2017	Daniele Croce, Giuseppe Piro and Pietro Tedeschi (CNIT), Mikolaj Dobski (PSNC)	Added security considerations and section on wireless technologies.
v0.6	December 20, 2017	Mikolaj Dobski (PSNC), Matteo Di Fraia (UNIDATA), Pavle Skocir (UNIZG), Daniele Croce and Fabrizio Giuliano (CNIT)	Completed sections 4.1.6 and conclusions. Updated component features.
v0.7	December 22, 2017	Matteo Pardi (NWX), Pavle Skocir (UNIZG), Daniele Croce and Fabrizio Giuliano (CNIT)	Fixed some figures and component payloads.
v0.8	January 9, 2018	Daniele Croce and Fabrizio Giuliano (CNIT), Pavle Skocir (UNIZG), Zvonimir Zelenika (VIP)	Overall deliverable review.
v1.0	January 13, 2018	Matteo Di Fraia (UNIDATA), Ilenia Tinnirello, Daniele Croce and Fabrizio Giuliano (CNIT), Zvonimir Zelenika (VIP)	Sym-agent and Innkeeper interfaces updated. Document cleanup.
v1.1	January 15, 2018	Daniele Croce (CNIT)	Revised version.
v1.2	January 22, 2018	Zvonimir Zelenika (VIP)	Overall check, added list of figures and tables.
v1.3	January 30, 2018	Matteo Pardi (NWX), Michael Jacoby (IOSB), Daniele Croce, Gennaro Boggia and Giuseppe Bianchi (CNIT)	Added figure of SSP architecture and MIM semantics. Document finalization.
v1.4	February 9, 2018	Mikolaj Dobski (PSNC), Daniele Croce and Pietro Tedeschi (CNIT)	Revised SSP security.

Table of Contents

List of Figures

List of Tables

1 Executive Summary

This deliverable presents the implementation of key features for the SymbIoTe Smart Space Middleware (S3M), involving the two lower domains of SymbIoTe Smart Space (SSP) and Smart Device (SDEV). It also reports on advanced wireless management in WiFi and LoRa networks, two popular technologies for our scenarios of interest.

The objectives pursued in the design and implementation of the key S3M features presented here are the automatic registration of resources, the IoT platform/device interoperability at the SSP level (platform transparency) and the security components necessary for such operations. It is important for the SSP to be able to provide its functionality even when the cloud components are not reachable, in order to allow the user to interact with the SSP in every kind of connectivity conditions. For example, in the Smart Residence or Smart Yachting Use Cases it is essential that devices keep working (i.e. they can be read and they can perform operations) even if it is temporarily (or even permanently) not possible to communicate with the Cloud.

The definition of software components for the S3M provides symbIoTe compliant gateways and devices with the functions required inside the SSP, locally implementing the mechanisms of resources registration and access by replicating the paradigm adopted in the Cloud domain. Therefore, the design of the S3M builds on the analysis of the relationships with the Core and the Cloud components and the complex interactions of software modules both inside and outside the SSP.

The later part of the deliverable is focused on wireless technologies management because SSPs and SDEVs are often equipped with one or more widespread wireless networks, such as WiFi, ZigBee, LoRa, etc., all operating in the same unlicensed bands. The proliferation of platforms based on those technologies is creating coordination problems among independent platforms, which may be faced by the symbIoTe middleware. Thus, the SSP may face the problem of identifying the coexisting networks and allow fast reconfigurations and integration of multiple network interfaces.

In particular, we focused on WiFi and LoRa networks, studying how to correctly identify coexisting networks operating in the same environment, measuring the impact of intra- and inter-technology interference, and offering coordination mechanisms to grant efficient network access to all connected smart objects.

Regarding WiFi, we show we can recognize inter-technology interference by off-the-shelf WiFi devices by monitoring the statistics of receiver errors. We thus develop an Artificial Intelligence tool (Artificial Neural Network) to recognize the source of interference, reaching an average accuracy of almost 99% in recognizing ZigBee, Microwave and LTE-U interference. We then show how to exploit inter-technology interference to set-up a low-rate bi-directional communication channel between heterogeneous technologies, such as WiFi and ZigBee co-existing in same space. This unconventional communication channel can be very useful for coordinating channel access between the wireless technologies, but also for other direct link cross-technology applications.

In LoRa we study the impact of imperfect-orthogonality in LoRa spreading factors (SFs) in simulation and real-world experiments. We also implemented a LoRa cell traffic generator, able to emulate the behavior of thousands of low-rate sensor nodes deployed in the same cell. Implications of this imperfect orthogonality for network planning are still under investigation.

2 Introduction

This deliverable presents the implementation of the SymbIoTe Smart Space Middleware, with all its features and functionalities. The main purpose of an SSP is to accommodate more interoperable IoT platforms and devices, allowing a sort of liaison from the cloud to the local components.

The deliverable also reports on wireless management of WiFi and LoRa networks. In particular, we focus on coexistence of technologies in unlicensed ISM (Industrial, Scientific and Medical) bands used by WiFi, ZigBee, LTE-U, etc., and how to recognize such interference from the WiFi nodes. We also show how this interference can be exploited to set-up an inter-technology communication channel between WiFi and ZigBee, which can be very useful both for coordinating channel access (i.e. reducing interference) and other direct link applications using just common smartphones or laptops which are only equipped with WiFi interfaces. Finally, we studied the impact of imperfect-orthogonality in LoRa spreading factors (SFs) and developed a LoRa cell traffic generator, able to emulate the behavior of thousands of low-rate sensor nodes deployed in the same cell.

2.1 Purpose and scope of the document

The aim of Deliverable 4.2 is to describe the implementation of the Smart Space and Smart Device domains, mapping the mechanisms already considered for the Application Domain and Cloud Domain, and analyzing how we can guarantee the same features also at a lower level. The deliverable also describes how to cope with some wireless communication issues that will possibly arise in future IoT scenarios.

The document reports on the technical work performed in two tasks, T4.2 (Wireless network virtualization and management, M9–M26) and T4.3 (Implementation of smart space middleware and client, M13–M30), which are currently still in progress.

2.2 Relation to other tasks and deliverables

The content of this deliverable is related to the outcome of Task 4.1 (Local Registration, Discovery and Interoperability of Smart Objects) which has previously defined the general design of the SSP as reported in deliverable D4.1 “symbIoTe middleware tools, protocols and core mechanisms”. In addition, some components of the symbIoTe SSP and SDEV domains are reused from symbIoTe Core Services, presented in deliverable D2.2 “symbIoTe Virtual IoT Environment Implementation”. Finally, since the SSP interacts with L1 clients, the security mechanisms in SSP are handled by an extended version of a “platform” AAM (Authentication and Authorization Manager), as described in D1.4.

2.3 Deliverable Outline

This document structure is as follows: Section 1 is the executive summary and Section 2 the introduction. An overview of SSP and SDEV Domains is provided in Section 3, where challenges and solutions are highlighted and explaining relationships with Core and Cloud Domains. Section 4 describes the SSP security considerations while Section 5 describes the middleware components implemented for SSP and SDEV. Section 6 presents the wireless technology solutions for WiFi and LoRa networks and Section 7 finalizes the document, reporting its conclusions. Additional implementation details are provided in Appendix.

3 Architecture overview

3.1 Smart Space Architecture

SymbIoTe Smart Spaces are environments (residence, campus, vessel, stadium, city area …) where one or more IoT platforms provide services. In order to integrate such environments into symbIoTe, we need to deploy proper software adapters (that we generically refer to as symbIoTe Smart Space Middleware, or S3 Middleware).

Some IoT platforms have a split local / cloud architecture, whereas others have only local or only cloud components. Depending on each IoT platform's architecture, the S3 Middleware is deployed either as a cloud component or as one of the Smart Space's appliances (or part thereof). Since the idea is to follow an approach as general as possible, the aim would be designing a software architecture that can be deployed in either way with no significant modifications. The functionality is duplicated at various domains, in Cloud Domain and Smart Space Domain (e.g. device management in the CLD and SSP): what would differ is the scope and the available hardware.

More specifically, the goal is:

- to keep the same architecture for software components breakdown, interfaces and communications paradigm
- to maximize code reuse (e.g. by creating libraries)
- to reuse entire software components, if possible
- to keep components as modular as possible, to be able to just plug platform specific code for each specific IoT platform

The SSP as a whole will expose (i.e. register, provide access to) the resources it contains, regardless of which "local" IoT platform they belong to. Therefore, SDEVs associated to the SSP will also be exposed directly, without being "mediated" by any of the local platforms. SymbIoTe compliant local IoT platforms within an SSP will be able to access all the resources associated to that SSP, provided that the required authentication and authorization policies are in place. This includes both resources provided by any co-located IoT platforms, and those provided by the locally associated SDEVs. This way, the interoperability role played by symbIoTe will be fully functional also at the SSP level. When more than one IoT platform is active in an SSP, the S3M shall thus be acting as a local resource interchange.

Since each IoT Platform manages its own device's resources according to its internal protocols (which shall remain opaque for symbIoTe), discovery and management of those resources will be in charge of the Platform itself. It will be the Platform's duty to register and de-register its device's resources with the S3M whenever a change happens. The S3M will then publish information about the Platform's resources in the same way it does for symbIoTe's SDEVs resources. Hence, SDEV resources will not be distinguishable from the resources of a native platform.

Smart Spaces must be able to accommodate both incoming apps (a user with a smartphone or tablet running a symbIoTe app) and incoming devices (symbIoTe Smart Devices). In both cases, the incoming entity should be identified, authorized and given a way to access the Smart Space's facilities. This must be possible even in case of temporary failure or degradation of Internet connectivity.

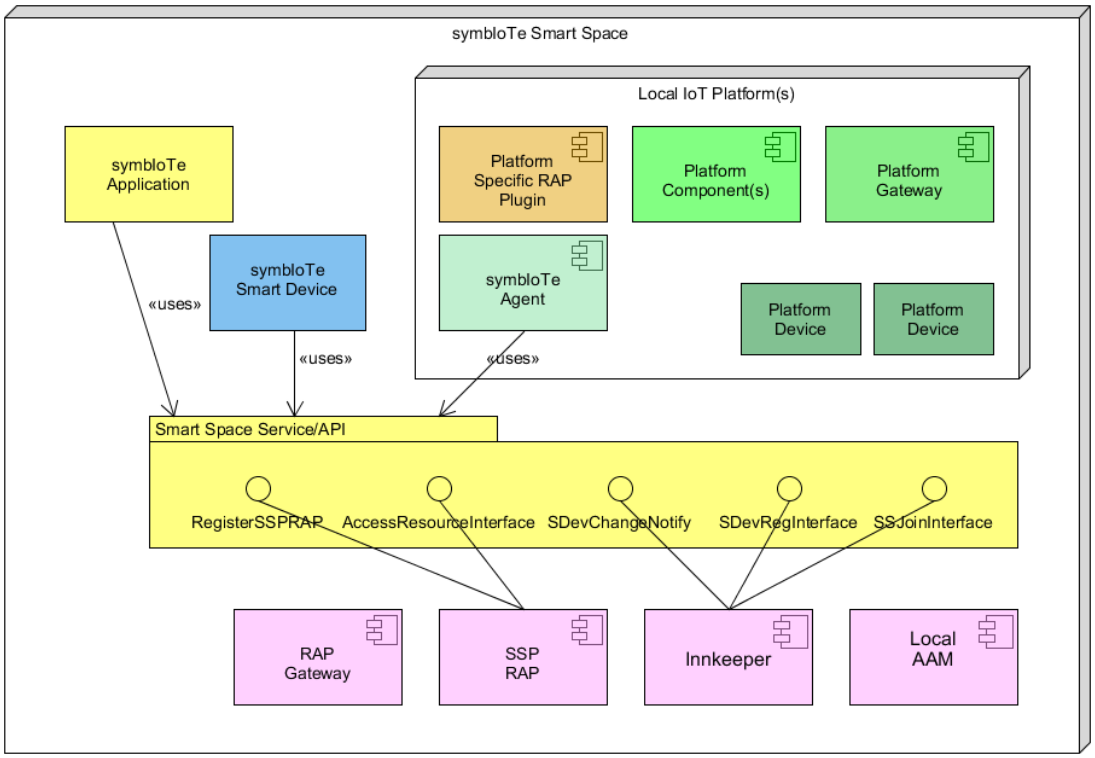


Figure 1: Architecture of Smart Space and Smart Device Domains

3.2 SymbIoTe middleware

To achieve the goal described previously, symbIoTe aims to equip the smart spaces with a software designed to integrate locally available devices or platforms; the symbIoTe middleware. The symbIoTe middleware is the software by which all the services of the Smart Space are originated. Its implementation must provide the following functions:

- interface with the core components, exposing local devices' resources for queries and actuations.
- interact with local smart devices and platforms providing a mean to connect and share new resources,
- maintain certain degree of autonomy allowing the middleware to function while no internet connection is available.

The middleware is designed to be implemented by three main components: The Innkeeper, the Resource access Proxy (RAP) and LocalAAM each one performing specific tasks as detailed below. Other minor components related to the SSP are: the RAP gateway (allowing access to local Smart Devices for clients that are outside the SSP) and the Platform and symbIoTe agents (the internal interface towards the SSP).

3.2.1 Innkeeper

The Innkeeper is the component in charge to receive registration from devices/platforms agents. It keeps the consistency of the Smart Space resources and communicates to the RAP for their reachability. Innkeeper is also in charge to communicate with the core level components to let the Smart Space integrate with the rest of the symbIoTe ecosystem.

3.2.2 Resource Access Proxy (RAP)

The RAP (Resource Access Proxy) receives and forwards request from and to the Smart Space. It is in charge for directing request to the intended resources.

3.2.3 LocalAAM

Local AAM (Authentication and Authorization Manager) is the guarantor of the symbIoTe security within the Smart Space. It enables the core centric security function while internet connection is established, but it becomes Smart Space centric when no internet connection is available (allowing the SSP to work also when disconnected).

3.3 Smart Devices

Most IoT devices are wirelessly connected to the Internet and possibly nomadic from one service platform to another. In the symbIoTe vision, a Smart IoT Device is a device in which a symbIoTe software component is installed in order to interact with a Smart Space. The interaction with the Smart Space is characterized by the physical space in which the Smart Device is located. Indeed, in such a physical space, the wireless medium and the technologies providing local connectivity represent some resources that need to be shared, in a coordinated manner, by the IoT platforms and the Smart Devices that are in radio visibility.

Moreover, the device contains additional resources, including sensing and actuating capabilities, applications, storage, etc., which may generally depend on the device type and can be accessed by exposing a symbIoTe-compliant API.

In order to define the architecture of this symbIoTe software component, we considered two main features for Smart Devices:

- ***Attaching to a smart space***, by configuring an authorized physical link to a selected gateway, in order to make available the device resources and participate to the service creation in the smart space.
- ***Roaming across heterogeneous SSPs****,* which are usually managed by heterogeneous providers.

The roaming concept for a Smart Device requires registering to different Smart Spaces but keeping the device identity in the symbIoTe Core and might not necessarily require a connectivity roam.

In other words, roaming across SSPs involve a logical registration of the smart device which can be completely decoupled by the attachment to the physical gateway.

3.4 Relationship with Core and Cloud components

Smart Spaces Middleware is a part of the symbIoTe ecosystem, and as such it needs to communicate with the upper layers, namely symbIoTe Cloud (CLD) and symbIoTe Core Services in the Application layer (APP). Access to the Smart Spaces can be provided to symbIoTe Smart Devices, defined as entities with no relation to a specific IoT platform, and to the IoT platforms by using dedicated Platform Agents.

When describing relationship of SSPs with Core and Cloud domains, we can distinguish between the following cases:

1. symbIoTe Smart Device as a part of the Smart Space,
2. Local IoT platforms as a part of the Smart Space,
3. symbIoTe Level-1 Compliant platforms as a part of the Smart Space, and
4. symbIoTe Level-2 Compliant platforms as a part of the Smart Space.

Each of the possible cases is described in more detail in the following subsections. The diagrams within this section show how resources are registered and accessed from the upper layers. They are simplified, as they do not show interactions related to security. WP4 will focus only on the first two cases, while the last two are merely considerations. It isn't planned in the use cases to have L1 and L2 platforms as a part of the SSP.

3.4.1 symbIoTe Smart Device

symbIoTe Smart Devices are defined as entities with no relation to a specific IoT platform. They rely on the functions provided by the symbIoTe middleware in any SSP and need to be accessible from the symbIoTe Core and by any symbIoTe app. They are registered in the Core Registry (with the assistance of the Innkeeper), and are searchable through Core Search component. The applications can access them through SSP RAP if the application is in the Smart Space, or through RAP Gateway if the application is outside the Smart Space. The interaction of Smart Devices with symbIoTe Core is shown in Figure 2.

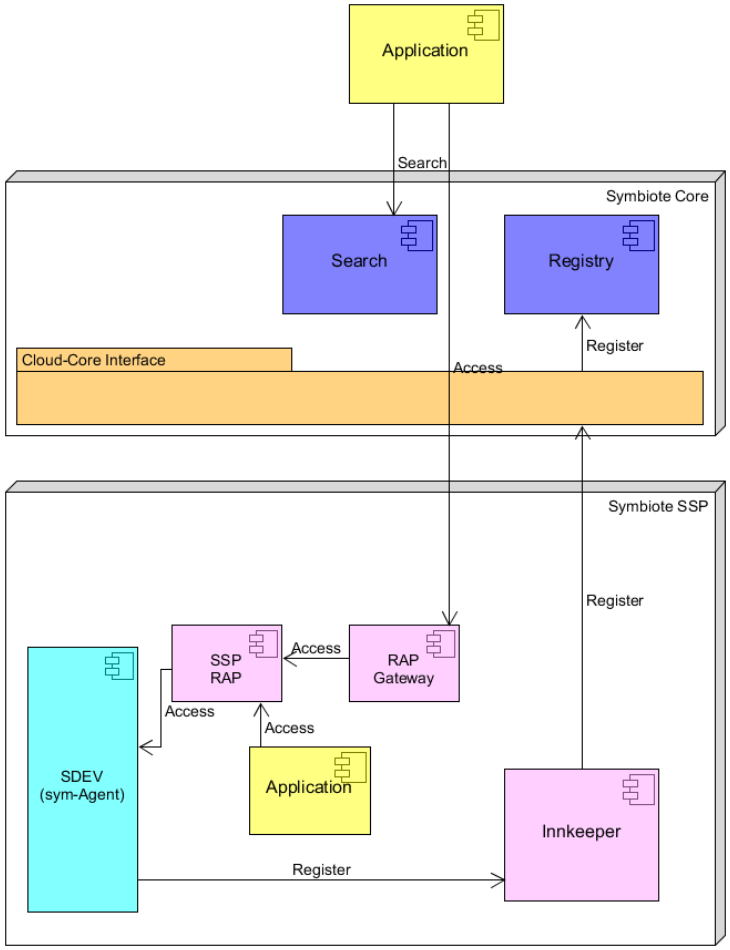


Figure 2: Relationship between SSP and APP domains for Smart Devices

3.4.2 Local IoT platform

Local IoT platforms in the context of symbIoTe are defined as platforms which do not contain Cloud layer and operate in a specified physical space. Such platforms can join symbIoTe Smart Space to share their resources to applications that would be interested in using them together with other locally available resources. In such a way symbIoTe Smart Space acts as local resource interchange at a certain physical space for IoT resources from local IoT platforms and Smart Devices.

Interaction between local IoT platforms and Smart Spaces is carried out through a Platform Agent, which has similar functionality as a Smart Device agent. Platform Agent, which is registered in the SSP, is granted necessary credentials to register its resources in the Smart Space and (if needed) in the Core Registry. As in the case of Smart Devices, access to local IoT platform is handled through SSP RAP (when application is in the Smart Space) or through RAP Gateway when the application is outside the Smart Space. The components interaction with the symbIoTe Core is shown in Figure 3.

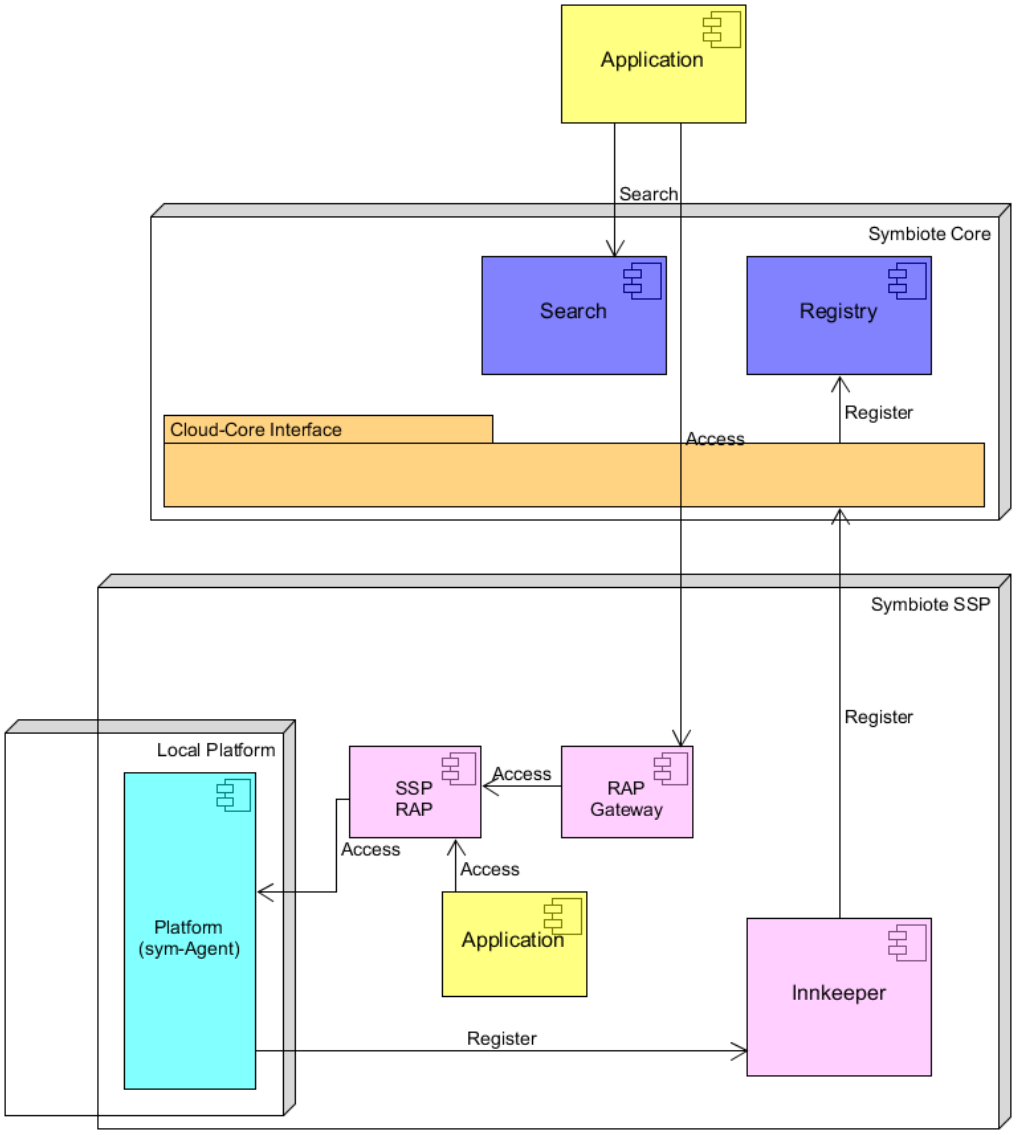


Figure 3: Relationship between SSP and APP domains for local IoT platforms

3.4.3 symbIoTe L1 platform

symbIoTe L1 platform can join a Smart Space. In such a case platforms have CLD components which enable interaction with the Core Services. Interactions with the Smart Spaces are possible through Platform Agent, as in the case with the local platforms. Resources can be registered in the symbIoTe Core through symbIoTe Cloud components and through symbIoTe Smart Space by using Platform Agent, as shown in Figure 4.

Platform Owner decides if the registration will be accomplished through Cloud components or Platform Agent. The choice should be made based on the functionalities of the symbIoTe framework a Platform Owner wants to employ. If the case when sharing resources within a physical environment is required, a Platform Owner should register the resources through the Platform Agent to the Smart Space. This case is marked as Option 1 in Figure 4. If registration to the Core suffices, making the resources discoverable to third party applications and developers only through Search component, the registration can be achieved through Registration Handler in Cloud. This case is marked as Option 2 in Figure 4. Depending on the registration choice, users can access the resources through RAP in CLD, or through RAP Gateway and SSP RAP in SSP. An application within the SSP can access resources shared by L1 platform through SSP RAP only if they are registered in the SSP.

