WInnComm-Europe 2017

2017 Wireless Innovation Forum European Conference on Communications Technologies and Software Defined Radio

WIRELESS INNOVATION FORUM™

17-18 May 2017

held in conjunction with ICMCIS, hosted by the University of Oulu, Finland

Proceedings of
WInnComm Europe 2017
Wireless Innovation European Conference on
Wireless Communications Technologies and Software Defined Radio

17-18 May 2017, University of Oulu

Editors: Marc Adrat, Lee Pucker, Stephanie Hamill

Thank you to our sponsors:

Google

LEONARDO

MOTOROLA SOLUTIONS

Media Sponsor:

THALES

Table of Contents

Papers

Software Defined Radio applied to Mission Oriented Sensors Array - A proposal to advanced Embedded Systems architecture

Nina Machado Figueira - Captain - Ph.D. Computer Science
Victor Bramigk - First Lieutenant - B.Sc. Computer Engineering
David Fernandes Cruz Moura - Major - Ph.D. Defense Engineering

May 2017

Abstract

The variety of Embedded Systems (manned and unmanned), sensors and waveforms available for communication, require the development of an architecture that facilitates the portability of sensorial processing applications (SPA). It is also desirable that this architecture enables the adaptability of the waveform and SPA according to the operating context. Thus, considering an embedded system with a Mission Oriented Sensors Array (MOSA) and transmitting data through an SDR, this will enable an architecture that integrates the SPA with a control system of the SDR. One of the advantages of this standardization remains in the development and portability of adaptive systems, modifying the waveform and SPA according to the operating conditions. The data channel features and the requested Quality of Service (QoS) are examples of operational conditions in the context of this work. One way to achieve this standardization is to use the SCA architecture in conjunction with the structure of a MOSA. Thus, an architecture integrating a MOSA with an SDR will be presented, both in an SCA environment, that promotes greater interoperability and portability of hardware and software.

Keywords: *SDR, SCA, MOSA, Embedded Systems,* C^4ISTAR

1 Introduction

In the past the simplicity of wars allowed victories to be won by the action of only one Armed Forces. Succeeding in missions was more related to the leadership of the chief, to the difference of personnel, to the use of mass and personal bravery than to the judicious coordination of elements of nature and of different organizations. The latest wars and conflicts have revealed that great victories have been achieved through integrated naval, land and air forces. Current conflicts tend to be limited, undeclared, of unpredictable duration, with fluid and diffuse threats. This context requires military forces capable of working together, with flexibility, versatility and mobility. [1] [5]

The planning of a Joint Operation differs from the Singular Operations by the heterogeneity of the employment processes and by the technical-professional peculiarities of the Component Forces. The importance of coordinating and integrating command, control, communication, computing and intelligence structures in order to facilitate surveillance and reconnaissance operations, C^4ISTAR Systems, as well as logistical operations is directly related to success in the operations of the knowledge age [4].

The network-centric warfare (NCW) elevated situational awareness to a level unimaginable ten years ago, for both the individual combatant and the chain of command. It streamlined decision-making, ensuring mission accomplishment, and controlling its execution over time. This made it possible to adapt them in real time to the need for combat. In this interoperability scenario, the need arose for integration of existing Decision Support Systems (C^4ISTAR).

Therefore, it became necessary to develop communication systems that interacted with heterogeneous systems without major changes in hardware. In this scenario emerges Software Defined Radio, capable of using various waveforms and cryptographic algorithms, defined through the software, according to the operational scenarios. This has been possible due to advances in several areas such as: analog-to-digital conversion, antennas, digital transmission, digital signal processing, software architecture and device processing capability such as General Purpose processor (GPP) [2].

The versatility of RDS allows embedded systems to operate in a variety of conditions and environments. Unmanned Systems are composed by a lot of embedded systems such as: communication, control, navigation, positioning, sensors payload. The main task of civil and military applications in Unmanned systems is the acquisition of data. However, due to the low data rate and the time requirements of the real time applications, on-board pre-processing is required before its transmission. The MOSA (Mission Oriented Sensor Arrays) [6] is a type of hardware and software architecture for Autonomous Systems that meets these requirements.

To allow the reuse of components (e. g. middleware), to avoid unnecessary processing between the two systems and to enable the development of adaptive applications, it is necessary to integrate the two architectures. With this, one can develop an application that, according to channel conditions, modifies the pre-processing of data.

This paper aims to present a proposal for integrating the RDS architecture into the MOSA architecture for the production and processing of real time data, composing an advanced architecture for Embedded Systems, raising the situational awareness of the decision maker and increasing the power of combat. To validate the concepts presented in this work a case study is planned to be carried out next November in the Amazon region.

The Section 2 presents the SDR Program of the Brazilian Ministry of Defense. Section 3 present the MOSA Architecture. Section 4 integrate the MOSA within a SDR-SCA compliance in an interoperability proposal. Finally, section 5 presents the concluding remarks and future works.

2 Software Defined Radio Defense Program

Software Defined Radios (SDR) technology represents the state of the art in military telecommunications worldwide. In this context, the Ministry of Defense has assigned the Army Technological Center (CTEx) of **(to** initiate the National Software Defined Radio Program of the Ministry of Defense (SDR Defense), whose objective is to carry out the development of radio equipment prototypes based on SDR technology to supply the needs of tactical communications as well as the Command and Control System of the Armed Forces[2].