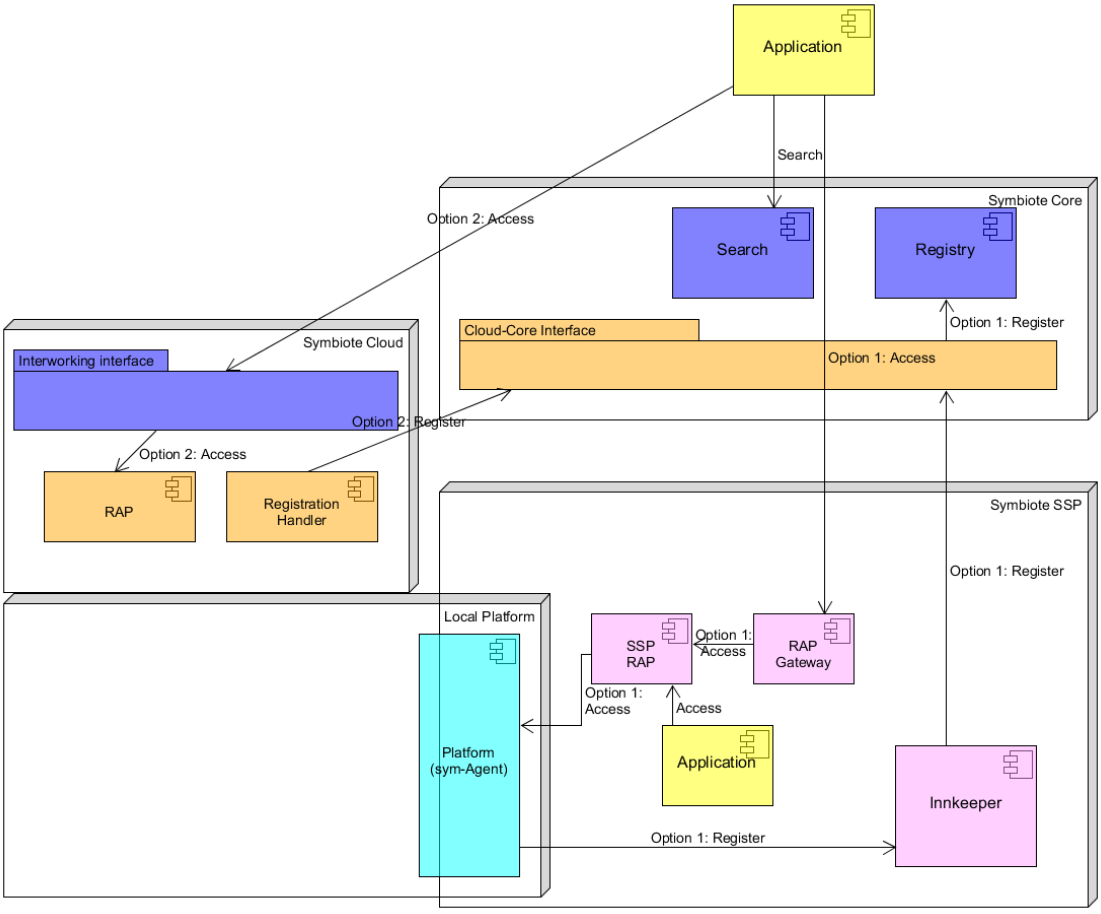


Figure 4: Relationship between SSP and APP domains for symbIoTe L1 platforms

3.4.4 symbIoTe L2 platform

symbIoTe L2 platforms that form symbIoTe federations can join a Smart Space. As it can be seen in Figure 5, symbIoTe CLD for L2 applications has an additional component, Platform Registry where information about the resources from the symbIoTe federations are stored. These resources are not registered to the symbIoTe Core. When a certain platform wants to share its resources to the federation, it firstly notifies the Platform Registry, which than forwards this information to other platforms in the federation. How this information is forwarded is not shown in Figure 5, since it is out of scope when considering relationships between L3/L4 with L2.

Platform Owner decides if the registration will be accomplished through Cloud components or Platform Agent. The choice should be made based on the functionalities of the symbIoTe framework a Platform Owner wants to employ. If the case when sharing resources within a physical environment is required, a Platform Owner should register the resources through the Platform Agent to the Smart Space. This case is marked as Option 1 in Figure 5. If a Platform Owner wants to exploit symbIoTe federation functionalities, e.g. reaching a Service Level Agreement (SLA) or make use of bartering mechanism for sharing resources between platforms, registration through Registration Handler to Platform Proxy is necessary. This case is marked as Option 2 in Figure 5.

Depending on the registration choice, users can access resources through RAP in CLD, or through RAP Gateway and SSP RAP in SSP. An application within the SSP can access resources through SSP RAP only if they are registered in the SSP. Native L2 applications can find the wanted resources through their own Platform Registry, this interaction is not shown in Figure 5. Native L2 applications can access only the resources which are shared by Registration Handler of the L2 platform.

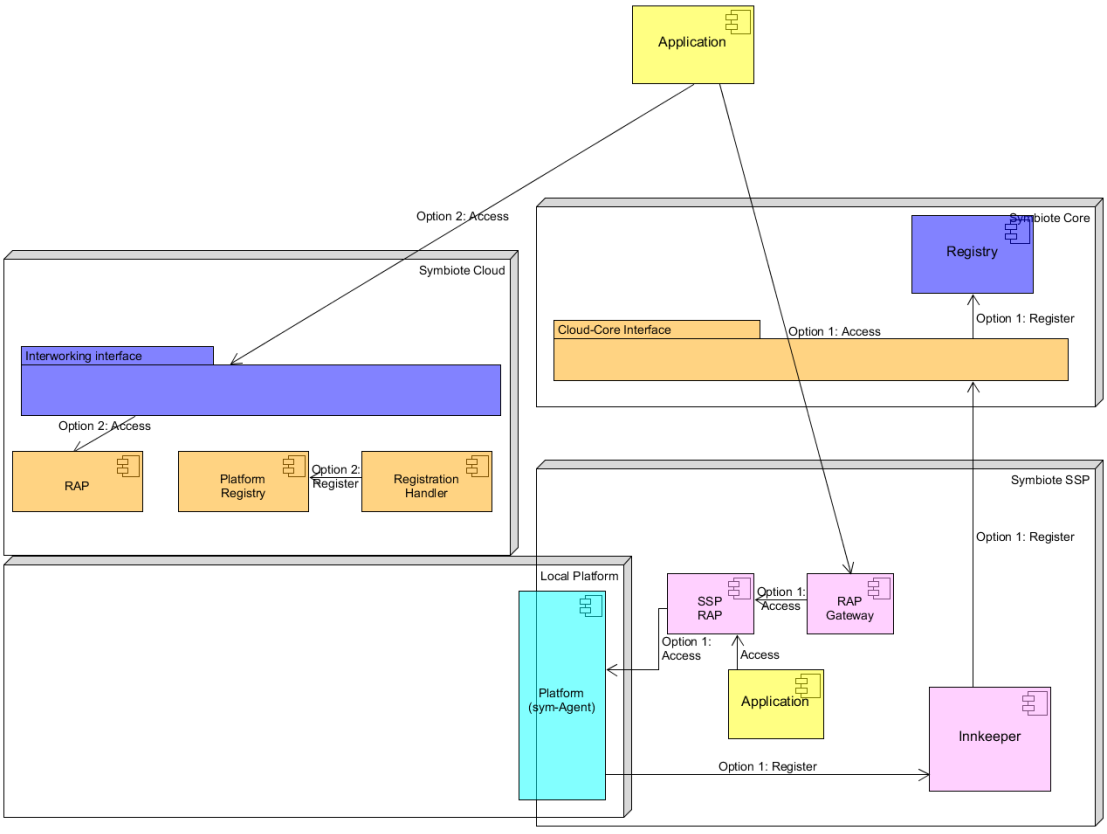


Figure 5: Relationship between SSP and APP domains for symbIoTe L2 platforms

4 SSP Security considerations

4.1 Scenario

Targeted security services for the smart space, to be implemented between SDEV, IoT Platforms, User/Application(s) and SSP Middleware, include:

1. Users' Authentication and Authorization
2. Authentication of SDEV directly connected to SSP middleware
3. Data Confidentiality Integrity and Authentication (CIA) between SSP middleware and Agent (of both SDEV and IoT platform)

SDEV and IoT Platforms can interact with SSP middleware through an agent. The SSP owner's is able to configure the SSP itself, including resource policies and the AAM database where users are defined. If the SSP integrates IoT platform controlled by third-party entities, the middleware introduces *proxy functionalities* for handling access control mechanisms.

Furthermore, this section proposes a solution that enables and manages lightweight security aspects for interactions between very constraint smart devices and SSP middleware, where it is necessary to guarantee a minimal communication baseline security. Instead, communication between users and SSP middleware, can be protected by adopting the techniques and security mechanisms already defined in the State of the Art.

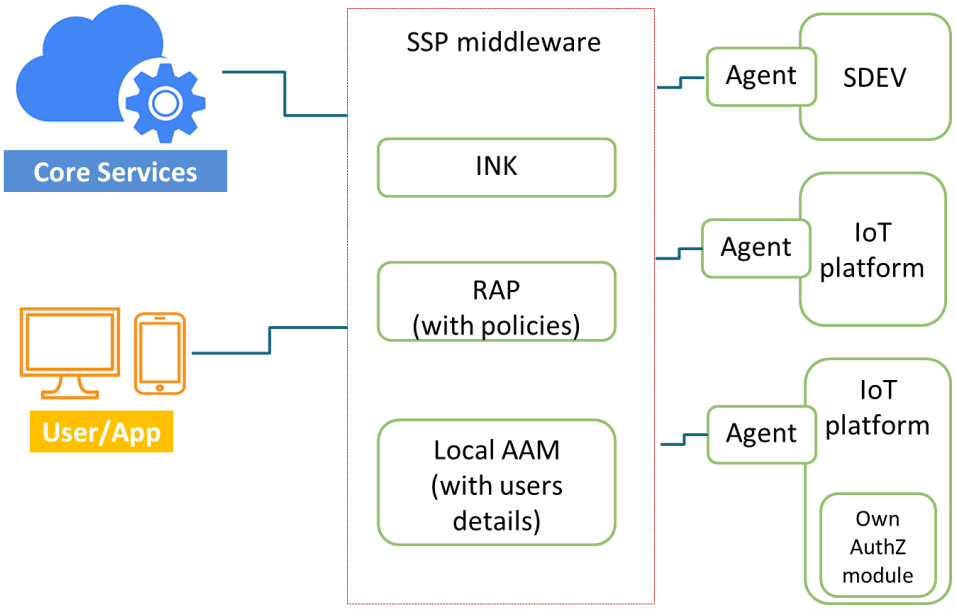


Figure 6: SSP Security Architecture

The Smart Space is under the control (e.g., can be configured) by the SSP owner. SSP middleware integrates the functionalities of Innkeeper, RAP and Local AAM. From one side Core Services and User/Application(s) are defined, while from the other side, Smart Devices and IoT Platforms are exposed as shown in Figure 6. SSP middleware exposes resources of:

- SDEV, directly connected to the middleware
- IoT platforms governed by the SSP owner
- IoT platforms governed by third-party entities

These entities can interact with SSP middleware through an agent.

4.2 System configuration

The SSP owner, can create users (e.g. by defining username and password or by exposing the users' registration form) and assign users' properties (i.e. attributes) in the Local AAM. For a resource under its (SSP's) control, the SSP owner defines access policies with the RAP in order to permit or deny access to resources.

For resources governed fully by third-party IoT platforms attached to the SSP we are proposing to reuse and extend the challenge-response mechanism defined in L1/L2. For 'L3 clients' Symbiote will offer a well-documented hashing algorithm for string concatenating a secret (delivered to the client through a 3rd party channel, therefore unknown in the symbiote ecosystem), the username and/or its client identifier along with the operation timestamp. This way the Platform Agent's authorization extension will be able to recreate the hash on its side and verify the match. The username/clientID and timestamp will be delivered as already implemented in the L1/L2 Security Request payload.

4.3 Users' Authentication and Authorization

Security services provided for the users, will be defined by reusing the conventional L1 security functionalities.

- The user logins to the Local AAM and receives a token with its attributes
- The user delivers the token to the RAP in the SSP middleware
- The RAP verifies locally that the token is authentic (also offline)
- The resource is provided if the set of attributes matches the access policy
    - If the resource is exposed by IoT platforms managed by third-party entities, the final access control is demanded to the own Authorization module of the platform using the secret hashing mechanism described above.

4.4 Authentication and configuration of security services within the SSP

Security services for the smart space, to be implemented between device and gateway/innkeeper, include:

1. Algorithm Negotiation
2. Peer Authentication
3. Key Agreement
4. Protection from attacks, including replay

The HTTP [1] protocol is used at the application layer. But TLS-based solutions [2] cannot be used between SDEV and SSP middleware. In fact, constrained devices belonging to

the smart space present limited processing power, storage space, and transmission capabilities. Lightweight solutions natively conceived for CoAP cannot be directly applied in our case. For this reason, an innovative approach is designed for symbIoTe.

4.4.1 Configuration and Negotiation of security services

With reference to the first 4 goals reported above, a lightweight protocol was conceived. It includes the following main phases:

- Peer Configuration (done by the network administrator)
- Negotiation of cryptographic techniques
- Key Agreement
- Peer Authentication (mutual)

The protocol ensures a secure interaction: negotiated secrets is unavailable to eavesdroppers, even by an attacker who can place himself in the middle of the connection. Moreover, in the case the protocol ends successfully, communicating peers can protect their communication through symmetric cryptography (e.g., AES, ChaCha20, etc.) and reliable mechanisms (i.e., messages include an authentication tag which protects them against tampering). Indeed, the protocol provides in output all the details needed to support the data confidentiality/authentication/integrity.

4.4.2 Negotiation Protocol

Negotiation is initiated by the device. Device and gateway/innkeeper agree on the cipher suite (i.e., cryptographic algorithms to use), negotiate and/or generates key material, and provide a proof of their authenticity. The protocol is designed to jointly support simple approaches, like pre-shared symmetric key (PSK), or more complex ones, like Elliptic-Curve Diffie-Hellman (ECDH) with RSA or ECDSA (when a certificate is used) exchange modes. The key agreement mechanism is chosen based on device capabilities. As commonly accepted, key materials are generated through a Key Derivation Function (KDF).

Let Security Context be the set of security parameters useful to setup security services.

Figure 7 provides a big picture of the negotiation protocol. Let Message Type Indicator (MTI) be a field that identifies the type of message. It may assume the following values:

- **0x10**: SDEV Hello
- **0x20**: GW_INK Hello
- **0x30**: SDEV AuthN
- **0x40**: GW_INK AuthN

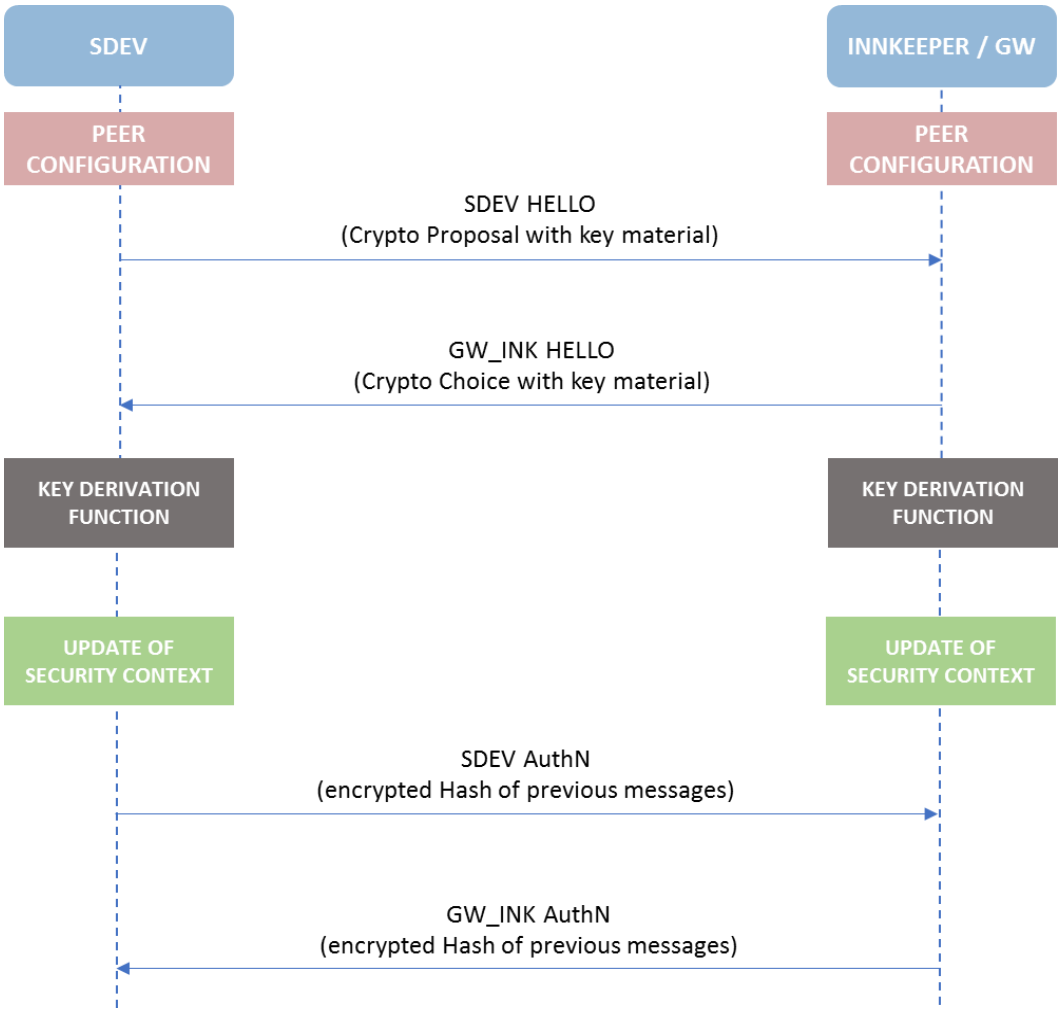


Figure 7: Protocol Message flow

Pre-Shared Key Configuration

A unique PSK for each device must be defined to circumvent the problem of a compromised PSK in a smart space. Only one key natively must be stored at the Gateway/Innkeeper side. Three different levels of security can be identified.

Basic level

- GW/INK stores a Master Secret Key (MSK), unique with the symbIoTe environment and shared among all SSPs
- Every device directly stores the PSK = $H(MAC_{ADDR}||MSK)$, where H is a hashing function
- GW/INK calculates PSK in real time

Intermediate level

- GW/INK stores a Master Secret Key (MSK), defined by the SSP owner for a given SSP. The core should know the MSK assigned to each SSP
- Each device belonging to the considered SSP is configured in order to store the PSK = $H(MAC_{ADDR}||MSK)$, where H is a hashing function

- GW/INK calculates PSK in real time
- In case of roaming, GW/INK of the new SSP could contact the core for obtaining the unique PSK assigned to a given device

Advanced level

- Each device is configured with a unique PSK.
- The PSK is stored in GW/INK and core by the device owner
- In case of roaming, GW/INK could contact the core for obtaining the unique PSK assigned to a given device

Despite the presence of multiple security levels, only the basic level will be implemented. Other levels of security can be easily integrated in the future.

SDEV Hello

It is sent in plaintext by SDEV. It contains the related MTI code (that is MTI=0x10), the MAC address of the SDEV (namely $SDEV_{MAC}$), the supported Cipher Suites (namely $CRYPTO_{PROPOSAL}$) that contain a list with the related Cipher Suite IDs (It is recommended to use the ID defined by IANA for TLS Parameters [3]), the preferred Key Derivation Function (namely $KDF_{PROPOSAL}$) and a cryptographic nonce (namely $SDEV_{NONCE}$).

Optionally additional key material (namely $SDEV_{MATERIAL}$) is present when Public Key Cryptography is implemented. It stores a X.509 certificate.

$SDEV_{MAC}$ is used for identifying the Smart Device.

$CRYPTO_{PROPOSAL}$ describes the list of cryptographic algorithms supported by SDEV. They are expressed according to the following structure:

- Key Exchange Algorithm (to establish a mechanism by which both parties will negotiate a key to communicate authentically)
- Symmetric Encryption Algorithm (to encrypt the messages)
- Message Authentication Code Algorithm (to create a message digest of message)

Note, more than one proposal can be reported within the list. The resulting protocol is, indeed, flexible. Possible examples include:

- **PSK_WITH_AES_128_GCM_SHA256**, where PSK is used to set a pre-shared key, AES as symmetric encryption algorithm with a 128-bit key, SHA256 as a pseudorandom function (PRF) based on HMAC with the SHA-256 hash function
- **PSK_WITH_CHACHA20_POLY1305_SHA256**, where PSK is used to set a pre-shared key, CHACHA20 as symmetric cipher with a 256-bit key, POLY1305 as a message authentication code that requires a 256-bit key and a message and produces a 128-bit tag

- **PSK_WITH_AES_128_CBC_SHA**, where PSK is used to set a pre-shared key, AES as symmetric encryption algorithm with a 128-bit key, SHA as a PRF based on HMAC with the SHA hash function or as a message authentication code algorithm
- **ECDH_ECDSA_WITH_AES_128_CBC_SHA**, which requires that the GW/INK certificate contains an ECDH-capable public key signed with ECDSA (both device and Gateway/Innkeeper perform an ECDH operation and use the resultant shared secret as the premaster secret), AES as symmetric encryption algorithm with a 128-bit key, SHA as a PRF based on HMAC with the SHA hash function or as a message authentication code algorithm
- **ECDH_ECDSA_WITH_AES_256_CBC_SHA**, which requires that the GW/INK certificate contains an ECDH-capable public key signed with ECDSA (both device and GW/Innkeeper perform an ECDH operation and use the resultant shared secret as the premaster secret, AES as symmetric encryption algorithm with a 256-bit key, SHA as a PRF based on HMAC with the SHA hash function or as a message authentication code algorithm

The $KDF_{PROPOSAL}$ field proposes the Key Derivation Function (KDF) to be used for generating session keys. Possible proposals include *PBKDF2* or *HKDF*. $SDEV_{NONCE}$ is used to protect the communication from replay attack together with GW_INK_{NONCE}. It builds the salt value in the KDF function. Optionally, $SDEV_{MATERIAL}$ is present when Public Key Cryptography is implemented. Therefore, it stores a X.509 certificate.

Authentication Encrypted with Additional Data (AEAD)

Some encryption algorithm like AES_128 with GCM or CHACHA20_POLY135, support AEAD. AEAD stands for “Authenticated Encryption with Additional Data” meaning there is a built-in message authentication code for integrity checking both the ciphertext and optionally additional authenticated (but unencrypted) data. In this protocol, $SDEV_{MAC}$||*sequence_number* is used as Additional Authenticated Data (AAD). The sequence number is the number of messages sent since the last handshake. It is incremented by 1 for each message. The sequence number could be start from a value obtained by the $SDEV_{NONCE}$ and GW_INK_{NONCE} sum, and it is used also to avoid replay attacks. In fact, for AEAD-less encryption algorithms, a sequence number field (*sn*) must be defined.

GW_INK Hello

As soon as the SDEV Hello is received, gateway or innkeeper verifies that $SDEV_{NONCE}$ is acceptable, $SDEV_{MAC}$ is stored within a database, and $CRYPTO_{PROPOSAL}$ contains an acceptable proposal. Then, it selects the most suitable Cipher Suite and sent back a new message containing the related MTI code (that is MTI=0x20), the selected Cipher Suite (namely $CRYPTO_{CHOICE}$), an optional Initialization Vector (IV), the nonce (namely GW_INK_{NONCE}), and optionally additional key material (namely $GW\_INK_{MATERIAL}$). $CRYPTO_{CHOICE}$ uses the same structure of the $CRYPTO_{PROPOSAL}$. IV can be used along with a secret key for data encryption. GW_INK_{NONCE} is used to protect the communication from replay attack and together with $SDEV_{NONCE}$. It builds the salt value in the KDF function. Optionally, $GW\_INK_{MATERIAL}$ is present when Public Key Cryptography is implemented. Therefore, it stores a X.509 certificate.

At this moment, SDEV and gateway or innkeeper calculates symmetric keys, according to the algorithm negotiated before. Two examples are reported later. What however is important to remark is that communicating peers will calculate the following keys:

- DK_1: derived key used to provide data confidentiality
- DK_2: derived key used to provide data authenticity

SDEV AuthN

This message is sent by SDEV that means the negotiation is completed and that the cipher suite is activated. It contains the related MTI code (that is MTI=0x30), the nonce (namely $SDEV_{NONCE\_2}$) to prevent replay attacks, and the encrypted hash of all the previous exchanged *Hello* hash messages combined, in both directions. It should be encrypted, since the negotiation is successfully done. After that the message is sent to GW/Innkeeper, the GW/Innkeeper can decrypt it and check if the received hashes match the calculated hashes.

GW_INK AuthN

This message is sent by GW/Innkeeper that means the negotiation is completed, that the cipher suite is activated. It contains the related MTI code (that is MTI=0x40), the nonce (namely $GW\_INK_{NONCE\_2}$) to prevent replay attacks, and the encrypted hash of all the previous exchanged *Hello* hash messages combined, in both directions. It should be encrypted, since the negotiation is successfully done. After that the message is sent to SDEV, the SDEV can decrypt it and check if the received hashes match the calculated hashes. At this point, SDEV and gateway/Innkeeper are authenticated.

4.5 Data Confidentiality

In this section an approach to guarantee data confidentiality/authentication/integrity for the messages exchanged between two entities is described. This mechanism will use the negotiated previous keys obtained during the key negotiation phase. The encrypted messages can be exchanged by using HTTP Protocol, including an object (i.e. *JSON* [4]) in the HTTP Message Body. If the AEAD algorithm is used, the payload is only encrypted by using DK_1.

$$\mathbf{ENC\_DATA = ENC_{DK1}\ (Data||}\boldsymbol{sequence\_number}\mathbf{)}$$

Otherwise, an HMAC signature must be calculated by using DK_2:

$$\mathbf{ENC\_DATA = ENC_{DK1}\ (Data||}\boldsymbol{sequence\_number}\mathbf{)}$$

$$\mathbf{SIGNATURE = HMAC_{DK2}\ (ENC\_DATA||}\boldsymbol{sequence\_number}\mathbf{)}$$

Key Material Derivation

The session key is derived through the KDF. Two examples are discussed below: Password-Based Key Derivation Function 2 (PBKDF2) and Hashed Message Authentication Code-based key derivation function (HKDF). This process is typically known as key stretching [5], [6], [7].

PBKDF2

The PBKDF2 key derivation function has five input parameters:

$$\mathbf{DK_1 = PBKDF2\ (PRF,\ PSK,\ SDEV_{NONCE}||GW\_INK_{NONCE},\ i,\ dkLen)}$$

where:

- PRF: pseudorandom function of two parameters with output length hLen (e.g. a keyed HMAC-SHA-256)
- PSK: the master key (or premaster key) from which a derived key is generated
- $SDEV_{NONCE}||GW\_INK_{NONCE}$: cryptographic salt
- i: number of iterations desired
- dkLen: the desired length of the derived key (it depends by the chosen cipher suite)
- DK_1 is the derived key

If AEAD algorithm is not used, derive another key for sign the data is recommended.

$$\mathbf{DK_2 = PBKDF2\ (PRF,\ } \textit{firstpart}\mathbf{(PSK/2)||SDEV_{NONCE}||GW\_INK_{NONCE},\ SDEV_{NONCE}||GW\_INK_{NONCE},\ i,\ dkLen)}$$

where DK_2 is the derived key used to sign the Message Authentication Code.

HKDF

The HKDF key derivation function has four parameters:

$$\mathbf{DK_1 = HKDF\ (alg,\ PSK,\ SDEV_{NONCE}||GW\_INK_{NONCE},\ info,\ dkLen)}$$

where:

- alg: hash function algorithm
- PSK: the master key (or premaster key) from which a derived key is generated
- (optional) $SDEV_{NONCE}||GW\_INK_{NONCE}$: cryptographic salt
- (optional) info: optional context and application specific information
- dkLen: the desired length of the derived key
- DK_1 is the derived key

If AEAD algorithm is not used, derive another key for sign the data is recommended.

DK2 = HKDF (alg, firstpart(PSK/2), SDEVNONCE||GW_INKNONCE, info, dkLen)

where DK_2 is the derived key used to sign the Message Authentication Code (MAC).

4.6 State Management Mechanism

A session identifier (session ID) can be adopted to keep a session between an SDEV and Gateway/Innkeeper for a limited time. This approach is useful in order to avoid key negotiation at every interaction. A method used to keep the session between the two entities, consists in using cookies. When cookies are adopted, the Gateway/Innkeeper requests to store a cookie in the SDEV by setting the Set-Cookie HTTP Response Header while sending the GW_INK Hello message.

This cookie contains a unique session ID (SESSIONID) and the expiration time (expires field). The session ID can be generated by using the $SDEV_{MAC}$ together to a random nonce. The expiration time indicates when the cookie should no longer be sent to the Gateway/Innkeeper and therefore may be deleted by the SDEV [8].

Example of Response Header

Set-Cookie: SESSIONID=$SDEV_{MAC}$+random_nonce();expires=Tue, 17 Oct 2017 23:45:00 GMT

The cookie is then sent back to the Gateway/Innkeeper by the SDEV using the Cookie HTTP Request Header on each request and thus the Gateway/Innkeeper is informed on each request the session ID currently assigned to the client.

Example of Request Header

Cookie: SESSIONID=$SDEV_{MAC}$+random_nonce()

In Appendix, paragraph 0, security negotiation protocol messages examples are included.

4.7 Security services between SSP middleware and IoT platforms

In order to protect the data communication between middleware and IoT Platforms, TLS (Transport Layer Security) protocol is the best choice to achieve message confidentiality, integrity and availability (CIA).

5 API component implementation

We now detail the design and implementation of the key features for the SymbIoTe Smart Space Middleware (S3M). Some of the components defined here are still under implementation or will be included in the next software releases (e.g. SSP Search in the Innkeeper). The software modules are publicly available on GitHub while the download links are in the appendix.

The objectives pursued in the S3M features are the automatic registration of resources, the IoT platform/device interoperability at the SSP level (platform transparency), and the security components necessary for such operations. Since the SSP must be able to function even when the cloud components are not reachable (recall for example of the Smart Yachting Use Case), the S3M will handle locally the mechanisms of resources registration and access, replicating the paradigm adopted in the Cloud domain. Therefore, the S3M is influenced by the analysis and implementation of the Core and the Cloud components and the complex interactions of software modules both inside and outside the SSP. Figure 8 shows the general architecture of the SSP with all its components.

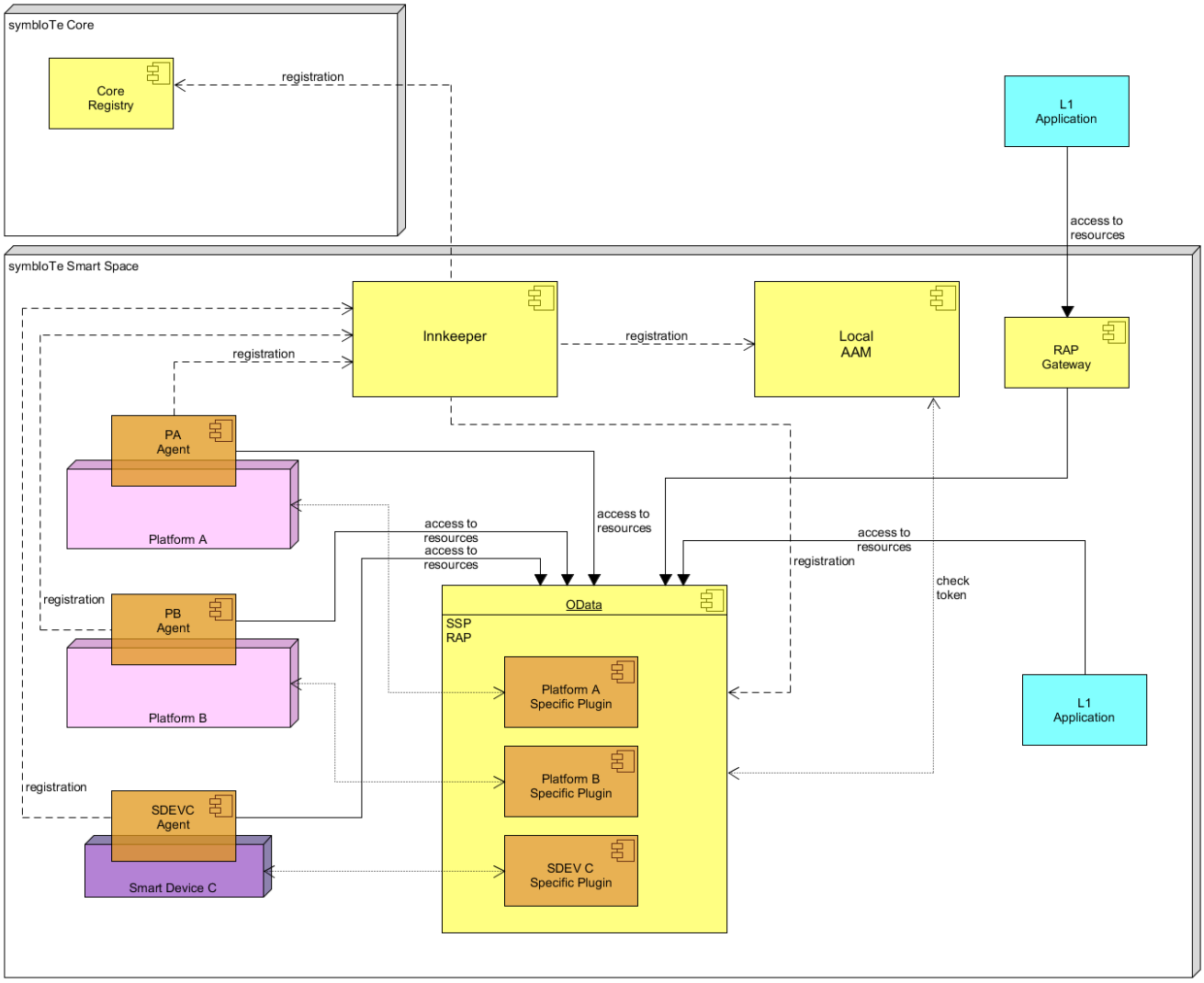


Figure 8: General architecture of the symbIoTe SSP.

5.1 SSP Registration and access of Resources

Platforms' resources can be registered into the SSP after their Platform Agents are registered in the Local AAM. While platforms ensure their trustworthiness using certificates, SDEVs might have not enough computational power to host certificates, so it have been designed the lightweight security protocol (section 4.4.2).

Depending on the entity (Platform or SDEV) that is going to be registered in the SSP, the process follows slightly different paths:

1. Platform:
    a. Platform Owner (PO)
        i. registers itself in the Local AAM
        ii. is granted a username and password for its Platform Agent from the SSP owner
    b. Platform Owner configures its Platform Agent with the newly acquired credentials
2. L3 SDEV:
    a. SDEV Owner registers the SDEV to SSP through the Innkeeper (getting a fresh Id) using the lightweight security protocol
3. L4 SDEV:
    a. SDEV Owner registers the SDEV to SSP through the Innkeeper (getting a fresh Id) using the lightweight security protocol
    b. Innkeeper registers the SDEV in the Core Registry (first seen SSP time only)
    c. Upon roaming, SDEV Owner registers the SDEV to the new SSP (using it's provided unique Id)
    d. Innkeeper verifies the id in the Core and updates SDEV's URL in the Core

Afterwards, Platforms and SDEVs are able to register their resources in the SSP:

i. Platform/SDEV Agent registers the resource to the Innkeeper, defining the resource access policy
ii. (if needed) the Innkeeper forwards the registration message to the Core
iii. The Innkeeper stores the resource metadata locally

An L1 application needs to get the proper access rights from the SSP, to get local access to resources. The L1 application will then access local resources through the SSP RAP, which needs to perform similar mapping actions than the ones required in the L1/Cloud case.

OData format is used for resource queries, then requests are mapped into plain JSON messages and sent towards each platform specific plugin. This process does not involve semantics, because during this sort of translation (from OData to JSON), only the format (so the syntax) is changing. It makes perfect sense to adopt the same syntax (i.e. OData) and semantics also in the SSP. Hence, SSP Resource Access Proxy (RAP) and Cloud

RAP can and should share the same source code, with minor changes for what concerns RAP Plugin adaptors.

Depending whether the Application is inside or outside the SSP, the request can follow a different path:

1. Application inside the SSP / SSP Platform:
    a. The user logins to the Local AAM and receives a token with its attributes
    b. The user delivers the token to the SSP RAP
    c. SSP RAP verifies locally that the token is valid
    d. SSP RAP performs a check revocation procedure towards AAM
    e. The resource access is provided if the set of attributes matches the access policy
    f. Request is forwarded to Platform/SDEV RAP plugin
2. Application outside the SSP:
    a. The user logins to the Local AAM and receives a token with its attributes
    b. The user delivers the token to the RAP SSP Gateway
    c. SSP gateway redirect request towards SSP RAP
    d. The RAP verifies locally that the token is valid
    e. The RAP performs a check revocation procedure towards AAM
    f. The resource is provided if the set of attributes matches the access policy
    g. Request is forwarded to Platform/SDEV RAP plugin

In addition to registration, the Platform Agent acts as a client for the other resources inside the SSP.

5.2 Meta Information Model extension

Figure 9 shows a part of the Meta Information Model (MIM) and how it is extended to support Smart Spaces and Smart Devices. It is used to store information about registered Systems in the symbIoTe Core components. Details about the MIM can be found in Deliverable D2.4 [9].

To support different kind if systems we introduced the *System* class with various four different subclasses: *SmartSpace*, *SmartDevice*, *Platform* and *Enabler*. A *SmartDevice* has the properties *available* and *mac* (to keep track of its identity) and can, as well as a *Platform*, be connected to a *SmartSpace*.

The complete model can be found on the symbIoTe website[1] and on Github[2].

[1] https://www.symbiote-h2020.eu/ontology/meta
[2] https://github.com/symbiote-h2020/Ontologies

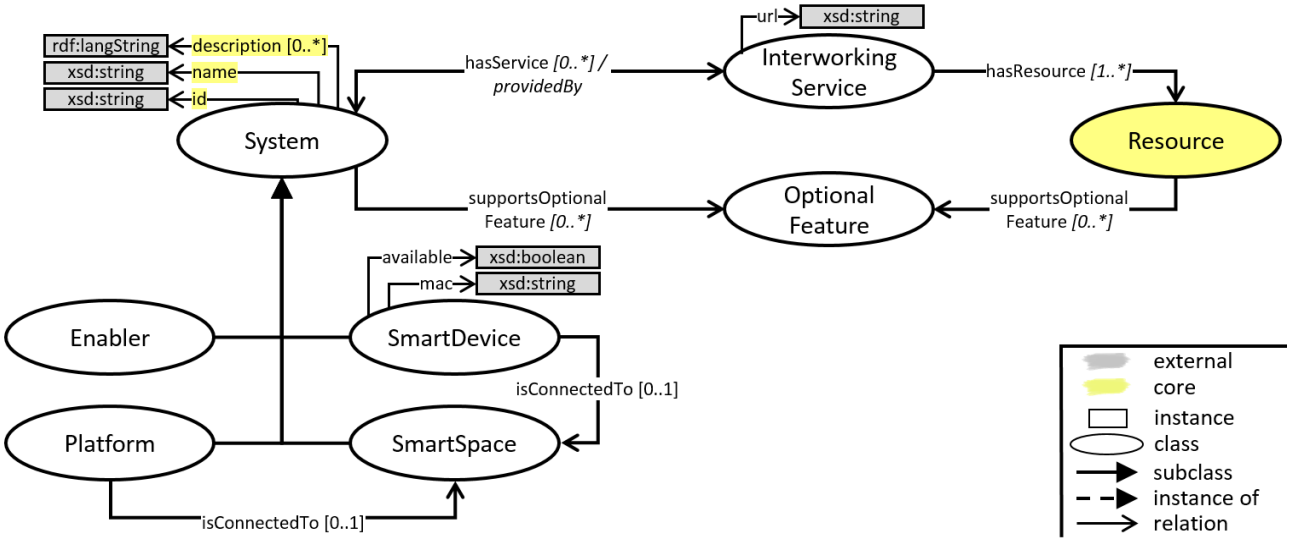


Figure 9: Partial Meta Information Model (MIM) with extensions for Smart Spaces and Smart Device.

5.3 SymbIoTe middleware

5.3.1 Component: Innkeeper

5.3.1.1 Innkeeper component description

This component is in charge of communicating with L3/L4-compliant applications and devices in order to enable their registration and interaction with the SSP. It mainly fills the role of a local registry in the SSP, providing a list of applications and devices currently registered in the SSP. Furthermore, it provides information about the SSP devices such as the current status and location.

Innkeeper is also required for interaction with upper layers. Specifically, it updates information (e.g. location) of L4 roaming devices in upper layers by communicating with the Registration Handler of the IoT platform owing the roaming device. Then, Registration Handler should forward the updated information to the symbIoTe core.

5.3.1.2 Innkeeper component interface

Innkeeper exposes REST interfaces to enable communication with L3/L4-compliant applications and devices. In the table below, there is a summary of Innkeeper's interfaces.

Table 1: Summary of Innkeeper's interfaces

#	Interface	Name	Message Type	From	Msg Consumers	Address/Queue	Payload	Description

1	Platform / L3/L4 SDEV Registration	/innkeeper/registry	REST	Sym-Agent	Innkeeper	POST	{
							String id,
							String name,
							String description,
							String url,
							String informationModel,
							}
								The L3/L4 SDEVs register in the SSP through the innkeeper
								NOTE: if the registration request is provided by an L3 SDEV id is a “fresh Id”. If the registration is provided by a L4 SDEV Id contains an unique Id.

2	Platform / L3/L4 SDEV Unregistration	/innkeeper/unregistry	REST	Sym-Agent	Innkeeper	POST	{
							String id,
							}	

3	Update Platform Credentials	/innkeeper/credential_update	REST	Innkeeper	Local AAM	POST	{
							String id,
							String name,
							String description,
							String url,
							String informationModel,
							}	

4	Update SDEV URL	/innkeeper/update_sdev_url	REST	Innkeeper	Core	POST	{
							String id,
							String name,
							String description,
							String url,
							String informationModel,
							}	

5	Resource Registration	/innkeeper/join	REST	Sym-Agent	Innkeeper	POST	{
							String id,
							String mac,
							String semanticDescription[3]
							}	

6	keep_alive_ssp	/innkeeper/keep_alive/	REST	Sym-agent	Innkeeper	POST	{
							String id
							}
								Send a keep-alive message.

7	Forward resource registration to core	/innkeeper/join_forward	REST	Innkeeper	Core	POST	{
							String id,
							String mac,
							String semanticDescription[3]
							}	

[3] Please note that this is a serialized and not formatted string representing the JSON of the semantic description of the resources.

8	*Resource query (RPC)*	*resource.searchRequested*	*event*	*Innkeeper*	*Innkeeper Search*	*Exchange: symbiote.resource, routing key equal to "Name" column*	*Query parameters*	*Event requesting query results for supplied query parameters*
9	*RPC response*	*resource.searchPerformed*	*event*	*Innkeeper Search*	*Innkeeper*	*Temporary queue used only for RPC response*	*Search Response*	*Query results, containing resources matching specified parameters*

Note: SSP search modules (7-8) are not implemented yet.

5.3.2 Component: SSP RAP

5.3.2.1 SSP RAP component description

This component enables symbIoTe-compliant access to resources within IoT platforms located in a SSP or to SDEVs. It must receive incoming access requests from applications/platform agents using a symbIoTe-compliant communication protocol and data format. A request must contain a unique identifier assigned to a resource. It must check that security policies included in the request are valid and that access to a particular resource can be granted.

The data generated by IoT platform / SDEV must be returned in a format which complies with the symbIoTe information model.

5.3.2.2 SSP RAP component interface

The SSP RAP exposes both REST and OData interfaces for direct resources' access. It also communicates with the Innkeeper component via function calls (both resides in the Middleware application). An additional interface is used for push mechanism, where notifications are linked via WebSocket with the client application.

In the table below, it is possible to see a summary of SSP RAP's external interfaces.

Table 2: RAP External Interfaces

#	Interface	Name	Message Type	From	Msg Consumers	Address/ Queue	Payload	Description
4a	Resource access read	/rap/Sensor/{resourceId}	REST	Application / Agent	RAP	GET	None - replies with the value of the resource	Event reading the value of a resource
4b	Resource access read	/rap/Sensor({resourceId})/Observations?$top=1	OData	Application / Agent	RAP	GET	None - replies with the value of the resource	Event reading the value of a resource
5a	Resource access read history	/rap/Sensor/{resourceId}/history	REST	Application / Agent	RAP	GET	None - replies with the history values of the resource	Event reading the value of a resource
5b	Resource access read history	/rap/Sensor({resourceId})/Observations	OData	Application / Agent	RAP	GET	None - replies with the history values of the resource	Event reading the value of a resource
6a	Resource access	/rap/Service/{resourceId}	REST	Application / Agent	RAP	POST	{ [“param_name”:“param_value”, ..] }	Event sending the value to a service
6a	Resource access	/rap/Service({resourceId})	OData	Application / Agent	RAP	PUT	{ [{“param_name”:“param_value”}, ..] }	Event sending the value to a service
6b	Resource access write	/rap/Actuator/{resourceId}	REST	Application / Agent	RAP	POST	{ [{ “capabiliy_name”: [{“param_name”:”param_value”}, ..] }, ..] }	Event writing the value of a resource
6b	Resource access write	/rap/Actuator{resourceId}	OData	Application / Agent	RAP	PUT	{ [{ “capabiliy_name”: [{“param_name”:”param_value”}, ..] }, ..] }	Event writing the value of a resource
7	Resource notifications	/notification	WebSocket	Application / Agent	RAP	client / server	the value of the resource	Event reading the value of a resource

5.3.3 Component: RAP Gateway

This component act as a gateway, allowing access to local Smart Devices for clients that are outside the SSP. Basically, it establishes a tunnel connection to the Smart Space local network, providing an access point for resources to be addressable from outside the SSP. The work related to RAP Gateway was conceptual; no implementation exists at the moment.

5.3.4 Component: Local AAM

The SSP by design interacts with L1 clients. Therefore, within the SSP will be deployed a slightly modified Platform AAM's logic. This will allow to satisfy L1 compliance requirements - common ecosystem security scheme - trust chaining for mutual authentication in the whole ecosystem as well as SSP actors attributes / resources management features described in sections 4.2 and 4.3 of this document. The identified modification is a new feature allowing registration of platform agents.

This component will however not be extended with the SSP-specific security schemes described in sections 4.4 onward. The reference Platform AAM is described in sections 5.2.2 and 5.4 of deliverable D1.4[4] with updates in D3.2 Annex B.

5.3.5 Component: symbIoTe Agent

The symbIoTe agent is the component core of the communication between the SSP and the SDEV. It acts as a translator between the custom language talked by the sensor and the symbIoTe ecosystem. Every device that wants to be symbIoTe compliant at SSP level needs to implement this module on their platform side.

To reach this scope, the agent uses a well-defined communication protocol to talk with the Innkeeper and RAP.

The following table shows the interfaces used by the agent to communicate with the SSP middleware, please note that this is the decrypted content. Such information is indeed encrypted using the keys shared with the lightweight security protocol defined in section 4.5:

[4] https://doi.org/10.5281/zenodo.830156

Table 3: symbIoTe Agent interfaces

#	Interface	Name ssp.symbiote.org	Message Type	From	Msg Consumers	Address /Queue	Payload	Description
1	join_ssp	/innkeeper/join/	REST	sym-agent	Innkeeper	POST	{ String id, String mac, String semanticDescription[5] }	Event requesting join of a new SDEV
2	keep_alive_ssp	/innkeeper/keep_alive/	REST	sym-agent	Innkeeper	POST	{ String id }	Send a keep-alive message.
#	Interface	Name <agent IP>	Message Type	From	Msg Consumers	Address /Queue	Payload	Description
3	actuate_resource_agent	/ActuateResourceAgent/	REST	SSP RAP	sym-agent	POST	{ String id, action { <observesProperty1>: "value1", <observesProperty2>: "value2" } }	Actuate a specific service
4	request_resource_agent	/RequestResourceAgent/	REST	SSP RAP	sym-agent	POST	{ String id }	Request a specific resource from SSP

[5] Please note that this is a serialized and not formatted string representing the JSON of the semantic description of the resources.

5.3.6 Component: IoT platform agent

IoT platform agent has similar interfaces as the SDEV Agent, with the additional feature of resources registration – i.e. integration and compliancy with L1/L2 security by using the ComponentSecurityHandler. It allows different users within a platform to register and manage the resources exposed by this platform in the SSP. All the interfaces belonging to IoT platform agent are listed in the table within the previous subsection.

IoT platform agent uses a different communication protocol than the one used by Smart Device and presented in Section 4.4. However, this is out of scope of this section, where only high level interfaces are described.

6 Wireless technologies

Smart Devices can be equipped with one or more widespread wireless technologies, such as WiFi, ZigBee and LoRa, operating in the unlicensed ISM bands, or 3G/4G cellular technologies. These technologies are often complementary in terms of coverage, reliability and costs. Therefore, for a given device with multiple available technologies, it is important to select the most suitable technology and access point to support a specific service in a given environment. The symbIoTe environment may face this problem, identify the coexisting networks and allow fast reconfigurations and integration of multiple network interfaces.

Moreover, apart from the single-device decisions, the proliferation of platforms based on WiFi and ZigBee technologies working on unlicensed bands, as well as the recent introduction of LTE in Unlicensed spectrum (LTE-U) is creating coordination problems among independent platforms, which may be faced by the symbIoTe middleware. Since multiple infrastructures can exist within the Smart Space, the symbIoTe middleware may enable bartering mechanisms among different IoT platforms to take care of the roaming smart devices.

Summarizing, although not tangible as hardware and software resources deployed in the Smart Space, the wireless medium represents a physical resource in the Smart Space, whose access needs to be orchestrated among coexisting IoT platforms and Smart Devices by means of the symbIoTe client and middleware modules. A dedicated local component may be considered for managing the access to this resource.

Note that wireless connectivity should be distinguished from the logical connectivity built by the SymbIoTe SSP Middleware as they represent two orthogonal problems. They have indeed been addressed in two different Tasks (T4.2 and T4.3) and are both mentioned this deliverable because the wireless medium is critical for the SSP architecture.

In particular, Task 4.2 has focused WiFi and LoRa networks which are relevant technologies for SymbIoTe. Regarding WiFi, we study inter-technology interference and show that such interference can be recognized by commodity WiFi devices by monitoring the statistics of receiver errors. Exploiting recent Artificial Intelligence (AI) advances, we develop an Artificial Neural Network (ANN) able to recognize the source of interference, reaching an average accuracy of almost 99% in recognizing ZigBee, Microwave and LTE-U interference.

We then show how inter-technology interference can be exploited to set-up a low-rate bi-directional communication channel between heterogeneous technologies, such as WiFi and ZigBee. Experimental results show the feasibility of the inter-technology communication channel. This unconventional communication channel can be very useful for coordinating channel access between WiFi and ZigBee networks (for example, we designed and implemented a cross-technology TDMA scheme, alternating WiFi and ZigBee transmissions). It can also be used for other direct link applications, such as reading measurements from ZigBee sensors, or configuring ZigBee actuators (e.g. an on/off power switch) by just using common smartphones or laptops which are only equipped with WiFi interfaces[6].

Regarding LoRa, we studied the impact of imperfect-orthogonality in LoRa spreading factors (SFs) in simulation and real-world experiments. First, we show that collisions

[6] Clearly, security must be assured by the Local AAM for such direct-link communications.

between packets of different SFs can cause packet loss if the interference power received is strong enough. Then, we measure the impact of inter-SF collisions, showing that non-orthogonality of the SFs can deteriorate significantly the performance especially of higher SFs (10 to 12) and that fading has virtually no impact when multiple gateways are available in space diversity. Finally, we developed a LoRa cell traffic generator, able to emulate the behavior of thousands of low-rate sensor nodes deployed in the same cell.

6.1 WiFi

Nowadays, we are witnessing an impressive success of IEEE 802.11 technology, better known as WiFi, for supporting the growing demand of wireless broadband connectivity. Public WiFi networks are deployed worldwide, with more than 50% of the total mobile traffic carried by WiFi [10]. The availability of WiFi networks is often considered as a commodity service driving immense economic value, and the unlicensed spectrum is becoming one of society's most valuable resources. Although WiFi is a dominant communication technology in this spectrum, many other low range technologies coexist in unlicensed ISM bands for supporting several vertical applications, such as building automation, smart metering systems, health care monitoring, surveillance systems, game remote controllers and so on. Moreover, cellular technologies are trying to extend their operation to ISM bands for increasing their capacity. Two different solutions have been envisioned by 3GPP in ISM bands, referred to as Licensed Assisted Access (LAA) [11] and LTE-Unlicensed (LTE-U) [12], which work respectively, with and without the listen-before-talk mechanism.

Although in WiFi carrier sense and adaptive modulation mechanisms have been included, it has been shown that serious performance impairments can arise in presence of exogenous interfering signals due to different technologies. For example, in [13] it is shown that the capacity of a good WiFi link can be reduced to zero in presence of analog phones, video cameras, or sensors based on IEEE 802.15.4 technology [14], [15], while other devices such as a Xbox controller and a microwave oven can half the throughput. The effect of sensors' interference on the WiFi link is impressive if we consider that the 802.11 and 802.15.4 technologies (hereafter referred as ZigBee and WiFi) are heterogeneous in terms of bandwidth (2 versus 20 MHz) and transmission power (e.g. 0 dBm for ZigBee and 20 dBm for WiFi). The possible reasons are that some WiFi implementations are unable to detect non-WiFi signals or introduce latencies [16] or because of the different timings to perform CSMA/CA [17], [18].

Regarding the interference with cellular technologies, several research studies are trying to characterize the impact on LTE transmissions on WiFi performance. Preliminary empirical and simulation results [19] show that WiFi performance can be critically affected even when LTE links operate at the minimum bandwidth of 1.4 MHz. In [20] it is demonstrated that even when utilizing the listen-before-talk principle, LAA-LTE heavily impacts WiFi performance, and that WiFi with MIMO performs worse than WiFi without MIMO when LTE interference is strong. Additionally, increasing distance between LTE and WiFi links does not necessarily decrease the impact of interference in indoor environments.

In this scenario, it is important to identify the presence of such coexisting technologies to allow possible countermeasures (e.g. finding a different channel) or activate some co-ordination mechanisms (e.g. setup a cross-technology TDMA scheme [21]). For this classification, the typical approach in the literature is to analyze the RSSI samples in the frequency and time domain [13], [22], or to perform a cyclostationary signal analysis and

blind signal detection [23] and other spectrum sensing techniques [24]. Although these approaches are very effective, they usually require to monitor the interfering signals for some seconds or use specialized hardware. Instead, in this deliverable we propose to simply monitor the reception errors of commodity WiFi cards, and then apply an artificial neural network in order to identify and characterize cross-technology interference. Following the initial approach of [25], we analyze the error domain, i.e. the error burst caused by the interfering signal. However, in this deliverable we exploit a more powerful classification tool, the Artificial Neural Networks, and we extend the analysis to the emerging LTE in unlicensed spectrum. Experimental results show that our solution provides excellent results, with an average of almost 99% accuracy.

After the correct identification of the coexisting networks, in order to harmonize channel access it is often necessary to introduce some coordination mechanisms by using multi-technology gateways [26], or increasing the robustness of transmission with error correction codes or multiple antennas [27]. Indirect coordination is also possible, for example by transmitting busy tones in an adjacent ZigBee channel for preventing WiFi nodes to interfere with the main ZigBee channel [28].

An alternative communication mean is proposed in [29], where special pulses, detectable by both the technologies, code some simple coordination messages. To this purpose, an inter-technology communication channel can be useful to directly notify network parameters and speed up the set-up of any coordination mechanism. More in general, this channel can be used for exchanging simple communication messages between different technologies, bridging two (or more) coexisting networks. Exploiting the error-based identification technique, in this deliverable we build an unconventional communication channel between interfering technologies by artificially provoking such reception errors. We specifically deal with 802.15.4 and 802.11 technologies (commonly referred as ZigBee and WiFi), and we create special interference patterns to be used as in-band messages for inter-technology WiFi/ZigBee communications.

This provides network designers the unique opportunity to explicitly send information to a different network, without the need of an external gateway or other indirect form of communication. The proposed solution is completely backward compatible towards legacy stations and can be used to send messages in parallel to both networks, e.g. to coordinate channel access between the interfering technologies, improving performances in highly congested scenarios. Through extensive testbed experiments, we demonstrate the feasibility of our approach. Finally, we show that inter-technology interference can be effectively exploited for improving coexistence, with an application example where WiFi and ZigBee share the channel in a TDMA-like fashion.

6.1.1 Error-based Interference Detection

We now show that inter-technology interference can be recognized by commodity (off-the-shelf) WiFi devices by monitoring the statistics of receiver errors. Indeed, while for WiFi standard frames the error probability varies during the frame reception in different frame fields (PHY, MAC headers, payloads) protected with heterogeneous coding, errors may appear randomly at any point during the time the demodulator is trying to receive an exogenous interfering signal. We thus detect and identify cross-technology interference on off-the-shelf WiFi cards by monitoring the sequence of receiver errors (bad PLCP, bad FCS, invalid headers, etc.) and develop an ANN to recognize the source of interference. The result is quite impressive, reaching an average accuracy of almost 99% in recognizing ZigBee, Microwave and LTE-U interference.

Although in this deliverable we focus on three interference sources, namely ZigBee, LTE and microwave ovens, our solution does not depend on the type of technology, but only requires a training phase based on the events generated in presence of a controlled source of interference. We then employ an ANN to identify the interference technology from the occurrence of the generated error events. The idea is that the proposed approach could be easily extended to any other type of interference.

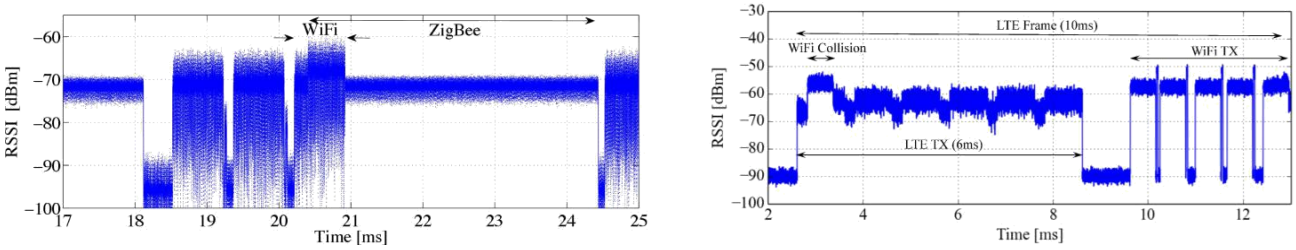
6.1.1.1 Background

In this section we briefly recall some key aspects of the MAC/PHY layers in WiFi, ZigBee and LTE that affect the power of cross-technology interference and the typical timings of transmissions and channel idle intervals. We also provide some background on ANNs.

Interference power. WiFi and LTE transmissions are typically performed at a maximum power of 15 or 20dBm, while ZigBee transmissions can span in the range [−25, 0]dBm. LTE transmission power is modulated because of power control mechanisms, which are usually not implemented in WiFi and ZigBee. Additionally, each WiFi channel is 20 MHz wide and is spaced of 5 MHz from the adjacent ones. ZigBee channels have only 2 MHz of bandwidth with 3 MHz of inter-channel gap bands (i.e. the center frequencies maintain the spacing of 5 MHz from the adjacent channels). It follows that four ZigBee channels are entirely included in a WiFi channel. LTE center frequencies in ISM bands coincide with WiFi ones, with a typical bandwidth of 5 MHz (but bandwidths as small as 1.4 MHz are possible).

Transmission times. Since the three technologies have been defined for different applications, the frame size, the data rates and the channel access units considered by the standards are quite different. For WiFi and ZigBee, channel access is performed on a per-packet basis, i.e. transmission times correspond to the time required for completing the transmission of a packet (or an aggregation/fragmentation of packets). ZigBee packets are small, with a maximum payload of only 127 bytes. Bytes are organized into 4-bit symbols that are mapped into 16 pseudo-random sequences of 32-chip transmitted at 2 Mchip/s (i.e. 250 Kbps), which correspond to a frame transmission interval of about 4 ms for the maximum frame size. WiFi frames are much longer, with a maximum frame size of 2358 bytes and multiple OFDM modulations and coding schemes available (from 6 Mbps up to 54 Mbps, which lead respectively to a maximum transmission time of about 3.2 ms and 0.37 ms).

For LTE, the channel access is performed on the basis of resource block allocations, which are organized into sub-intervals lasting a fixed time of 1 ms within a frame of 10 ms. Packet transmissions are achieved by scheduling a given set of resource blocks in one or multiple consecutive frames. Although the total number of resource blocks used for each packet depends on the employed data rate and multiple rates are available (up to 25.2 Mbps for 5 MHz of bandwidth with 300 sub-carriers, 64-QAM modulation, and a symbol time of 71.4 μs), the channel occupancy time in each channel access is fixed according to the LTE frame structure.



(a) (b)

Figure 10: Interference between technologies:
temporal trace (RSSI samples) of WiFi-ZigBee (a) and WiFi-LTE collisions (b).

***Intervals between transmissions*.** Different channel access schemes are employed in WiFi, ZigBee and LTE for unlicensed bands. WiFi and ZigBee are mostly based on random access although channel sensing is performed with different granularity: ZigBee spends 128 µs for detecting the channel activity and 192 µs to switch from reception to transmission mode. Since WiFi slots are much shorter (9 µs), if a WiFi transmission is originated during this switching time, it cannot be detected by the ZigBee node. Figure 10-a shows a channel occupancy trace acquired by means of a USRP node in a network in which a WiFi node coexist with a ZigBee one.

In the figure we clearly observe that each transmitter is characterized by a specific RSSI value and frame transmission time: WiFi frames occupy the channel for less than 1 ms with a RSSI value of -65 dBm, while ZigBee frames last 4 ms with a RSSI value of -72 dBm. The figure also shows that a ZigBee transmission can overlap with WiFi, in case a WiFi frame is transmitted during the time spend by ZigBee for switching from sensing to transmission mode. LTE transmissions in licensed bands are organized into frames of 10 ms that start at regular time intervals.

For operating in unlicensed bands, two different adaptations have been envisioned: employing duty cycles for periodically suspending frame transmissions, while keeping the synchronization of time instants at which frame transmissions can start (LTE-U); employing listen-before-talk before transmitting each frame (LTE-LAA). In this second case, when the medium is sensed as busy, the deferral time is given by a fixed time of 10 ms for maintaining the synchronization of frame starting times (with the so called FBE mechanism) or it is given by a random slotted deferral time compensated by a varying channel occupancy time (with the so called LBE mechanism). In our work, we emulate both the LTE-U and LTE-LAA approach, by assuming that LTE frame transmissions can start only at regular time intervals. Figure 10-b gives an example of the interaction between an LTE-U transmission with 6 active and 4 silent subframes (i.e. 6 ms on and 4 ms off) and a WiFi station which tries to access the same channel: the figure shows that WiFi packets can collide with LTE and that part of the channel time is wasted due to the consequent backoff.

Artificial Neural Networks. ANNs are a class of powerful machine learning ("**Artificial Intelligence**") tools that can be used to solve classification and regression problems. They can be distinguished in two types of architectures, depending on the types of connection between neurons: in the feedback architectures, the presence of connections between neurons of the same layer or between neurons of the previous layer realizes a feedback connection.

In the feedforward architectures, the connections between the neurons do not allow feedback between layers, and the signal is transmitted only to the neurons belonging to the next layer. In this article we used a neural network model called Multi-Layer Perceptron (MLP), a widely used feedforward ANN, formed by one input layer, one or more inner layers called “hidden”, and a layer of output neurons. The training of the network is usually done with an algorithm called back propagation, which is basically divided into two phases. In the first phase, called forwarding, controlled inputs are applied to the network, causing the activation of the neurons of the input layer. The signal propagates to the next layers, finally reaching the output neurons. The error between the desired output and the obtained result for each neuron is then computed.

In the second phase, called backwarding, the error value is propagated backward and the weights of each link accordingly modified with an optimization method, which aims to minimize the output error with respect to all the network weights. Finally, the network “model selection” is achieved by choosing between a set of hyper-parameters. The hyper-parameters define the structure of the network, such as the number of hidden layers and the type of activation function, and the configuration of the training algorithm, such as the regularization factor and the learning rate. Comparing the performance scores of all possible combinations of parameters, we finally select the model giving the best score.

6.1.2 Error analysis in WiFi receivers

6.1.2.1 Monitoring Receiver Errors

In [25], we have shown that WiFi cards receiving non-WiFi modulated signals generate error patterns significantly different, in terms of occurrence probability and time intervals between consecutive errors, from the ones generated by collisions with other WiFi transmissions. In presence of wide-band noise and exogenous interference signals, WiFi receivers demodulate a sequence of completely random bits and try to interpret these bits according to the format of WiFi frames. Being all the bits random, the probability of having a specific error heavily depends on the format of the expected frame.

Most commercial WiFi cards track the occurrence of different receiver events, such as the start of a synchronization trial, the detection of wrong PLCP, the end of a frame transmission, etc., by means of specific counters implemented in internal registers. As a reference WiFi receiver, we considered a WiFi card (namely, Broadcom bcm4318) for which the card internal registers are documented and an interface for reading the register values is available [30]. For producing a temporal trace of the receiver events, storing the ordered sequence of event type and occurrence time, we implemented a monitoring process devised to sample at regular intervals the receiver registers. Indeed, the event occurrence cannot be detected by the card host as an interrupt signal, but needs to be indirectly identified by comparing the state of the receiver registers in consecutive sampling times. We set a sampling interval equal to 250μs as a trade-off between detection delay and tracking complexity, while avoiding the overloading of the card to host interface. Because of the periodic sampling, multiple receiver events can occur in the same monitoring interval. Event samples are represented by a vector of eight components, whose value represents the counter of each different event type. We also sampled another card register, called busy time register, which does not track the occurrence of receiver events but rather the cumulative time during which the receiver remains active. The differences among consecutive values of the busy time register can be mapped into a logical idle/busy state of the channel as observed by the receiver.

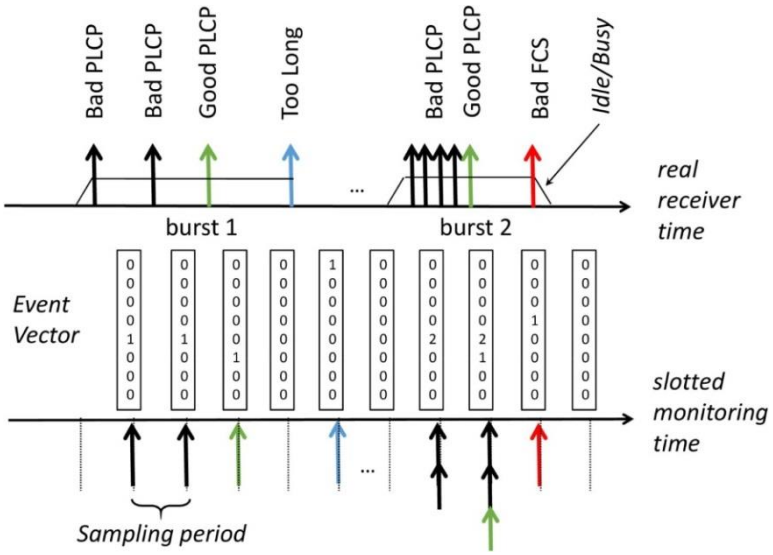
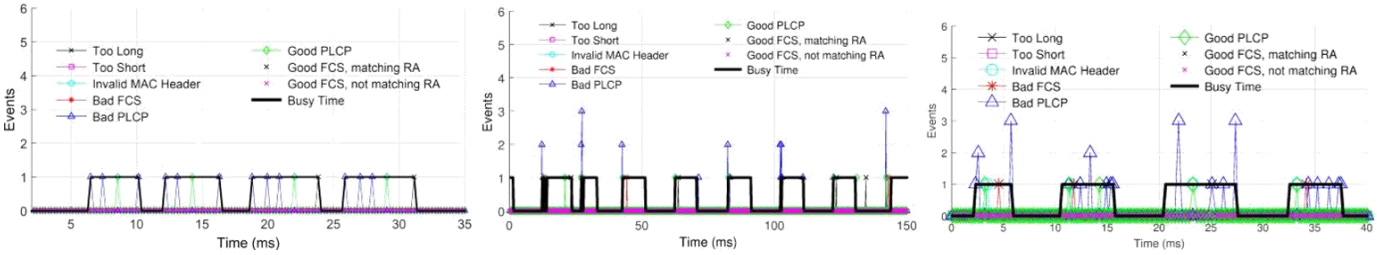


Figure 11: Mapping between a real trace of receiver events and the time-slotted vectors generated by the monitoring process.

Figure 11 shows the operation of our monitoring process: a real trace of receiver errors is mapped into a time series of event vectors, in which we can easily recognize consecutive error bursts due to the same interfering transmission. Error bursts can be originated for many different reasons: for example, a checksum failure can follow the detection of a good PLCP, or multiple (failed or not) synchronization trials are performed after a bad PLCP event. The total number of receiver events in a burst depends on the duration of the interfering transmission and on the receiver implementation, i.e. on the reset time required by the demodulator for performing consecutive synchronization trials. Each burst can be delimited by observing the time interval elapsed from the previous and next events, and/or by considering the channel transitions from idle to busy and from busy to idle as delimitation times.

6.1.2.2 Temporal Analysis

Testbed. For our experiments, we set up a testbed at the University of Palermo and placed a monitoring WiFi node together with heterogeneous interfering sources. Four different interfering sources have been considered: a ZigBee transmitter, a LTE transmitter, a WiFi transmitter and a microwave oven. All nodes have been set to a few meters distance between each other and the transmitting nodes are programmed to work on different interfering and non-interfering channels. ZigBee nodes used in our testbed are based on Microchip MRF24J40 transceiver. The ZigBee frames are transmitted at 250kbps with a length of 127 bytes. WiFi transmitter has been implemented by using the same Broadcom card used by the WiFi monitoring node, with a frame length of 1500 bytes transmitted at 24 or 36 Mbps. The LTE-U transmitter, instead, was implemented on a SDR (Software Defined Radio) platform based on USRP B-210 and the srsLTE framework [31]. We considered a downlink interfering stream with 5 MHz of bandwidth and 300 sub-carriers, centered on channel 11. Following the standard, the whole frame allocation time is 10ms composed of 10 sub-frames. The frame structure has been organized introducing silent intervals and a fixed sub-frame pattern, with mask [1,1,1,1,1,1,0,0,0,0] where 1 indicate transmission allowed and 0 transmission denied.



(a) (b) (c)

Figure 12: Bursts of receiver events corresponding to the reception of ZigBee, Microwave and LTE-U interference respectively.

Results. Figure 12 shows three traces of receiver events when receiving ZigBee, Microwave and LTE-U interference. Figure 12-a, for example, shows four ZigBee packets, with error events spaced approximately 1ms from each other. Figure 12-b shows the error events caused a Microwave oven. From the figure, it can be clearly recognized the periodical radiation pattern of the oven, with 10 ms of activity and 10 ms idle.

During radiation, channel is sensed as busy by the WiFi node, but error events are pretty different from the ones caused by ZigBee transmissions, since they are concentrated at the beginning and at the end of the radiation interval (rather than being continuously repeated). This can be due to the power-on and power-down ramp of the Microwave, being the demodulator unable to work when the radiation power is stable. Finally, Figure 12-c shows the receiver events in presence of LTE transmissions. Under this interference source, the WiFi receiver behavior resembles the ZigBee interference with the granularity of consecutive synchronization trials equal to regular intervals of 1 ms. However, occasionally, some events are closer to each other. We also observed, the occurrence of the first synchronization trial is not always synchronized with the activation of the channel busy register: for example, in the figure at time 20 ms the busy channel state switches to 1, while the first event vector with non-null components (namely, three Bad PLCP events) are revealed after 2 ms.

6.1.3 Interference detection

6.1.3.1 ANN Features

The experimental results presented in the previous section show that, although all non-WiFi interfering signals generate the same type of errors with similar statistics, their temporal analysis could be exploited for discriminating among different interfering sources. From the qualitative description of Figure 12, it clearly emerges that several features can be exploited for such discrimination, such as:

1. The number of events generated by the monitoring process during the same interfering burst, which depends on the interfering power, with an higher number of synchronization trials performed in case of LTE-U signals;
2. The length of the error burst, delimited by means of the correlation between the error vectors and the channel busy register, which depends on the transmission time of the interfering source;

3. The temporal gap between consecutive errors within the same burst, which might be symptom of power ramp effects (e.g. for the Microwave oven).

We propose to classify the interference sources through an MLP neural network with one hidden layer because this already provides good results and is computationally less expensive than networks with multiple hidden layers. As depicted in Figure 13, features in the input layer are organized in 10 neurons (8 representing the counter of 8 types of reception errors [25], one for the error burst length in µs, one representing the maximum distance between two consecutive events in the same burst in µs). For example, Table 4 shows the input features generated by the last LTE-U error burst shown in Figure 12-c. In the output layer, instead, we will have 4 neurons, each of them mapping the relevant interfering technology (WiFi, ZigBee, Microwave, LTE-U). The MLP network was implemented in Python using the scikit-learn machine learning library [32], trained using back propagation with a Cross-Entropy loss function. Details on how the ANN was designed and built, i.e. the number of neurons in the hidden layer, the activation function and the regularization factor can be found in appendix.

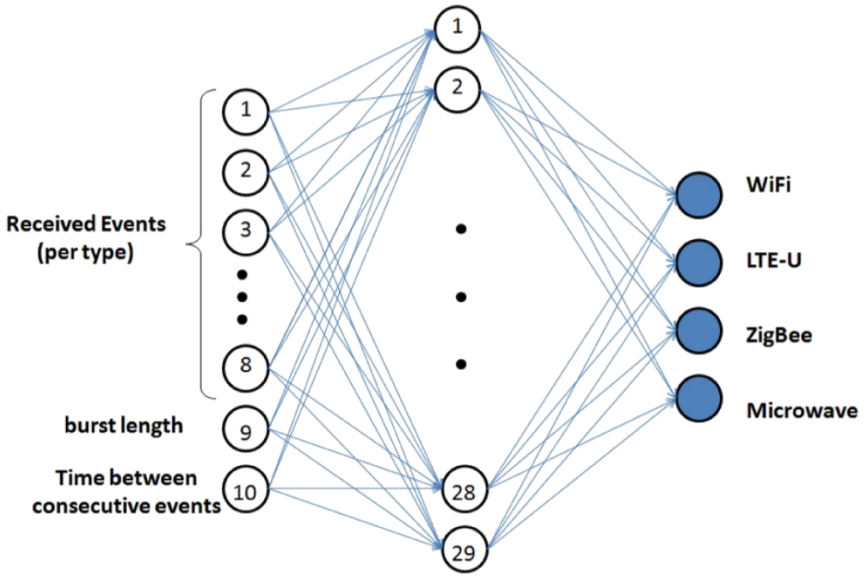


Figure 13: Structure of the MLP neural network used in our experiments.

Table 4: Input record relative to the last error burst of Figure 12-c.

Features	value
Too Long	0
Too Short	0
Invalid Mac Header	0
Bad FCS	1
Bad PLCP	4
Good PLCP	1
Good FCS, matching RA	0
Good FCS, not matching RA	0
Burst length	5370
Time between consecutive events	1100

6.1.3.2 Classification performance

We trained the network on the entire training set and evaluated the classification accuracy on the test set. To this purpose, we used a test set of 2020 burst samples (505 samples per class) representative of the four categories WiFi, ZigBee, Microwave and LTE-U. Table 5 shows the confusion matrix of the classifier, which obtains an average accuracy of 98.6%. The few errors are between ZigBee and LTE-U, because of the similarity of the error burst, as shown in Figure 12. Finally, to verify the robustness of the model, we evaluated the classifier on the entire dataset composed of 67653 elements. Table 6 shows that the classification performance is maintained even considering such a larger dataset, confirming the excellent results shown on the test set, reaching an average accuracy of almost 99%. Although in this deliverable the focus was to identify the interference caused by ZigBee, WiFi, microwave and LTE-U, the proposed approach could be easily extended to additional interfering technologies operating in the ISM band, e.g. Bluetooth or cordless phones.

Table 5: Confusion matrix for the test set.

	WiFi	**ZigBee**	**Microwave**	**LTE-U**
WiFi	**100.0**	0.0	0.0	0.0
ZigBee	0.0	**97.5**	0.4	2.1
Microwave	0.0	0.0	**100.0**	0.0
LTE-U	0.0	2.2	0.7	**97.0**

Table 6: Confusion matrix for the entire data set.

	WiFi	**ZigBee**	**Microwave**	**LTE-U**
WiFi	**100.0**	0.0	0.0	0.0
ZigBee	0.0	**98.2**	0.4	1.3
Microwave	0.0	0.0	**100.0**	0.0
LTE-U	0.0	3.4	0.6	**96.0**

6.2 Inter-Technology WiFi/ZigBee Communications

We now show how inter-technology interference can be exploited to set-up a low-rate bi-directional communication channel between heterogeneous technologies, which coexist in ISM bands. In particular, we focus on WiFi and ZigBee networks, whose high density deployments make coexistence a critical issue. We monitor the transmission duration of the interference and, after recognizing ZigBee interference from WiFi off-the-shelf receivers, we precisely measure the channel busy intervals to map time duration to communication symbols. A similar approach is used on the ZigBee receivers for making the communication channel bi-directional. Extensive experimental results show the feasibility of the inter-technology communication channel.

As a possible application, we designed and implemented a cross-technology TDMA scheme, alternating channel intervals to WiFi and ZigBee nodes. This unconventional communication channel can be very useful not only for coordinating channel access

between WiFi and ZigBee networks, but also for other direct link applications, such as reading measurements from ZigBee sensors, or configuring ZigBee actuators (e.g. an on/off power switch) by just using common smartphones or laptops which are only equipped with WiFi interfaces.

Note that in absence of direct communication mechanisms, WiFi and ZigBee can only communicate through “upper layers” via IP – i.e. both of them would need to communicate through symbIoTe L1 or L2, but this requires full compliance of both devices/gateways or platforms.

6.2.1 Related work

Based on the energy-detection capability of different cards, in [29] an inter-technology communication is pro-posed by pre-pending a customized preamble to each legacy packet, which contains multiple energy pulses whose gaps convey information for the coexisting technology. The approach is very promising, but cannot be implemented in commercial devices. Another interesting idea, presented in [32], uses specific RF-powered tags able to communicate with commodity WiFi devices. These tags can then use the existing WiFi infrastructure to access the Internet. While a particular hardware design is necessary for the implementation of the RF tags, in our work we use only standard hardware, bridging WiFi and ZigBee nodes.

Inter-technology communication channels between WiFi and ZigBee have been also considered for different applications. In [30], a unidirectional WiFi to ZigBee channel is used for notifying the presence of pending WiFi traffic to a multi-technology WiFi/ZigBee terminal, in which the WiFi interface is switched off for energy saving. The low-power ZigBee interface receives the messages sent by the Access Points, that are opportunistically encoded in terms of interference patterns, and wakes up the WiFi interface when required. Finally, in [33], energy profiles are used to communicate from WiFi to ZigBee with a unidirectional link and an alphabet set of 100 words. In this deliverable, we propose instead a multi-purpose bi-directional scheme for communication be-tween legacy WiFi and ZigBee devices, with an alphabet of 256 words (1 Byte) which makes the implementation much simpler.

6.2.2 Exploiting interference for unconventional WiFi/ZigBee communications

In this section we show how the capability of detecting interfering signals originated by a competing technology can be exploited for activating an inter-technology communication channel. Clearly, classification and communication are two independent tasks and can be executed in different ways. However, classification is necessary to implement the inter-technology communication, isolating the interfering traffic of interest. We exploit the classification technique explained previously to detect ZigBee frames in WiFi networks with the analysis of the error patterns caused by interference. As shown in Figure 14, the basic idea for the set-up of the inter-technology communication channel is to code a message into an interference signal with variable duration that can be recognized by the other technology. This can be achieved by defining a transmission scheme in which each inter-technology data symbol is mapped into a WiFi or ZigBee frame of different length. Receivers that can-not demodulate the frame (because their technology is different from the transmitter one) measure the busy time duration and decode the associated symbol.

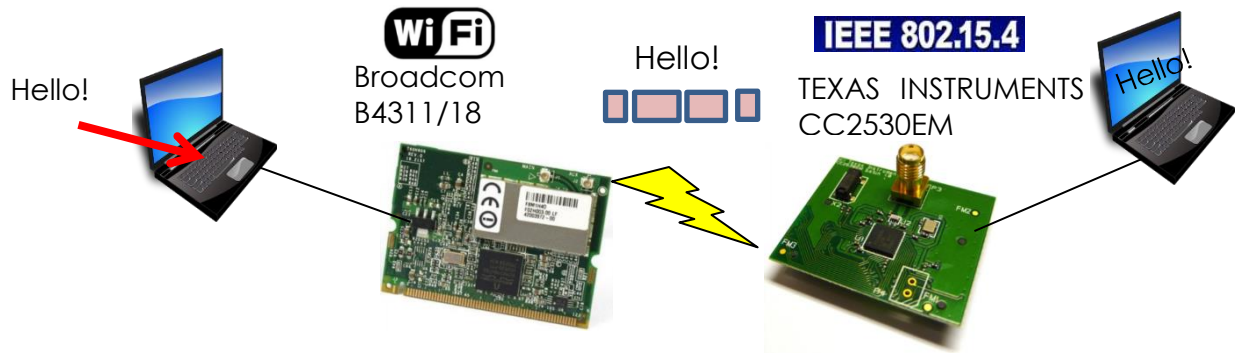


Figure 14: Inter-technology communication between WiFi and ZigBee cards.

Similar approaches have been investigated in [29] and [30]. However, in the first case the nodes are based on Software-Defined-Radio, rather than commercial devices, because it is required to transmit a customized preamble with energy pulses opportunistically spaced. In the second case, the inter-technology communication channel is unidirectional (from WiFi to ZigBee) because it has been designed for a specific power-saving application in which a feedback channel is not required. We argue that working with commercial devices and supporting a bi-directional channel can be very useful in many practical scenarios and in particular for enabling inter-technology coordination mechanisms.

We consider a scenario in which ZigBee and WiFi networks interfere in a symmetrical way, with performance impairments for both the networks. In case each network is able to infer about the presence of the coexisting heterogeneous technology, for example by monitoring the statistics of the receiver errors, we propose to exploit this capability for setting-up an inter-technology communication channel. The basic idea is modulating the duration of the inter-technology interference for coding simple messages to be exploited for improving coexistence. Since interference is due to the transmission of frames built according to a different standard, such a modulation can be achieved by transmitting frames with variable transmission times.

Note that intra-technology data can be carried in parallel by the same frames used for inter-technology communication. Fragmentation and/or zero padding allows the adaptation of the payload to the desired frame lengths. Therefore, the inter-technology channel is transparent to standard communications and backward-compatible with legacy stations already present in the network.

6.2.3 Frame Length Modulation

We designed two different modulation schemes for the ZigBee-to-WiFi and WiFi-to-ZigBee links, taking into account the different features of the two technologies. On one side, the accuracy on the measurement of the channel busy intervals depends on the carrier sense granularity of the receivers, which is much smaller for WiFi (less than 10µs) than ZigBee (more than 100µs). On the other side, the variability range of the frame duration depends on the maximum pay-load size (2304 byte for WiFi, 127 byte for ZigBee), while the frame duration granularity depends on the maximum data rate of the transmitters, which lead to a granularity lower than 1µs for WiFi transmissions (1 byte at 54 Mbps) and 32µs for ZigBee transmissions (1 byte at 250Kbps).

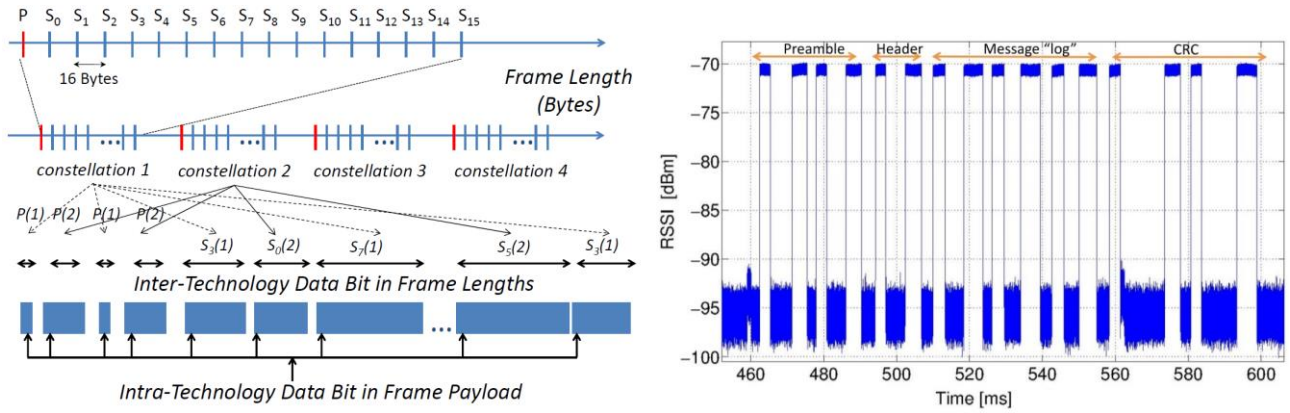
For the WiFi-to-ZigBee link, the main limit is due to the ZigBee receiver that is able to distinguish only interference intervals whose difference is higher than 32µs. This makes the system vulnerable to the environmental noise, i.e. to transmissions originated by other

coexisting networks (not involved in the inter-technology communication) that can be erroneously mapped into valid symbols. To cope with these features, we defined 4 constellations of 16 data symbols; in each constellation, symbols are equally spaced of 128 µs (i.e. 16 bytes of difference from one frame length to the next one, transmitted at 1 Mbps). Constellations are selected dynamically according to the environment traffic and node for avoiding collisions with intervals of equal duration due to other interferers. Decisions on symbol demodulation are based on the usual minimum distance approach, while interference intervals not corresponding to the range values of the selected constellation are simply discarded. The maximum gross rate of the channel is about 1 kbps (i.e. 4 bit/frame, with a maximum inter-frame time of about 4 ms for the first constellation with the smaller payloads).

For the ZigBee-to-WiFi link, the main limit is due to the WiFi receiver behavior, which performs periodic resets in case of long interference intervals (every 1 ms) and measures wrong busy times when a Good PLCP event is erroneously triggered (with probability 1/4, as discussed in [30]). To cope with these features, we limited ZigBee symbols to frame durations lower than 1 ms (maximum packet length of 31 bytes), and defined a single constellation of 8 symbols spaced of 1 byte (i.e. 24–31 bytes). Indeed, since interfering transmissions due to other WiFi networks are generally longer than 1 ms and WiFi receivers have a very high carrier sense granularity, the probability to have a collision with intervals of equal duration due to other interferers is very low. The minimum gross rate of the channel is about 3 Kbit/s (i.e. 3 bit/frame, with a frame duration lower than 1 ms). However, there is a very high probability to have a wrong measurement of the interference duration in case of Good PLCP events. To mitigate this problem, it is necessary to use a channel coding scheme which reduces the channel rate according to the introduced redundancy level.

6.2.3.1 Communication Protocol

Different communication protocols can be defined on top of the inter-technology communication scheme, according to the specific application that exploits the WiFi/ZigBee communication channel. For example, ZigBee nodes involved in periodic (and sporadic) traffic could reserve in advance some channel access intervals for protecting the traffic from WiFi interference. The channel allocations can also be negotiated dynamically to adapt to traffic changes or give priority to a particular application. The channel can also be used for exchanging simple data: for example, common laptops or smart-phones, only equipped with WiFi interfaces, could read the measurements performed by ZigBee sensors or configure ZigBee actuators directly (e.g. an on/off power switch).



(a) (b)

Figure 15: Inter-technology communications:
(a) modulation scheme and (b) exemplary message for the WiFi-to-ZigBee link.

Figure 15 shows an exemplary message-based and byte-oriented communication protocol. Each message starts with a preamble (4 symbols following a pre-defined pattern) and includes the following fields: a header of 1 byte (i.e. two symbols) for specifying the message length; the payload, whose length may vary in the range 0-255 bytes; a final CRC of 2 bytes. It follows that the minimum message has a length of 10 symbols (4 preamble + 2 header + 4 CRC symbols) and requires the transmission of at least 10 different frames. The message preamble is built by transmitting special symbols not used for inter-technology data. As indicated in Figure 15-a for the WiFi-to-ZigBee link, the special symbols P(i) are placed right before the starting of each constellation i (belonging to [0; 3]) and symbols S j(i) of two different constellations are inter-leaved j ([0; 15]). Figure 15-b shows the temporal RSSI trace acquired by a USRP monitoring node corresponding to a real message. In the example, the payload is coding a control command activating a default log mode.

6.2.4 Experimental results

We implemented our WiFi-to-ZigBee and ZigBee-to-WiFi modulation and demodulation scheme on commercial devices. More into details, we built the inter-technology communication link by using an embedded Alix PC with a Broadcom bcm4318 WiFi card, and system-on-chip by Texas Instruments (CC2530) with Z-Stack [34]. An additional WiFi node has been used for producing environmental background traffic, which is the most common scenario. For the sake of simplicity, we avoided experiments with ZigBee background traffic, which could be added in a future work.

6.2.4.1 Link Reliability

Table 7 and Table 8 quantify the inter-technology link reliability, respectively, for the WiFi-to-ZigBee and ZigBee-to-WiFi links. In particular, the tables show the symbol error probability for eight different symbols, which correspond to eight symbols selected in one of the available constellation for the WiFi-to-ZigBee link, and to the whole set of available symbols for the ZigBee-to-WiFi link. The error probability is de-composed into two different figures: the loss rate, that is the probability of completely missing the reception of the symbol, because the interference is not recognized as a different technology interference or because the measured duration is not recognized as a valid symbol; the false positive rate, that is the probability of detecting one symbol different from the transmitted one.

Table 7: Loss rate and false positive rate for WiFi-to-ZigBee links (in percentage).

	$WiFi_H$		$WiFi_L$		$WiFi_H$ +$interf_L$		$WiFi_L$ +$interf_H$	
Symb.	**Loss**	**False**	**Loss**	**False**	**Loss**	**False**	**Loss**	**False**
S_1	0.00	0.45	0.00	0.20	0.00	3.30	1.10	14.95
S_2	0.39	1.55	0.00	0.20	0.00	3.30	3.40	14.55
S_3	0.00	0.30	0.00	0.20	0.00	3.15	3.85	14.05
S_4	0.00	0.10	0.00	0.25	0.00	2.40	7.75	14.35
S_5	0.00	0.10	0.05	0.25	0.00	2.00	0.00	13.40
S_6	0.00	0.40	0.05	0.25	0.00	2.20	0.00	14.85
S_7	0.00	0.20	0.00	0.35	0.00	2.70	0.95	14.65
S_8	0.00	0.10	0.00	0.50	2.05	0.20	0.95	14.70

Table 8: Loss rate and false positive rate for ZigBee-to-WiFi links (in percentage).

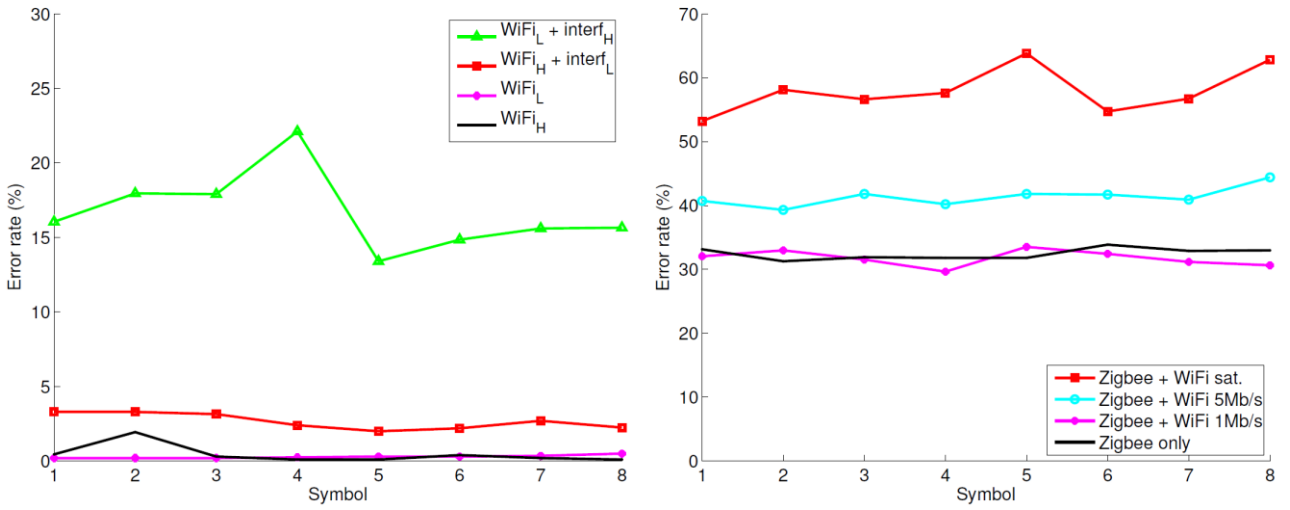
	ZigBee only		**ZigB.+$WiFi_{1Mb/s}$**		**ZigB.+$WiFi_{5Mb/s}$**		**ZigB.+$WiFi_{sat.}$**	
Symb.	**Loss**	**False**	**Loss**	**False**	**Loss**	**False**	**Loss**	**False**
S_1	33.13	0.00	29.37	2.68	40.30	0.40	52.50	0.70
S_2	31.26	0.00	31.26	1.71	38.60	0.70	57.60	0.50
S_3	31.89	0.00	28.20	3.33	41.50	0.30	56.10	0.50
S_4	31.80	0.00	26.67	2.97	39.80	0.40	56.60	1.00
S_5	31.62	0.18	33.06	0.45	39.60	2.20	61.60	2.20
S_6	33.78	0.09	31.44	0.99	39.60	2.10	50.60	4.10
S_7	32.70	0.18	30.63	0.54	38.90	2.00	52.70	4.00
S_8	32.61	0.36	29.82	0.81	41.20	3.20	56.40	6.40

In the first WiFi-to-ZigBee experiment, we configured different scenarios by varying the distance from the WiFi transmitter to the ZigBee receiver, as well as to the additional WiFi interferer. We classified two different levels of received powers (either the useful power or the interfering power) as high power and low power. Each level has been indicated with a subscript: the high-level power is -24dBm and the low-level power is -42dBm. For example, WiFiH means that the power received from the WiFi transmitter is -24dBm, while interfL means that the power received from the WiFi interferer is -42dBm. The WiFi interferer has been configured with a data rate of 1 Mbps, saturation traffic, and variable payload lengths (but different from the values of the constellation spaces). As evident from the table, in most of the considered scenarios the symbol errors are below 5%, and only with a powerful source of interference, false positives raise to 14%.

In the second ZigBee-to-WiFi experiment, we configured four different scenarios, in which ZigBee is transmitting to WiFi without interference or with an environmental traffic of increasing intensity (namely, 1 Mbps, 5 Mbps and saturation traffic with a fixed payload of 1500 bytes). The data rate of the WiFi interferer has been set to 36 Mbps, which leads a

packet transmission interval lower than the inter-technology constellation space. From Table 8, it is evident that false positives are very low even in saturated conditions thanks to the higher precision in the busy time measurements. However, losses are higher because Good PLCP events trigger the virtual busy time mechanism which destruct the real airtime measurement of symbol transmissions. It follows that in this direction redundancy schemes are necessary. However, since the communication speed is at least three times faster than in the other direction, the inter-technology channel capacity can be opportunistically balanced, leading to a symmetric bi-directional channel rate of approximately 1 kbps.

Figure 16 summaries the above results by showing the total symbol error rate, in the two directions, for all the described scenarios.



(a) (b)

Figure 16: Total error rate for WiFi-to-ZigBee links (a) and ZigBee-to-WiFi links (b).

6.2.4.2 Application Example

As an example of interesting application exploiting the inter-technology communication channel, we consider the possibility to set-up a TDMA scheme in which different technologies are allowed to transmit in non-overlapping periodic time intervals (with a frame structure), following their legacy MAC protocol within their slot. A control message is sent from the WiFi transmitter to the ZigBee nodes, coding a special command for activating this access mode. In case of positive feedback from the ZigBee coordinator, the WiFi transmitter sends a further command for configuring the slot size allocated to each technology and specifying the duration of a synchronization WiFi frame. ZigBee coordinator can confirm this allocation or asking for increasing or reducing its slot. In our experiments, such a negotiation lasts less than 0.5s. The bottom trace shown in Figure 17 is a channel access trace captured by the USRP monitoring node after the negotiation, when the TDMA mode is activated with an equal slot size for both the technologies. Different transmitters are identified by different RSSI values measured by the USRP channel sniffer. By comparing this trace with the upper one showing normal access operations, it can be inferred that the inter-technology TDMA scheme increases the rate of ACK transmissions (i.e. successful transmissions), which are represented by the narrow spikes following WiFi frames.

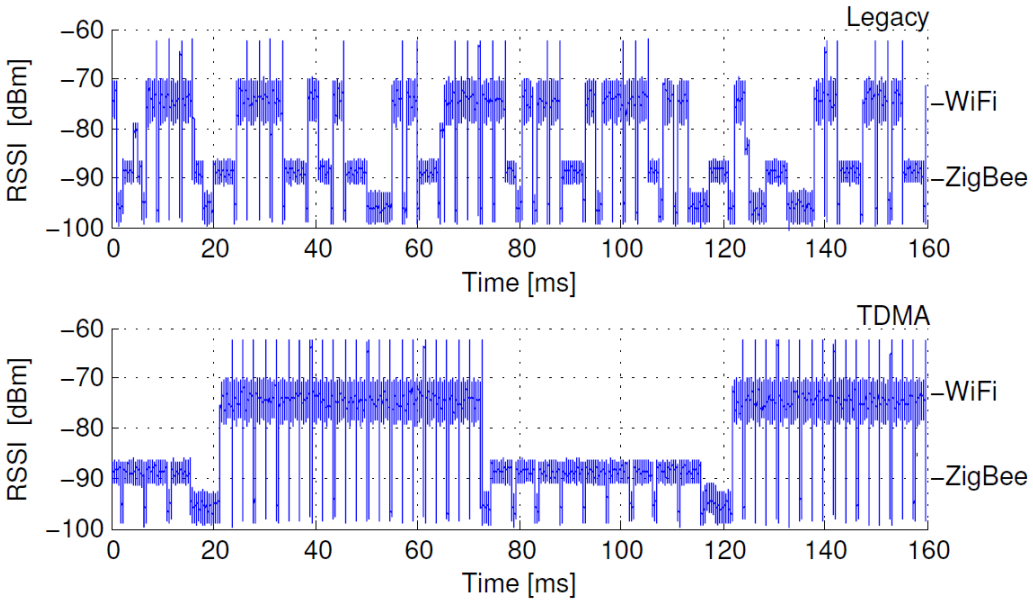


Figure 17: Coexistence between WiFi and ZigBee networks:
standard protocols (top trace) and inter-technology TDMA (bottom trace).

In conclusion, leveraging on the error patterns analysis and busy time measurements, it is possible to build a low-rate communication channels between ZigBee and WiFi coexisting networks, using commodity WiFi and ZigBee cards. Mapping interference durations to communication symbols, a bi-directional communication channel between the two coexisting networks is set up, achieving nominal speeds of about 1Kbps. This feature paves the way to innovative applications (e.g. reading measurements from ZigBee sensors directly by using common smart-phones with WiFi interfaces) and offers network designers the great opportunity to explicitly co-ordinate channel access between the interfering technologies (improving performances in highly congested scenarios), without the need of an external gateway or other indirect form of communication.

6.3 LoRa

We now consider the emerging LoRa (Long Range) technology, which represents a critical example of wireless technology working in high-density IoT settings. Indeed, LoRa technology is conceived for Low Power Wide Area Networks (LPWAN), with low data rate requirements per single device, large cells and heterogeneous application domains, leading to extremely high numbers of devices coexisting in the same cell. For this reason, LoRa provides different possibilities to orthogonalize transmissions as much as possible - Carrier Frequency (CF), Spreading Factor (SF), Bandwidth (BW), Coding Rate (CR) - and provide simultaneous collision free communications. However, despite the robustness of the LoRa PHY [35] patented by Semtech, in WAN scenarios where multiple gateways can be installed, the scalability of this technology is still under investigation [36]. Current studies are mostly based on simulation results [37] and assume that the utilization of multiple transmission channels and spreading factors lead to a system that can be considered as the simple superposition of independent (single channel, single spreading factor) sub-systems. This is actually a strong simplification, especially because the spreading factors adopted by LoRa are pseudo-orthogonal [38] and therefore, in near-far

conditions, collisions can prevent the correct reception of the overlapping transmissions using different spreading factors.

For characterizing these phenomena in this deliverable:

1. We analyze LoRa chirp modulation both numerically and experimentally, showing that collisions be-tween packets of different SFs can indeed cause packet loss if the interference received is strong enough.
2. We quantify the power threshold for which such loss occurs, for all combinations of SFs.
3. We modified and extended LoRaSim [39], an open-source LoRa simulator presented in [36], to measure the impact of both inter-SF collisions and fading (which was also not taken into account in previous works)
4. We developed a LoRa cell traffic generator, able to emulate the behavior of thousands of low-rate sensor nodes deployed in the same cell.

First, we analyzed numerically LoRa modulation (based on chirp spread spectrum) and LoRa access mechanism (based on aloha and duty cycle enforcement), including simple models for traffic sources and physical channels, and we developed a software receiver, able to process a signal trace received by a well-known USRP software-defined-radio (SDR) platform.

Second, we designed and implemented a traffic generator for LoRa-based networks (to be run on a second USRP platform) to generate, in a controlled and repeatable manner, a combined radio signal given by the superposition of multiple LoRa signals, generated by different devices operating in the same cell, as they would be seen by a LoRa gateway when colliding. We use these first two elements to prove that if a collision occurs between packets of different SF, both packets are correctly received only when the difference in received power is below a certain threshold (which we quantify for each combination of SFs), otherwise only the packet with highest RSSI is correctly received.

Third, to measure the impact caused by this non-orthogonality, we modified and extended the LoRaSim simulator to include such phenomena and model log-normal fading.

Fourth, we present and validate a LoRa cell traffic generator, able to emulate the behavior of thousands of nodes by using a single Software Defined Radio (SDR) platform. Differently from traditional generators, whose goal is creating packet flows which emulate specific applications and protocols, our focus is generating a combined radio signal, as seen by a gateway, given by the superposition of the signals transmitted by multiple sensors simultaneously active on the same channel.

Our experimental results show that non-orthogonality of the SFs can deteriorate significantly the performance of LoRa communications, especially of higher SFs (10 to 12) where packets last longer and are thus more vulnerable to collisions. With multiple neighboring cells, this phenomenon is further exacerbated and, in some scenarios, high SFs can experience severe losses. In the same scenarios, when considering the SFs perfectly orthogonal, the capacity is significantly overestimated, as the measured losses are almost halved. We also demonstrate that in multi-gateway or multi-cell topologies fading has virtually no impact on performance, because antenna diversity gives more chances to receive the same packet on different gateways. Finally, with our LoRa cell traffic generator, we are able to test the performance of real hardware (gateways and devices) under different LoRa network planning solutions.

6.3.1 LoRaWAN networks and protocols

In this section we briefly describe the LoRaWAN network architecture and MAC/PHY protocols, in order to present the main features that affect the network capacity. LoRaWAN is a protocol specification built on top of the LoRa PHY technology, a PHY layer patented by Semtech that works on ISM bands (namely, at 868Mhz with 3 default channels in Europe) with a limited duty cycle allowed to each device [41]. The typical LoRaWAN network consists of end-devices (sensors or actuators), multiple gateways which forward packets coming from the wireless medium to a backhaul interface, and a logically centralized server which analyzes information collected by all the devices and optimizes the network configuration.

Differently from traditional cellular networks, end-devices are not associated to a particular gateway to have access to the network. Gateways serve simply as link layer relayers and forward all the packets received from end-devices to the network server, by adding information regarding the reception quality. Thus, in general, the same packet can be forwarded by multiple gateways and the central server is responsible of detecting duplicate packets and choosing the appropriate gateway for transmitting downlink packets (if any). Gateways are usually equipped with multiple transceivers for receiving simultaneously on multiple (configurable) frequency channels, while end devices can dynamically select one transmission channel, among a set of available ones, at each transmission attempt. Some channels are reserved for data transmission, one channel is reserved for gateway's responses, while some other channels are usually served for control information and in particular for transmitting network joining requests from end devices.

LoRaWAN provides an open source MAC layer that is based on a simple Aloha protocol, in order to minimize the complexity and the energy consumption of the devices. End devices do not perform carrier sense; moreover, they listen to the medium for receiving packets only in special time windows after uplink transmissions (class A devices), or at regular time intervals (class B devices). Only class C devices have continuously active receivers. Frames can be acknowledged or not. In the acknowledged mode, the modulation format can be automatically adapted as function of the number of retransmissions (for increasing or decreasing robustness).

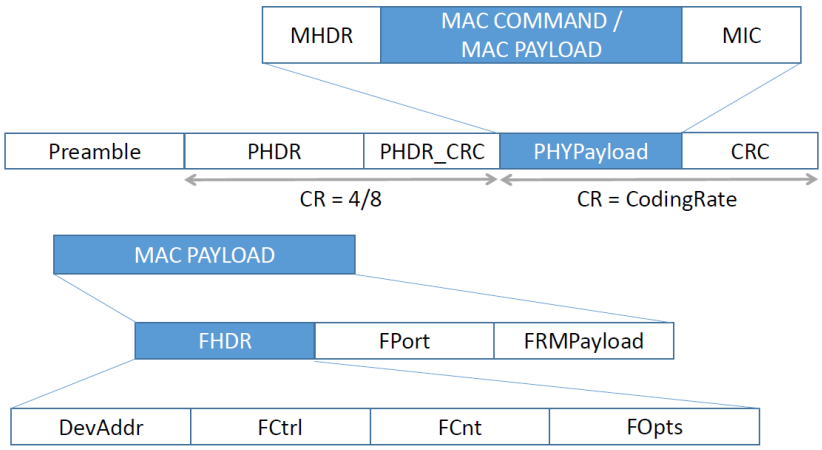


Figure 18: LoRa frame structure.

Figure 18 summarizes the structure of a LoRaWAN frame. The frame starts with a preamble, which is used for synchronization and for defining the modulation format

(Specifically, the spreading factor used for modulating the frame, as clarified in the following sub-section). The preamble is followed by a PHY Header and a Header CRC, whose total length is 20 bits encoded with the most reliable code rate of 4/8. The PHY header contains information such as payload length and whether the optional payload CRC is included or not in the frame. The MAC header is one byte only; it defines the protocol version and message type, i.e., whether it is a data or a management frame, whether it is transmitted in uplink or downlink, whether it shall be acknowledged. Finally, the Frame header contains network-level information, such as the device address (composed of a network identifier and device-specific identifier given upon association), network control information about the configuration of the uplink data rate parameters, a sequence number, and frame optional data for commands. A port number is used for distinguishing flows between gateways and end devices, including flows containing MAC commands for the configuration of devices.

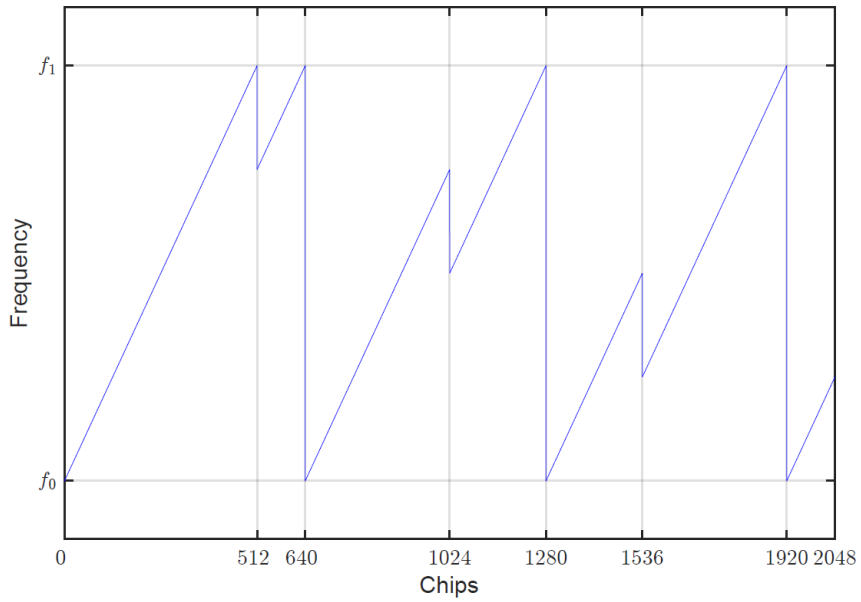


Figure 19: Modulating signal with SF = 9 for one basic upchirp and three symbols: 128, 256 and 384.

6.3.1.1 LoRa modulation

LoRa modulation is derived from Chirp Spread Spectrum (CSS), which makes use of chirp signals, i.e. frequency-modulated signals obtained when the modulating signal varies linearly in the range [f0; f1] (upchirp) or [f1; f0] (downchirp) in a symbol time T. Binary modulations, mapping 0/1 information bits in up-chirps/downchirps, have been demonstrated to be very robust against in-band or out-band interference [35]. LoRa employs a M-ary modulation scheme based on chirps, in which symbols are obtained by considering different circular shifts of the basic upchirp signal. The temporal shifts, characterizing each symbol, are slotted into multiples of time Tchip = 1/BW , called chip, being BW = f1-f0 the bandwidth of the signal. It results that the modulating signal for a generic n-th LoRa symbol can be expressed as:

$$f(t) = \begin{cases} f_1 - n \cdot k \cdot T_{chip} + k \cdot t & \text{for } 0 \leqslant t \leqslant n \cdot T_{chip} \\ f_0 + k \cdot t & \text{for } n \cdot T_{chip} < t \leqslant T \end{cases}$$

where k = (f1-f0)/T is the slope of the frequency variations. The total number of symbols (coding SF information bits) is chosen equal to 2^{SF}, where SF is called spreading factor. The symbol duration T required for representing any possible shift is 2^{SF} • Tchip = 2^{SF}/BW. It follows that, for a fixed bandwidth, the symbol period increases with SF, enlarging the temporal occupancy of the signal. Figure 19 shows the modulating signal used for a basic upchirp and three examples of circular shifts obtained for SF = 9: the symbol time is 512 Tchip, while the three exemplary shifts code the symbols 128, 256 and 384. The preamble of any LoRa frame is obtained by sending a sequence of at least ten upchirps, followed by two downchirps. Payload data are then sent by using the M-ary modulation symbols. LoRa provides three BW settings (125, 250 or 500 KHz) and seven different SF values (from 6 to 12). In general, a larger bandwidth translates in a data rate increase and a receiver sensitivity deterioration. Conversely, higher spreading factors can be used for improving the link robustness at the cost of a lower data rate.

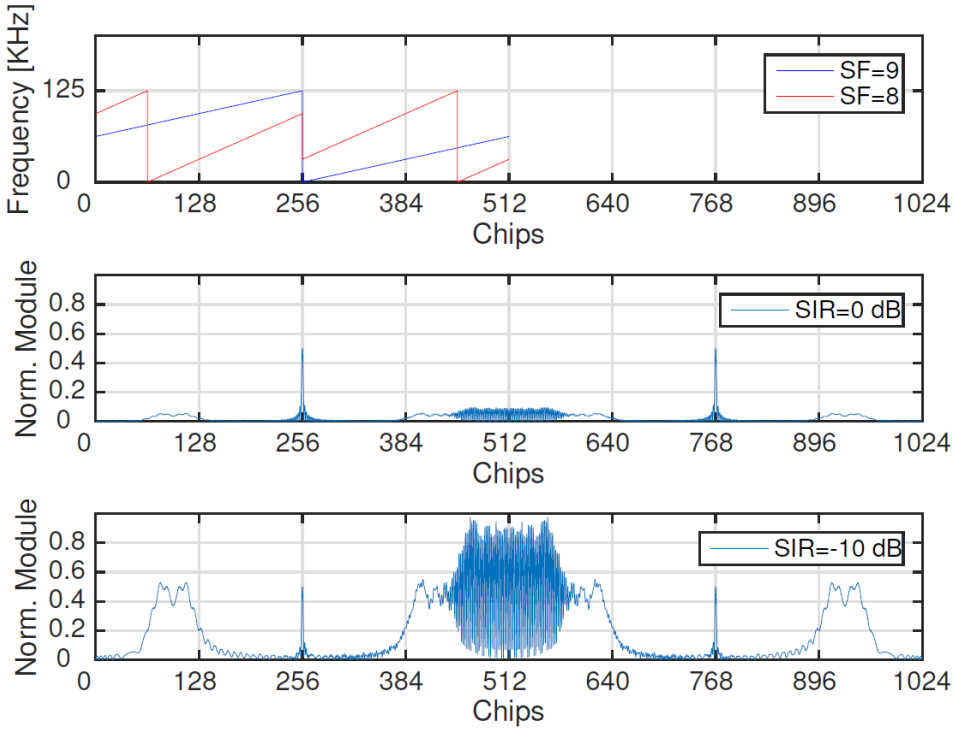


Figure 20: An example of combined signal, obtained by the sum of two modulated chirps with SF = 9 (blue line) and SF = 8 (red line), and correlation results with the basic upchirp with SF = 9.

An interesting feature of LoRa modulation is the pseudo-orthogonality of signals modulated under different spreading factors. This feature can be exploited for enabling multiple concurrent transmissions that can be easily separated by the receiver thanks to the fact that the cross-energy between two signals, say s1(t) and s2(t), modulated with different spreading factors is almost zero regard-less of the starting of the symbol times:

$$E_{s_1,s_2}(\tau) = \int_0^T s_1(t) \cdot s_2(t-\tau)^* dt \simeq 0$$

where T is the symbol period of the signal with the highest spreading factor. An example of signal separation is illustrated in Figure 20, where one signal modulated with SF = 9,

representing symbol 256 (blue line), is overlapped to two symbols modulated with SF = 8 (red line). For sake of presentation, in the figure, the symbol times of the two signals start at the same time. For receiving the signal modulated with SF = 9 it is enough to correlate the overlapped signal with the relevant SF = 9 upchirp signal. The middle plot of the figure shows the correlation results (i.e. the variation of the cross-energy between the overlapped signal and the upchirp signal at different temporal shifts). The figure clearly highlights two peaks when the upchirp signal is shifted of 256 and 256 + 29 chip times Tchip: indeed, in these positions the similarity of the sum signal with the basic upchirp at SF = 9 is high because the upchirp overlaps with one of the two piecewise chirps of the symbol modulated with SF = 9. Conversely, in the other positions, the correlation is almost zero as the symbol modulated with SF = 8 and the upchirp are almost orthogonal to the basic upchirp.

Note that the separation mechanism works as long as the amplitudes of the overlapping signals are comparable or belong to a given range. The bottom plot of Figure 20 shows the correlation results when the signal modulated with SF = 8 has an amplitude 20 times higher than the amplitude of the symbol at SF = 9. In this case, correlation peaks due to cross-energy can occur in positions different from the symbol time (e.g. around 512 Tchip) and demodulation errors can occur.

6.3.2 Impact of Spreading Factor Imperfect Orthogonality

In this section we study the impact of imperfect-orthogonality in LoRa spreading factors (SFs) in simulation and real-world experiments. We analyze LoRa modulation numerically and show that collisions between packets of different SFs can indeed cause packet loss if the interference power received is strong enough. Then, we validate such findings using commercial devices, confirming our numerical results. Finally, we modified and extended LoRaSim, an open-source LoRa simulator, to measure the impact of inter-SF collisions and fading (which was not taken into account previously in the simulator). Indeed, since LoRa is quite a recent technology, relatively few works have already been published on its performance and most of the current studies are based on simulation results. For example, authors in [36] implemented and used the LoRaSim simulator [39] to investigate the capacity limits of LoRa networks. They showed that a typical deployment can support only 120 nodes per 3.8 ha, although performance can be improved with multiple gateways. The paper in [38] compared the performance of LoRa and ultra-narrowband (Sigfox-like) networks, showing that ultra-narrowband has a larger coverage but LoRa networks are less sensitive to interference. Finally, authors in [40] presented details on the patented LoRa PHY and introduced gr-lora, an open source SDR-based implementation of LoRa PHY.

All of these works, however, assume that the SFs adopted by LoRa are perfectly orthogonal, thus simplifying the analysis and consequently the network capacity. Instead, in this deliverable we show that, due to the imperfect orthogonality of the SFs, inter-SF collisions can prevent the correct reception of the transmissions with serious impact on LoRa performances. Our results show that non-orthogonality of the SFs can deteriorate significantly the performance especially of higher SFs (10 to 12) and that fading has virtually no impact when multiple gateways are available in space diversity.

Our goal is to quantify how the power ratio between a reference signal and an overlapping interfering signal can affect the demodulation results, even when the interfering signal is transmitted with a spreading factor different from the desired signal one. To this purpose, we consider both simulation results, based on a MATLAB implementation of a multi-

channel LoRa demodulator, and experimental results, based on a commercial LoRa receiver and a USRP used as an artificial synthesizer of LoRa overlapped signals.

6.3.2.1 MATLAB simulation results

We implemented a LoRa demodulator in MATLAB based on the parallel correlation of the received signal with a bank of upchirp signals modulated with different spreading factors (from SF = 7 to SF = 12). For identifying the transmission symbol, we observed that for circular shifts lower than an half of the symbol period (i.e. for the first half of all the possible symbols), the first correlation peak has a smaller amplitude than the second one. This is due to the fact that the first piecewise chirp overlaps with the reference upchirp for a lower duration that the second one. On the contrary, for symbols shifted of more than T=2, the first peak has a bigger amplitude than the second one. We exploited this property for implementing a decision logic based on the identification of the highest correlation peak. In brief, the MATLAB demodulator works as follows: i) cross-correlating the received signal and the reference upchirp at the selected spreading factor; ii) looking for two (related) correlation peaks within the symbol periods, whose distance is T ; iii) choosing the highest peak for quantifying the symbol shift.

We also implemented another demodulation solution, proposed in [42] and [43], which works by multiplying the received signal with the basic downchirp signal in order to obtain two piecewise signals at a fixed frequency fi and fi+BW (whose value fi is proportional to the circular shift of the symbol), by down-sampling the resulting signal with a sampling rate equal to the bandwidth of the signal for creating an alias of the signal fi + BW at frequency fi and by finally performing an iFFT on the signal for identifying fi.

We performed a number of simulations for testing the MATLAB implementations of LoRa demodulator in case of reception of two overlapping signals modulated with different spreading factors. Our goal is identifying a Signal to Interference Ratio (SIR) threshold under which the correct demodulation of the desired signal is affected by errors. In each simulation run, we created an overlapped signal by summing one random symbol modulated with a reference spreading factor SFref with a number of random interfering symbols modulated with a different spreading factor SFint. For each run we can distinguish two cases: if SFref > SFint then the reference symbol is combined with more than one interfering symbol (e.g. for SFref = 12 and SFint = 10, the number of interfering symbols N overlapping with the reference signal is equal to 4). Conversely, if SFref < SFint, the same interfering symbol is overlapped to multiple symbols of the reference signal. The amplitude Aint of the interferer signal is set to one, whereas the amplitude Aref of the reference is multiplied with a tunable value leading to different SIR values:

$$A_{int} = \sqrt{10^{-SIR/10}} \cdot A_{ref}$$

The resulting combined signal has been then processed by the MATLAB de-modulators, in absence of noise on the channel and assuming perfect synchronization of the symbols. The results achieved with the two implementations (correlation- and FFT-based) are comparable both in terms of symbol demodulation performance and complexity. In the following, we only show results for the FFT-based demodulator.

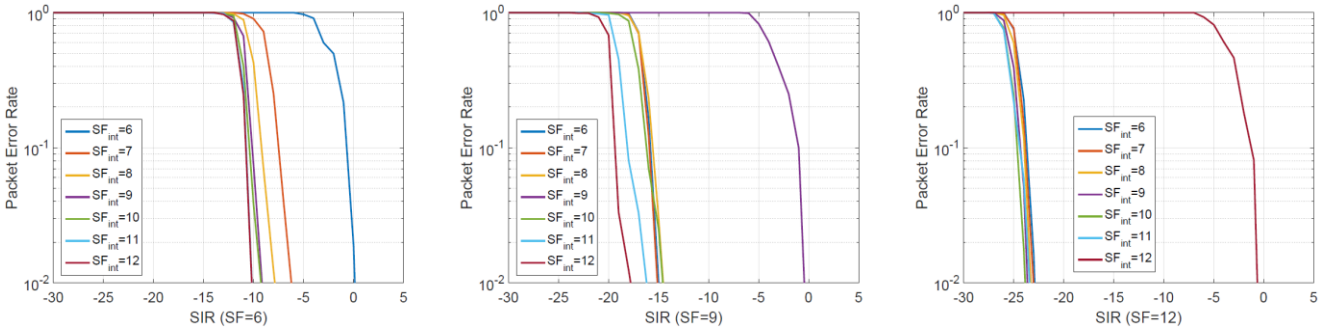


Figure 21: PER of three different spreading factors in function of the SIR.

A Packet Error Rate (PER) statistic has been generated comparing the de-modulated symbols with the modulated ones. For each simulation run, we randomly generated 100 packets of 10 Bytes each with SFref against a correspondingly number of symbols with SFint (increased or decreased by appropriate powers of 2 to cover the SFref symbol length), and varied the SIR values from 30 dB to +6 dB with steps of 1:5 dB. Figure 21 shows the results of these simulations for three different SFref values as an example. The curves in the figures represents the error probability for one selected SFref against all the possible SFint obtained with our MATLAB demodulator. Similar results have been obtained with the other implementation. From the figure, it can be easily recognized that for each SFref it exists a minimum SIR threshold below which the success probability starts degrading (high PER). Furthermore, the smaller the interfering spreading factor, the higher the SIR must be to have an acceptable PER.

Table 9: SIR thresholds of correct demodulation for the different SFs in Matlab simulations.

SF_{ref} \ SF_{int}	6	7	8	9	10	11	12
6	0	-6	-7	-9	-9	-10	-10
7	-8	0	-9	-10	-12	-13	-13
8	-12	-11	0	-11	-13	-15	-15
9	-15	-15	-14	0	-14	-16	-17
10	-18	-18	-18	-17	0	-17	-19
11	-20	-20	-20	-20	-20	0	-20
12	-23	-23	-23	-23	-23	-23	0

Table 9 summarizes the SIR thresholds considering a PER of 1%. In the table, we also consider the case when the interfering signal has the same spreading factor of the reference signal. As also documented in the Semtech specifications, LoRa modulations achieves very high probability of capture effects even with low SIR values (approximately 0dB for the different spreading factors in our simulations, versus 6dB specified in [44]). In other words, it is very likely that in case of collisions between two signals modulated with the same spreading factor, the strongest signal can be correctly demodulated.

6.3.2.2 USRP experiments

To validate the thresholds found with the Matlab simulator, we performed a number of experiments on real LoRa links in case of continuous collisions of the packets. To this purpose we used a Semtech SX1272 transceiver, controlled by an Arduino Yun, for sending packets (with a 19-byte payload) modulated with all the possible spreading factors

and a USRP B210 for recording the traces. After this preliminary phase, we manipulated the traces extracting exactly one packet for each SFref and adding a variable silence period to it as inter-packet delay. This operation gave us the necessary flexibility for combining traces at different spreading factors through Gnuradio and send them over the air via the USRP. Using two file source blocks, we loaded continuously one trace at SFref and one trace at SFint. The delay between packets at different spreading factors was set in order to achieve the condition of continuous collision.

Figure 22: An example of collision between SFref = 12 (the darker chirps) and SFint = 10 (the brighter chirps) with SIR = 4dB.

Figure 22 shows an example of overlapping symbols modulated with SFref = 12, SFint = 10 and with a SIR = 4 dB. We then: i) modified the amplitude of the interferer packets according to the desired SIR level through a multiplier block; ii) summed the two traces with a sum block; iii) sent the combined signal over the air through an USRP block. Then, the combined signal sent by the USRP was received by the SX1272 transceiver which logged, through an Arduino LoRa library, the packets received correctly. Finally, the packet success probability was calculated. The results of the experiments are summarized in

Table 10, for the same subset of reference and interfering spreading factors presented in the previous subsection. Comparing this table with Table 9 it can be observed that the thresholds for correct demodulation are higher with the real transceiver.

Table 10: SIR thresholds of correct demodulation for different SFs with SX1272 transceiver.

SF_{ref} \ SF_{int}	7	8	9	10	11	12
7	3	-5	-6	-7	-8	-8
8	-6	3	-7	-8	-10	-10
9	-7	-7	3	-9	-11	-11
10	-9	-9	-9	3	-12	-13
11	-13	-13	-13	-13	3	-14
12	-16	-16	-16	-16	-16	3

6.3.3 Simulation-based capacity results

To measure the impact caused by the non-orthogonal SFs, we modified and extended the LoRaSim simulator [39] to include such phenomena. We modified LoRaSim so that if a collision occurs between packets of different SF, both packets are correctly received only when the difference in received power is below a certain threshold (in the experiments we set a conservative threshold of 6 dB for all collisions), otherwise only the packet with highest RSSI is received and the other discarded[7]. We also extended the simulator to account for log-normal fading which was not implemented yet. For the sake of brevity, we provide here a summary of the findings, while the complete simulation-based study can be found in appendix.

From our results, already with one LoRa cell in isolation, the performance deteriorates quickly as the traffic rate of the nodes increases, especially for high SF (10-12) where packets last longer and are thus more vulnerable to collisions. The situation worsens when considering also neighboring cells where inter-SF collisions deeply affect the performance of the highest SFs. For example, in a scenario with 16 cells, SF 12 experiences 30% loss already with 500 nodes, while if we consider the SFs as orthogonal the loss is almost halved.

Considering the impact of fading, our simulations show that fading has a certain impact on the single cell scenario, while in presence of with multiple cells fading has virtually no impact on performance. This is due to antenna diversity, which gives more chances to receive the same packet on different gateways, compensating the probability of being lost on one link because of fading.

As detailed in the appendix, these results are particularly important for network operators to correctly plan the parameters of their LoRa networks. In particular, increasing the SF might not always be a good solution to improve performance in presence of congestion or to compensate fading channels. Indeed, despite being more robust to noise, our results show that higher SFs are more vulnerable to inter-SF collisions and that fading becomes negligible in presence of multiple base stations. Instead, it could be more effective to use a low SF in order to lower the chances of packet collisions.

In conclusion, because of imperfect orthogonality between different spreading factors, LoRa network cell cannot be studied as a simple super-position of independent networks working on independent channels. Although parallel reception of multiple overlapping frames is generally possible, when the power of the interfering signal significantly overcomes the reference signal, the correct demodulation of the reference signal can be prevented. This can happen in typical near-far conditions, when the interfering signal is much closer to the gateway than the reference signal, or when multiple interfering signals are simultaneously active. Deploying multiple gateways can mitigate the capacity loss due to these phenomena, because each gateway experiences a different SIR value. Moreover, it also mitigates the probability of bad channel conditions for the reference signal due to fading.

Implications of imperfect orthogonality between different spreading factors for network planning are still under investigation. For example, allocating higher spreading factors, characterized by lower receiver sensitivities, to far users could not necessarily improve their link capacity in case of congested networks. Indeed, higher spreading factors could be more prone to collisions due to longer transmission times.

[7] We omit the case of collisions between packets of different BW for the sake of simplicity.

6.3.4 Cell-level Traffic Generator

High-density LoRa deployments are difficult to study in real testbeds or installations, especially when we can expect that hundreds or even thousands of mobile devices (i.e. smart objects and sensors) are simultaneously active in the same environment. For this reason, we developed a synthetic LoRa cell traffic generator, in order to test the performance of real hardware (gateways and devices) under different LoRa network planning solutions.

While traditional traffic generators are limited to the generation of packet flows emulating specific applications and protocols [46], [47], our focus is on the generation of a combined radio signal given by the super-position of the signals, generated by multiple devices operating in the same cell, as seen by a gateway. Similar approaches have been recently proposed for studying high-density WiFi networks [48], resulting from installations in a stadium, a conference hall, an airport, etc. Commercial solutions try to emulate multiple devices by means of a special contention scheduler working on a single radio interface [49], [50] or by embedding multiple radio interfaces in the same traffic generator [51]. SDR-based solutions, summing overlapping signals in software before generating the resulting aggregated signal on the air, have also been explored in [52] for studying high-density cognitive networks.

Our solution is based on SDR and software-based combination and scheduling of signals to be transmitted in real-time, according to a desired traffic and cell scenario. To this purpose, we modeled LoRa modulation (based on chirp spread spectrum) and LoRa access mechanism (based on aloha and duty cycle enforcement). Simple models for traffic sources and physical channels are also included. We also provide a software receiver, able to process the trace produced by the traffic generator and evaluate low-level statistics to be compared with the statistics of a real gateway.

The LoRa cell configuration, specified by means of a GUI, includes the tuning of the following parameters: number of nodes, nodes distribution (uniform, uniform in a given area), source rates, bandwidth, spreading factor allocation (random, distance-based), transmission power (uniform, distance-based). The parameters can be differentiated for groups of nodes, thus facilitating the planning for large cells. Finally, in Figure 23 and Figure 24 are shown the GUIs of the transmitter and receiver modules of the emulator. Details of our traffic generator can be found in Appendix.

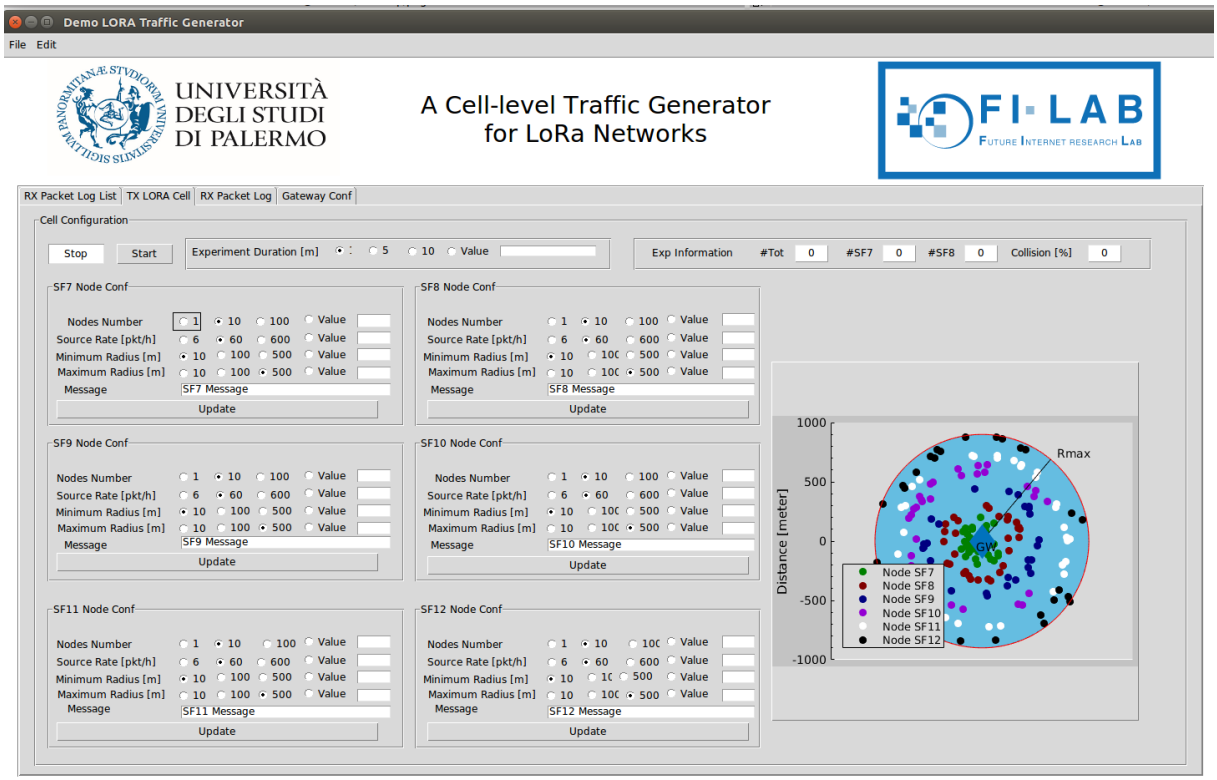


Figure 23: The transmitter GUI of the LoRa cell-level emulator.

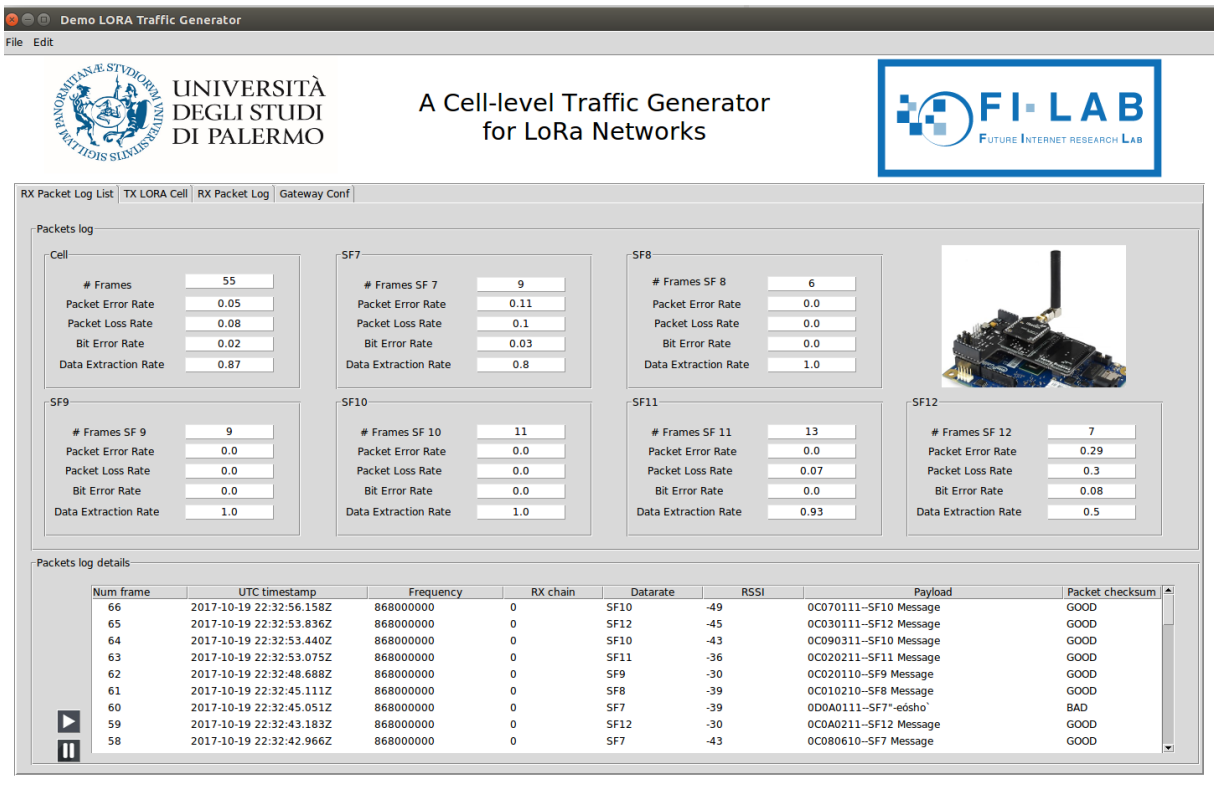


Figure 24: The receiver GUI of the LoRa cell-level emulator.

7 Conclusions and Next Steps

This deliverable detail the ecosystem needs to interact with devices at the lower level. It shows the communication mechanism between the various software modules that forms the S3M, the security mechanism used to communicate between the local agent and the SSP, and the relationship between the CLD components – reporting all the activities of Task 4.3.

The deliverable also reports on the activities of Task 4.2, studying the coexistence of different wireless technology in the same SSP environment, focusing on wireless radio protocols like WiFi and ZigBee. Indeed, coordination of wireless technologies and devices (causing both intra and inter-technology interference) is becoming necessary due to performance problems appearing when multiple users, gateways and technologies compete for limited spectrum. This communication can be implemented on SSP-level (as opposed to SymbIoTe L1 or L2 communication) as it would probably bring most benefits. Moreover, the wireless analysis deeply investigates the LoRa radio protocol and the corresponding network architecture.

At the status of the deliverable, the skeleton of the SSP middleware is defined and implemented. Final part of the work is to fine-tune all the interfaces to make the middleware an useful and commercially attractive system. To achieve that, users must find it useful which means that applications have to be already available at the time users start running it. In other words, it is important to bring into the SymbIoTe community as much platform and application as possible. Makers community is also a good audience for the symbIoTe services. In order to access this, good source of users' library and platform should be integrated into the project.

8 References

[1] RFC-2616 - Hypertext Transfer Protocol -- HTTP/1.1
[2] RFC-5246 - The Transport Layer Security (TLS) Protocol Version 1.2
[3] IANA-TLS - Transport Layer Security (TLS) Parameters, URL: https://www.iana.org/assignments/tls-parameters/tls-parameters.xml
[4] RFC-7159 - The JavaScript Object Notation (JSON) Data Interchange Format
[5] RFC-4868 - Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec
[6] RFC-5869 - HMAC-based Extract-and-Expand Key Derivation Function (HKDF)
[7] RFC-8018 - Password Based Key Derivation Function 2 (Appendix-B.1)
[8] RFC-6265 - HTTP State Management Mechanism
[9] SymbIoTe Deliverable 2.4: Revised Semantics for IoT and Cloud resources.
[10] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html
[11] 3GPP Release 13, http://www.3gpp.org/release-13
[12] 3GPP Release 10, http://www.3gpp.org/specifications/releases/70-release-10
[13] S. Rayanchu, A. Patro, and S. Banerjee. Airshark: detecting non-WiFi RF devices using commodity wifi hardware. In Proc. of IMC 2011.
[14] S. Pollin, I. Tan, B. Hodge, C. Chun, and A. Bahai. Harmful Coexistence Between 802.15.4 and 802.11: A Measurement-based Study. In Proc. of CrownCom, 2008.
[15] R. Gummadi, D. Wetherall, B. Greenstein, S. Seshan. Understanding and Mitigating the Impact of RF Interference on 802.11 Networks. In Proc. of ACM SIGCOMM ’07, Pages 385-396.
[16] J. Huang; G. Xing; G. Zhou; R. Zhou. Beyond Co-existence: Exploiting WiFi White Space for ZigBee Performance Assurance. ICNP, 2010.
[17] X. Zhang, K. G. Shin. Enabling Coexistence of Heterogeneous Wireless Systems: Case for ZigBee and WiFi. In Proc. of ACM MobiHoc ’11.
[18] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis. Surviving WiFi Interference in Low Power ZigBee Networks. In Proc. of SenSys 10, pages 309-322, 2010.
[19] A. M. Cavalcante et al., Performance Evaluation of LTE and WiFi Co-existence in Unlicensed Bands, 2013 IEEE 77th Vehicular Technology Conference (VTC Spring), Dresden, 2013, pp. 1-6.
[20] Yubing Jian, Chao-Fang Shih, Bhuvana Krishnaswamy, Raghupathy Sivakumar. Coexistence of WiFi and LAA-LTE: Experimental Evaluation, Analysis and Insights. IEEE International Conference Communication Workshop (ICCW), 2015
[21] P. De Valck, I. Moerman, D. Croce, F. Giuliano, I. Tinnirello, D. Garlisi, E. De Poorter, B. Jooris, Exploiting programmable architectures for WiFi/ZigBee inter-technology cooperation, EURASIP Journal on Wireless Communications and Networking, Vol. 1, pp. 1–13, 2014.
[22] K. Lakshminarayanan, S. Sapra, S. Seshan, and P. Steenkiste. RF-Dump: An Architecture for Monitoring the Wireless Ether. In Procs. of CoNEXT 09, Dec. 2009.
[23] O. Zakaria. Blind signal detection and identification over the 2.4 GHz ISM band for cognitive radio. In MS Thesis USF09
[24] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty. Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey. Comput. Netw., 2006.

[25] D. Croce, D. Garlisi, F. Giuliano and I. Tinnirello, Learning from Errors: Detecting ZigBee Interference in WiFi networks, in Proc. 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET) 2014.
[26] R. Gummadi, H. Balakrishnan, and S. Seshan. Metronome: Coordinating Spectrum Sharing in Heterogeneous Wireless Networks. 1st Int. Workshop on Communication Systems and Networks (COM-SNETS), 2009.
[27] S. Gollakota, F. Adib, D. Katabi, and S. Seshan. Clearing the RF smog: making 802.11n robust to cross-technology interference. In Proc. of ACM SIGCOMM 2011, pages 170-181, 2011
[28] X. Zhang, K. G. Shin. Enabling Coexistence of Heterogeneous Wire-less Systems: Case for ZigBee and WiFi. ACM MobiHoc 2011.
[29] Xinyu Zhang, K.G. Shin. Gap Sense: Lightweight coordination of het-erogeneous wireless devices. in Proc. of IEEE INFOCOM 2013, Turin Italy, pp.3094-3101.
[30] IEEE 802.11 Registers: http://bcm-v4.sipsolutions.net/802.11/Registers/
[31] srsLTE Github repository: https://github.com/srsLTE/srsLTE
[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau. ”Scikit-learn: Machine Learning in Python”. Journal of Machine Learning Research Vol. 12, pp. 2825-2830, 2011.
[33] Kameswari Chebrolu and Ashutosh Dhekne. Esense: communication through energy sensing. In Proc. 15th international conference on Mobile computing and networking (MobiCom ’09). Beijing, China.
[34] CC2530 Development Kit and Z-Stack (see http://www.ti.com/tool/cc2530dk).
[35] Semtech. LoRa Modulation Basics. AN1200.22, Revision 2, May 2015. www.semtech.com
[36] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, “Do LoRa Low-Power Wide-Area Networks Scale?”, In Proc. of MSWiM 2016. ACM, New York, pp. 59-67.
[37] B. Reynders and S. Pollin, "Chirp spread spectrum as a modulation technique for long range communication", In SCVT 2016, Mons, pp. 1-5.
[38] B. Reynders, W. Meert and S. Pollin, "Range and coexistence analysis of long range unlicensed communication", In ICT 2016, Thessaloniki, pp. 1-6.
[39] Available at http://www.lancaster.ac.uk/scc/sites/lora/
[40] M. Knight, B. Seeber. Decoding LoRa: Realizing a Modern LPWAN with SDR. Proceedings of the GNU Radio Conference, [S.l.], v. 1, n. 1, sep. 2016.
[41] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, Lorawan specification v1.0, Technical report, LoRa Alliance, Tech. Rep., 2015.
[42] O. Bernard, A. Seller, N. Sornin, "Low power long range transmitter", European Patent Application EP 2763321 A1 by Semtech Corporation, 2014
[43] C. Goursaud, J.M. Gorce, "Dedicated networks for IoT: PHY / MAC state of the art and challenges", in EAI endorsed transactions on Internet of Things, 2015.
[44] Semtech Corporation, "LoRa SX1272/73 Transceiver Datasheet", 2015.
[45] SimPy Event discrete simulation for Python. Available at https://simpy.readthedocs.io.
[46] L. Kelch, T. Pigel, L. Wolf, A. Sasse, "Traffic Generator for HSDPA Network Simulations", In ICCVE 2014, Vienna, 25-33.
[47] D. Kaulbars, F. Schweikowski, C. Wietfeld, "Spatially Distributed Traffic Generation for Stress Testing the Robustness of Mission-Critical Smart Grid Communication", In Proc. of GLOBECOM Workshops 2015, San Diego.
[48] G. Alvarez et al., "Multi-client Emulation Platform for Performance Testing of High Density 802.11 Networks", In Wireless Days (WD) 2016, 1-6.

[49] CT520 LANforge WiFIRE 802.11a/b/g/n WiFi Traffic Generator. www.candelatech.com
[50] Multi-Client Traffic Generator/Performance Analyzer www.veriwave.com
[51] TM500 Network Tester www.cobhamwireless.com
[52] S. Papadakis, A. Makrogiannakis, M. Surligas, P. Papadakis, Single SDR Multiple Primary & Secondary Users Emulation Platform, in EuCNC 2015, Paris.

9 Appendix

9.1 SymbIoTe Smart Space Middleware

Component/service name	Smart Space Middleware
URL of source codes	https://github.com/symbiote-h2020/SmartSpaceMiddleware
URL of Javadoc documentation	Under development
List of features included	• Innkeeper • SSP RAP • SymbIoTe Agent • IoT platform agent
Notes	The components listed above are the key S3M features. These will be complemented by new ones (e.g. SSP Search) in the next software releases.

9.2 Local AAM

Component/service name	Local AAM logic
URL of source codes	https://github.com/symbiote-h2020/AuthenticationAuthorizationManager
URL of Javadoc documentation	Generated on-demand using swagger gradle task.

List of features included	• Users management
	  ○ Accounts
	  ○ Authorization attributes
	• Users' authentication and authorization payloads management

Notes	The logic of this component is to be extracted to be compliant with the monolithic deployment idea. Security Payloads and documentation are available https://github.com/symbiote-h2020/SymbIoTeSecurity

9.3 Security Negotiation Protocol - Messages Examples

Protocol messages examples include Request and Response from SDEV to Gateway/Innkeeper (and viceversa) in JSON.

9.3.1 JSON Data-Format

When JSON [RFC-7159] is adopted for carrying encrypted data, should contains the following definitions:

```
Content-Type: application/json
Accept: application/json
```

9.3.2 SDEV HELLO

```
{
  "SDEVHello": {
    "mti": "0x10",
    "SDEVmac": "aa-bb-cc-dd-ee-ff",
    "cp": "0x00a8, 0xccab, 0x008c, 0xc004, 0xc005",
    "kdf": "PBKDF2",
    "nonce": "7050182458",
    "x509":
"MIIDXTCCAkWgAwIBAgIJAJC1HiIAZAiIMA0GCSqGSIb3DfBAYTAkFVMRMwEQYDVQQ
IDApTb21lLVN0YXRlMSEwHwYDVxaWRnaXRzIFB0eSBMdGQwHhcNMTExMjMxMDg1OTQ
0WhcNMTAJjyzfN746vaInA1KxYEeI1Rx5KXY8zIdj6a7hhphpj2E04C3Fayua4DRHy
ZOLmlvQ6tIChY0ClXXuefbmVSDeUHwc8YuB7xxt8BVc69rLeHV15A0qyx77CLSj3tC
x2IUXVqRs5mlSbvA=="
  }
}
```

9.3.3 GW_INK HELLO

```
{
  "GWInnkeeperHello": {
    "mti": "0x20",
    "cc": "0x00a8",
    "iv": "110131581702",
    "nonce": "5742335597",
    "x509":
"DFIDXTBBAkWgAwFKxgIJAJC1HiIAZAiIMA0GCSqGSIb3DfBAYTAkFVMRMwEQYDVQQ
IDApTb21lLVN0YXRlMSEwHwYDVxaWRnaXRzIFB0eSBMdGQwHhcNMTExMjMxMDg1OTQ
0WhcNMTAJjyzfN746vaInA1KxYEeI1Rx5KXY8zIdj6a7hhphpj2E04C3Fayua4DRHy
ZOLmlvQ6tIChY0Clfwue9hr97FGTR3s7fhUHwc8YuB7xxt8BVc69rLeHV15A0qyx77
CLSj3tCx2IUXVqRs5mlsgVhA=="
    }
}
```

9.3.4 SDEV AuthN (with AEAD)

```
{
```

```
  "SDEVAuthn": {
    "mti": "0x30",
    "aad": "aa-bb-cc-dd-ee-ff;12792518056",
    "authn":"
IAkgIAkNCsKFCS4Jw6oJfwkuCS4JwrQJwpwJw6EJXgkuCcK1CTYJdAlrCcK6DQouCS
MJKgkjCSYJNQkuCcKWCcOmCcKICWAJw7EJLgkuCUIJUA== "
  }
}
```

9.3.5 SDEV AuthN (without AEAD)

```
{
  "SDEVAuthn": {
    "mti": "0x30",
    "sn":"12792518056",
    "authn":"
IAkgIAkNCsKFCS4Jw6oJfwkuCS4JwrQJwpwJw6EJXgkuCcK1CTYJdAlrCcK6DQouCS
MJKgkjCSYJNQkuCcKWCcOmCcKICWAJw7EJLgkuCUIJUA==",
"sign":"4bb787275abbf88ecd548ef8b018f014c09316dcb67b7765834cd53723
3b4b3f"
  }
}
```

9.3.6 GW-INK AuthN (with AEAD)

```
{
  "GWINKAuthn": {
    "mti": "0x40",
    "aad": "aa-bb-cc-dd-ee-ff;12792518057",
    "authn":"
IAkgIAkNCsKFCS4Jw6oJfwkuCS4JwrQJwpwJw6EJXgkuCcK1CTYJdAlrCcK6DQouCS
MJKgkjCSYJNQkuCcKWCcOmCcKICWAJw7EJLgkuCUIJUA=="
     }
}
```

9.3.7 GW-INK AuthN (without AEAD)

```
{
  "GWINKAuthn": {
    "mti": "0x40",
    "sn":"12792518057",
    "authn":"
IAkgIAkNCsKFCS4Jw6oJfwkuCS4JwrQJwpwJw6EJXgkuCcK1CTYJdAlrCcK6DQouCS
MJKgkjCSYJNQkuCcKWCcOmCcKICWAJw7EJLgkuCUIJUA==",

"sign":"4bb787275abbf88ecd548ef8b018f014c09316dcb67b7765834cd53723
3b4b3f"
  }
}
```

9.4 Design of the Artificial Neural Network

9.4.1 Features extraction and normalization

We classified the interference sources through an MLP neural network with one hidden layer because this already provides good results and is computationally less expensive than networks with multiple hidden layers. As already explained in Figure 13, features in the input layer are organized in 10 neurons (8 representing the counter of 8 types of reception errors [25], one for the error burst length in μs, one representing the maximum distance between two consecutive events in the same burst in μs). In the output layer, instead, we have 4 neurons, each of them mapping the relevant interfering technology (WiFi, ZigBee, Microwave, LTE-U).

The MLP network was implemented in Python using the scikit-learn machine learning library [32], trained using back propagation with a Cross-Entropy loss function. Since MLP is sensitive to work on normalized data, i.e. on features of Gaussian distribution with zero mean and unit variance, we preprocessed our data by removing the average value and dividing the values by the feature's standard deviation. The dataset was randomly divided into two parts (using the train test split function of scikit-learn), the training set and the test set: the first one is used for training and validating the neural network, the second one is used for evaluating the classification accuracy. We considered a training set of 4716 samples (equally distributed between LTE-U, WiFi, ZigBee and Microwave oven), while the test set was composed of 2020 samples. Each sample is constituted by a vector of ten features associated to the interference that caused it. From the point of view of the library it is interpreted as a float vector, with ten features.

Finally, the hyper-parameters of the network, i.e. the number of neurons in the hidden layer, the activation function and the regularization factor have been studied in the Model Selection phase, as discussed in the following sub-section.

9.4.2 Model Selection

The model selection phase consists in comparing the performance obtained by changing different hyper-parameters, and choose accordingly the hyper-parameters that maximize the classification accuracy. To avoid the overfitting problem, we carried out a "k-fold" cross-validation with k = 10: we divided the training set into 10 equal parts and, at each step, one sub-sequence of the data set was used to evaluate the accuracy of the model trained with the remaining nine sub-sequences. We used the GridSearchCV function of scikit-learn to carry out an exhaustive "grid" search over the space of hyper-parameters considered in our analysis, and performed a k-fold cross-validation for each obtained model. Specifically, the space of hyper-parameters was configured by considering the following factors:

1. solvers: L-BFGS, adam, SGD with constant learning rate, SGD with adaptive learning rate;
2. number of neurons in the hidden layer: from 1 to 50;
3. regularization factor "alpha" (L2 penalty): 10^{-1}, 10^{-2},10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7};
4. activation function: identity, logistic, tanh, ReLU.

For solvers adam and SGD the initial learning rate was set to 10^{-3} (default value in scikit-learn), which controls the step-size in updating the weights. SGD was set with a nestorovs momentum of 0.9, while in adam the exponential decay rate for estimates of first and

second moment were set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. All solvers have tolerance tol = 10^{-4}. The solvers iterate until convergence (determined by tol) or up to a maximum number of iterations (never reached in our experiments).

Table 11: Optimization solvers with relative training times.

Solver	Accuracy	Training time	Iterations
SGD with constant learning	93.8%	100.54 s	276
SGD with adaptive learning	93.8%	98.90 s	276
L-BFGS	98.5%	7.48 s	305
Adam	96.1%	18.56 s	276

In Table 11 it is shown the average accuracy, the time required for training the weights of the optimization algorithms and the number of iterations until convergence. The optimization were run on a laptop PC with dual core 1.8 GHz CPUs and 4 GB of RAM. It is clear that L-BFGS method converges faster and with higher accuracy. Figure 25 shows that, for a given configuration of the other hyper-parameters, increasing the number of neurons in the hidden layer improves the accuracy until a limit value of about 98.5%.

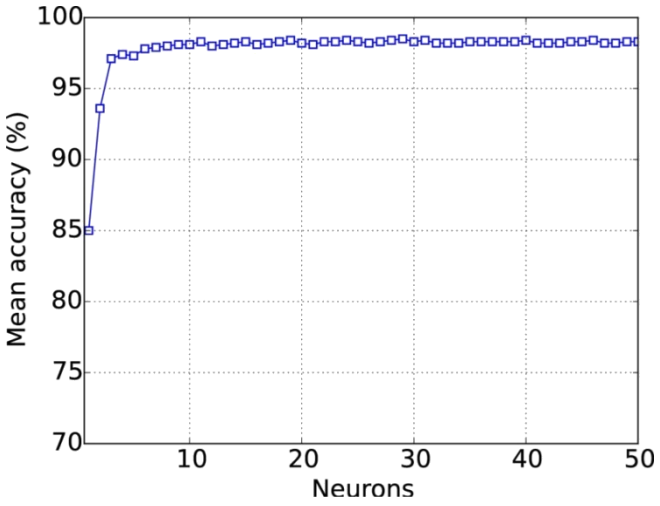


Figure 25: Average accuracy versus the number of neurons in the hidden layer.

Table 12: Average accuracy obtained by different activation functions.

Function	Accuracy
Identity	87.6%
Logistic	98.5%
ReLU	98.2%
tanh	98.1%

Table 13. Average accuracy obtained by varying the regularization factor.

Alpha:	0.1	0.01	0.001	10^{-4}	10^{-5}	10^{-6}	10^{-7}
Accuracy:	98.5%	97.9 %	97.5%	97.7%	97.9%	97.8%	97.9%

Table 12 and Table 13 show the performance achieved with different activation functions and regularization factors. In particular, the logistic function reaches a higher accuracy compared to other activation functions, while the optimal regularization factor alpha was 10^{-1}. The final hyper-parameters derived by the model selection phase result in the MLP architecture shown in Figure 13, where we omit the bias node for the sake of simplicity, with 29 neurons in the hidden layer.

9.5 LoRa simulations

To measure the impact caused by the non-orthogonal SFs, we modified and extended the LoRaSim simulator [39] to include such phenomena. LoRaSim is a Python-based discrete event simulator implemented with SimPy [45] by the authors of [36] (which we warmly thank for publishing the source code). It allows nodes to be placed in a 2-dimensional space with up to 24 LoRa gateways which act as sinks for the data retrieval (uplink communication only). The parameters that can be defined are: Transmit Power (TP), Carrier Frequency (CF), Spreading Factor (SF), Bandwidth (BW) and Coding Rate (CR). The node's behavior is described by the average packet transmission rate λ and packet payload B (with fixed preamble of 8 symbols). Finally, the gateways can receive multiple signals with different SF and BW combinations.

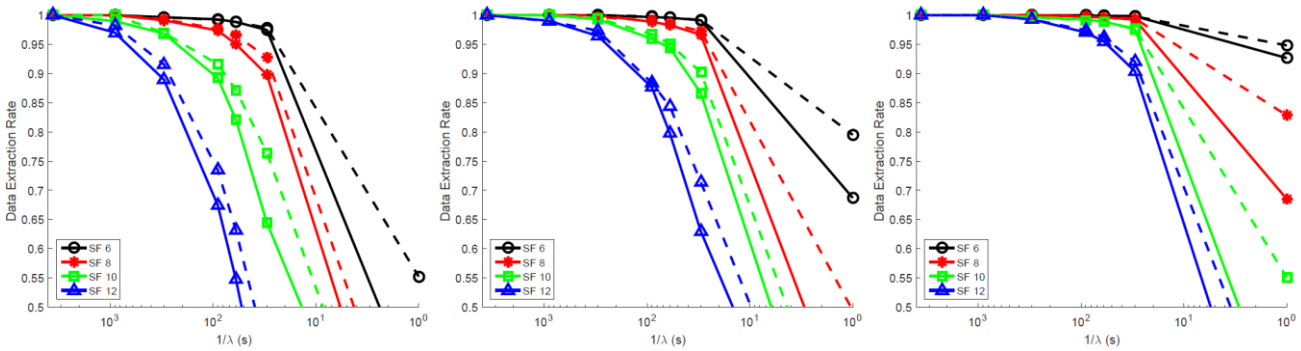
We modified LoRaSim so that if a collision occurs between packets of different SF, both packets are correctly received only when the difference in received power is below a certain threshold (in the experiments we set a conservative threshold of 6 dB for all collisions), otherwise only the packet with highest RSSI is received and the other discarded[8]. We also extended the simulator to account for log-normal fading which was not implemented yet. As we will show, fading has very limited impact on LoRa communications when packets can be received by several gateways, while it can affect the reception of the packets when there is only one gateway in visibility. Unless differently noted, the transmission parameters used by the nodes are the following: TP=14dBm, CF=868MHz, BW=500kHz, CR=4/5 and B=20Bytes.

For comparison purposes, we use the same metric used in [36] to evaluate the performance of LoRa networks, namely the Data Extraction Rate (DER), defined as the quota of packets correctly received by at least one gateway over a period of time. Indeed, in LoRa deployments all messages are received by the backend system, independently of the final destination of the packet, i.e. each transmitted message should be received by at least one base station. Thus, the achievable DER depends on the position, number and behavior of LoRa nodes and gateways, and its values vary between 0 and 1, with 1 representing an optimal LoRa deployment.

[8] We omit the case of collisions between packets of different BW for the sake of simplicity.

9.5.1 Single cell

The first set of experiments aims to assess the impact of the non-orthogonality of LoRa's SFs. To this purpose, Figure 26 shows the DER for 1000 nodes in a single isolated cell, with 1, 4 and 16 gateways respectively. The 7 SFs are uniformly assigned to the nodes (i.e. each SF is assigned to 1000/7 nodes), although for easiness of presentation in the figure we show only a subset of the SFs, namely nodes using SFs 6, 8, 10 and 12. Finally, the dashed lines correspond to the results obtained by considering the SFs as perfectly orthogonal.

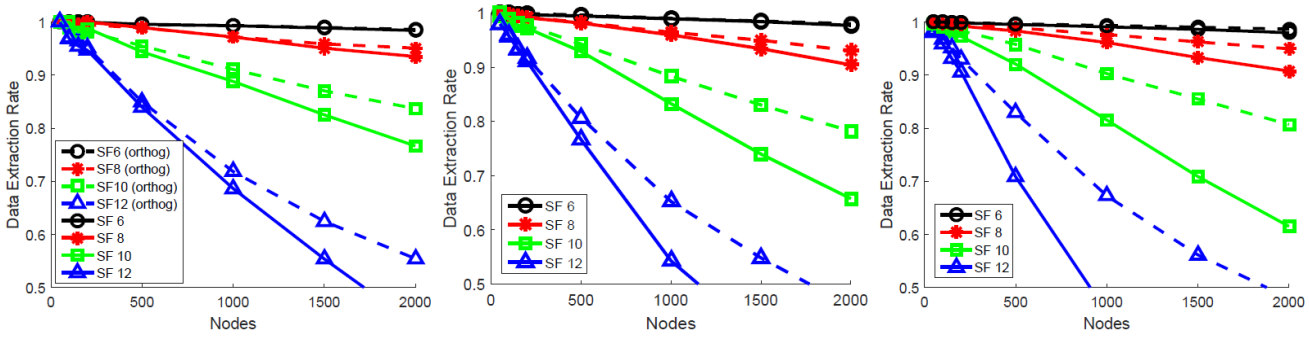


(a) 1 base station. (b) 4 base stations. (c) 16 base stations.

Figure 26: Capacity of a single LoRa cell with one or more gateways.

From the figure, it is clear that the impact of non-orthogonality is limited when the traffic rate of the nodes is low (higher values of 1/λ). However, the performance deteriorates quickly as collisions appear, especially for high SF (10-12) where packets last longer and are thus more vulnerable. As demonstrated in [36], when there are more gateways the capacity increases thanks to antenna diversity, mitigating the impact of collisions both between packets with same SF or different SFs. However, this comes at the cost of installing several base stations to cover the same area, which might not always be economically feasible.

9.5.2 Multiple cells

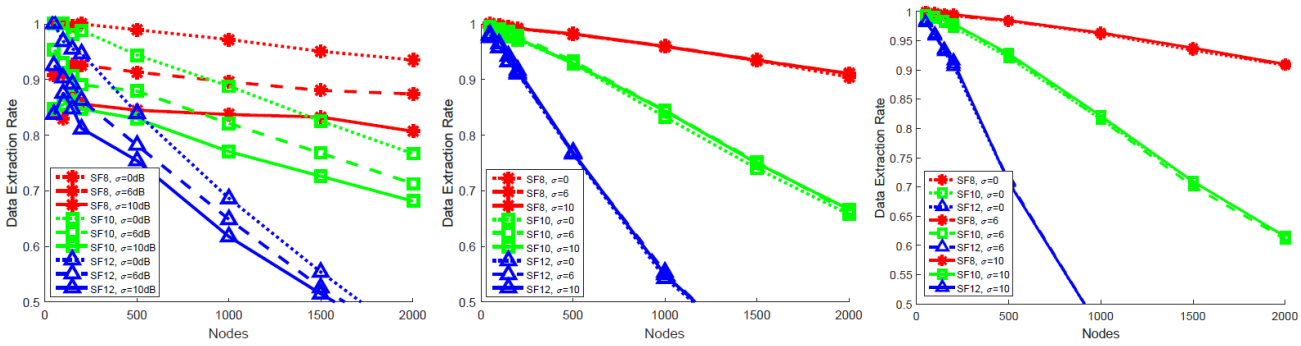


(a) 1 cell. (b) 4 cells. (c) 16 cells.

Figure 27: LoRa performance in a single isolated cell and with multiple adjacent cells (one gateway per cell).

A second set of experiments aims to assess the impact of non-orthogonal SFs in presence of multiple adjacent LoRa cells. Figure 27 shows the performance of LoRa in a single isolated cell and in a grid composed of 4 and 16 adjacent cells - one gateway per cell. In these experiments we set the nodes rate to a fixed λ= 1/90 (s^{-1}) and increase the number of nodes in each cell, up to 2000 nodes per cell. Again, the dashed lines correspond to the results obtained by considering the SFs as perfectly orthogonal, while the solid lines represent the performance taking into account inter-SF collisions. Figure 27 shows that such collisions deeply affect the performance of the highest SFs, especially when the number of adjacent cells increases. For example, with 16 cells, SF 12 experiences 30% of DER loss already with 500 nodes, while if we consider the SFs as orthogonal, the DER loss is almost halved.

9.5.3 Impact of fading



(a) 1 cell. (b) 4 cells. (c) 16 cells.

Figure 28: Performance in presence of log-normal fading for a single isolated cell and with multiple adjacent cells (one gateway per cell).

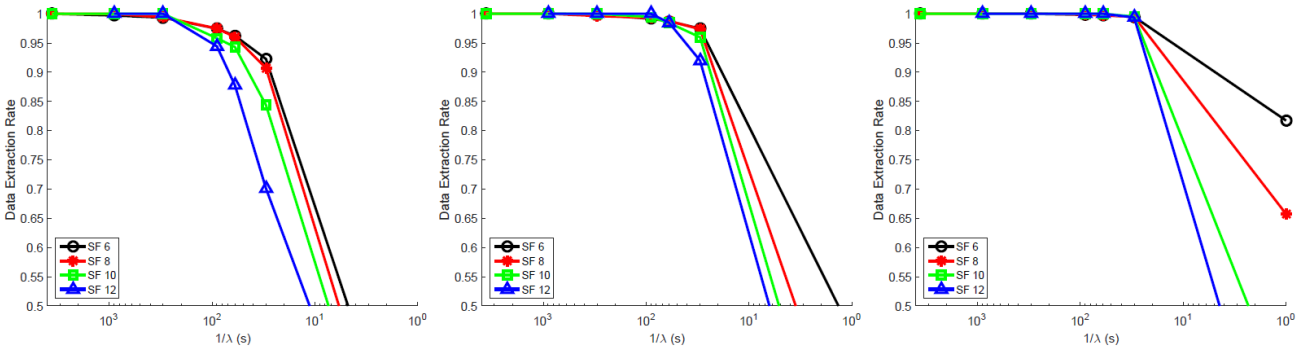
Up to now, in the simulations we considered only collisions between nodes, ignoring the impact of fading. We thus extended the LoRaSim simulator to include such phenomena, with a log-normal model of increasing σ = 0; 6; 10dB. Figure 28 shows the performance of LoRa with the joint effect of inter-SF collisions and fading, again in a single and multi-cell topology with 4 and 16 adjacent cells respectively. Interestingly, while fading has a certain impact on the single cell scenario, with multiple cells fading has virtually no impact on performance. This is due to antenna diversity which gives more chances to receive the same packet on different gateways, compensating the probability of being lost on one link because of fading.

9.5.4 Network planning hints

The above results are particularly important for network operators to correctly plan the parameters of their LoRa networks. In particular, our results suggest that increasing the SF might not always be a good solution to improve performance in presence of congestion or to compensate fading channels. Indeed, despite being more robust to noise, our results show that higher SFs are more vulnerable to inter-SF collisions and that fading becomes negligible in presence of multiple base stations. Instead, it could be more effective to use a low SF in order to lower the chances of packet collisions.

As a rule of thumb, since the length of each chirp doubles every time the SF increases, it could be sufficient to half the number of nodes in the next higher SF, so to balance the doubled time-on-air of the corresponding packets. In Figure 29 we run the same experiments of the ones in Figure 26, although distributing the SF proportionally as we just explained. The figure shows that the performance is now similar for all SFs, with slightly reduced capacity for the nodes with lower SFs (SF 6 supports now 2^7 times more nodes than SF 12) but a much higher capacity for the higher SFs.

Finally, it should be noted that in real deployments network planning is more complex because performance is affected by the transmit power of the nodes and on the distance from the gateway (respectively, uniform and random in our experiments).



(a) 1 base station. (b) 4 base stations. (c) 16 base stations.

Figure 29: Capacity of a single LoRa cell with one or more gateways, nodes distributed according to spreading factors.

9.6 LoRa Traffic Generator

The main component of our Traffic Generator is obviously the software-defined implementation of a LoRa transceiver. Indeed, this block is essential for producing elementary LoRa-modulated signals, as a function of the traffic source schedules, which are then translated on time and attenuated according to the physical channel model of each node. Details of the LoRa modulation and demodulation process have been described previously. We implemented in MATLAB the computation of the LoRa signal samples, including the symbols composing LoRa PHY preamble. No channel coding has been currently considered. The samples are then sent to the DAC of the USRP board for transmission either towards another USRP (we also implemented the corresponding LoRa demodulator in MATLAB), or towards a real LoRa device or gateway.

Our goal is to analyze the cell-level performance of a LoRa network under different network plans, i.e. under different allocations of per-node spreading factors and transmission powers. Indeed, rather than working with random spreading factors and uniform transmission powers, more robust configurations can be reserved for users experiencing bad channel qualities. We assume that all the nodes emulated by our traffic generator are configured for working on the same channel, while the number of nodes and their distribution within the cell can be configured in each experiment.

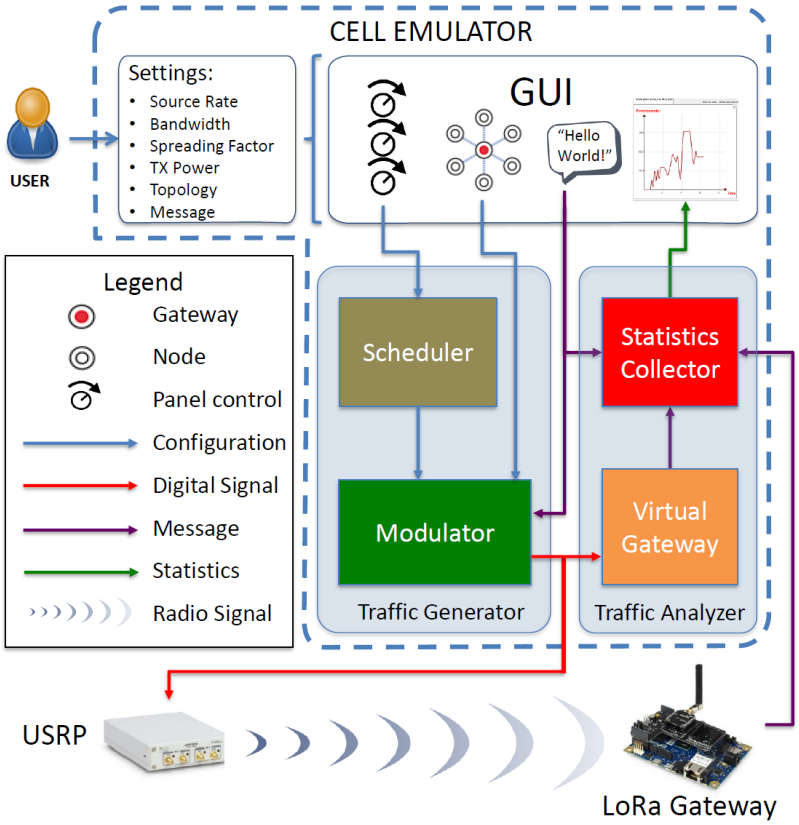


Figure 30: LoRa cell emulator architecture.

Figure 30 summarizes the software architecture of the LoRa cell emulator and the hardware used for the demo set-up: a real LoRa Gateway, implemented by using a Raspberry Pi 3 piloting the iC880A gateway transceiver, receives over-the-air the signal produced through a USRP B210, which results from the superposition of multiple LoRa-modulated signals generated by the cell nodes simultaneously active. The LoRa cell emulator is composed of three main software modules: (a) a Traffic Generator, (b) a Traffic Analyzer and (c) a Graphical User Interface (GUI). The first two modules are written in MATLAB, whereas the GUI is Python based.

The traffic generator, in turn, is composed of:

1. Scheduler, which is responsible of generating the activity intervals of each node according to the source rate and duty-cycle constraints;
2. Modulator, which receives the schedule of the nodes involved in transmission and generates the cell-level aggregated signal by summing up the LoRa modulated signals of the nodes simultaneously active, as shown in Figure 31. Before combing the signal generated by each node into the cell-level signal, a path-loss attenuation model is applied by scaling and delaying the relevant chirps.

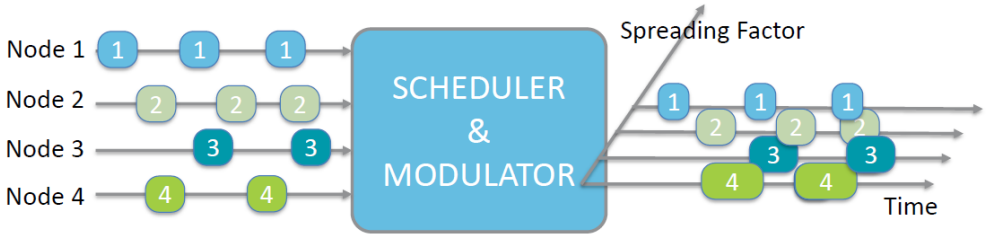


Figure 31: An example of per-node LoRa modulated signals, which are scheduled on time before being combined into the cell-level aggregated signal.

The digital signal is sent to both the USRP, for being transmitted over the air, and the Traffic Analyzer. The Traffic Analyzer is composed of a Virtual Gateway and a Statistics Collector. The former implements a LoRa demodulator working in parallel on multiple spreading factors, for identifying message preambles and then correlating each message with the corresponding basic upchirp. The latter compares the symbols demodulated in each message with the ones sent by the nodes, in order to produce low-level error statistics. Finally, the signal transmitted over the air is demodulated also by a real gateway, whose packet error statistics are sent to the statistic collector module for comparison with the results obtained by the virtual gateway.

10 Acronyms

AAD	Additional Authenticated Data
AEA	Authentication Encrypted with Additional Data
AAM	Authentication and Authorization Manager
ANN	Artificial Neural Network
BW	Bandwidth
CF	Carrier Frequency
CIA	Confidentiality Integrity and Authentication
CR	Coding Rate
CRC	Cyclic redundancy check
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSS	Chirp Spread Spectrum
DER	Data Extraction Rate
ECDH	Elliptic-Curve Diffie-Hellman
FBE	Frame based equipment
FCS	Frame Check Sequence
FFT	Fast Fourier Transform
GUI	Graphical User Interface
HKDF	Hashed Message Authentication Code-based key derivation function
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
ISM	Industrial, Scientific and Medical
JWT	JSON Web Tokens
KDF	Key Derivation Function
LAA	Licensed Assisted Access
LAAM	Local AAM
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno
LBE	Load based equipment
LPWAN	Low Power Wide Area Networks
LTE	Long Term Evolution
LTE-U	LTE-Unlicensed
MAC	Medium Access Control
MIMO	Multiple Input Multiple Output
MLP	Multi-Layer Perceptron

MSK	Master Secret Key
MTI	Message Type Indicator
OFDM	Orthogonal Frequency-Division Multiplexing
PBKDF2	Password-Based Key Derivation Function 2
PER	Packet Error Rate
PLCP	Physical Layer Convergence Procedure
PSK	pre-shared symmetric key
RAP	Resource Access Proxy
ReLU	rectified linear unit
RSSI	Received signal strength indication
S3M	SymbIoTe Smart Space Middleware
SDEV	Smart DEVice
SDR	Software Defined Radio
SF	Spreading Factor
SGD	Stochastic Gradient Descent
SSP	Smart SPace
TDMA	Time Division Multiple Access
TLS	Transport Layer Security
TP	Transmit Power
3GPP	The 3rd Generation Partnership Project