2.1 NIPCAD

Since SDR is a complex technology, a need to subdivide the program into intermediate projects was identified identified the need to implement a program, composed of intermediate projects. At CTEx, the Nucleus of Innovation and Research in Communications Applied to Defense (NIPCAD) was created to carry out research and development in the area, as well as to manage technical development, the human and financial resources of the Armed Forces and of the development agencies allocated to that program.

Faced with the great scientific and technological challenge of mastering the entire R &D process of military equipment based on SDR technology, as well as initiating research in the area of Cognitive Radios to meet the tactical communications demands of the Armed Forces, NIPCAD/CTEx has been establishing partnerships with universities, teaching and research institutions and companies. The main research institutions that integrate the Program are the Naval Systems Analysis Center and the Navy Research Institute. Among the foundations, the Center for Research and Development in Telecommunications (CPqD) stands out.

2.2 SDR Project

The SDR Project is multidisciplinary and should contribute to the interoperability in the tactical communications of the Armed Forces, as well as to act in the cyberspace with freedom of action and security of communications.

The objectives of the project will be achieved through the development of radio prototypes, based on the SDR concept, capable of providing communication protocols adherent to the doctrine of the Brazilian Armed Forces, to the employment scenarios specific to the performance of these Armed Forces, as well as offering efficiency, availability and security in communications, both in terms of Electronic and Cybernetics Warfare.

In addition, the SDR Project has the following objectives: training of highly qualified human resources, knowledge domain of a strategic area for Brazil, promotion of the Defense Industrial Base, especially those related to the telecommunications sector, strengthening institutional links between civil and military research institutes, as well as creating conditions to promote research and developments in the area of Cognitive Radios.

The SDR-Defense Program comprises two development cycles. The first one aims to develop the prototype of vehicular radios that can be shipped in naval and terrestrial vectors. The second involves the development of prototypes of smaller and lighter radios, called handheld and manpack. The first development cycle is expected to last 10 (ten) years and is composed of 4 (four) phases.

During this period, prototypes of vehicular RDS are being developed operating in the HF, VHF and UHF bands. Analog and digital waveforms to operate on all these spectral bands, which will be integrated with various cyber-security mechanisms, Transmission security (TRANSEC) and communications security (COMSEC). Also being used the SCA waveform development environment kit to facilitate the development of new waveforms. This project uses the incremental development, in which new functionalities will be added to the prototypes of a certain phase to generate prototypes of the subsequent phase. Implementation of the first development cycle started in December 2012, and the scopes and predictions for the completion dates of the four phases of the first R &D cycle are as follows:

- First phase (SDR-vehicular operating in the VHF range): December 2018;
- Second phase (SDR-vehicular operating in the HF and VHF bands): December 2019;
- Third phase (SDR-vehicular operating in the HF, VHF and UHF bands): December 2020;
- Fourth phase (updating the HF, VHF and UHF waveforms, as well as completion of the waveform development platform): December 2022.

Figure 1 shows the Roadmap for the RDS-Defense Program.

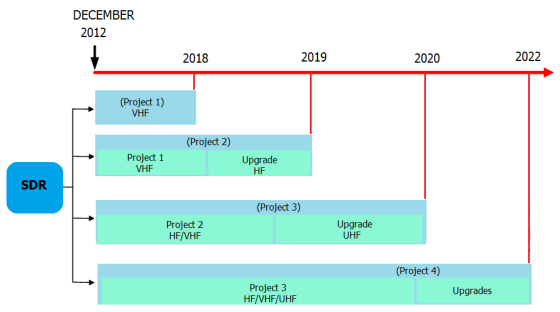


Figure 1: Roadmap SDR Program

The design of the first phase was divided into modules, one for management, one for integration and the others for the development of specific parts of the prototypes, such as: waveforms, security, front end and SCA core.

Figure 2 shows the front views of the two types of prototype intended to be developed in the first development cycle of the RDS-Defense Program (RDS-Defenses Vehicle Versions). Figure 3 shows the overview of the prototype

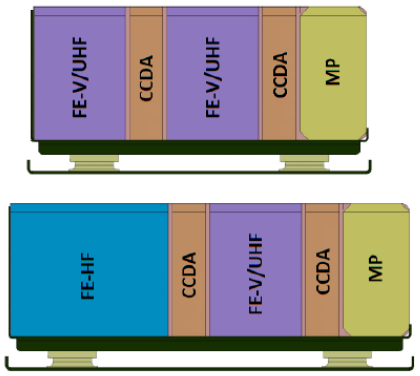


Figure 2: SDR Modules

These prototypes are composed of:

- MP - Processing Module, where the signal processing in base-band is performed;
- FE-V/UHF - Front end operating in the VHF and UHF bands, which are responsible for generating the electromagnetic waves to be irradiated by the antennas and to perform analogue filtering;
- MSCA - SCA Middleware Module;
- MFOSCA - SCA Waveform (VHF) Module;
- CCDA - Digital-Analog Control and Conversion, an integral part of the Radio Frequency Module;
- MSEG - Security Module;
- FDSCAC - SCA Compliance Development Tool;
- MPLAN - Communications Mission Planner.

Figure 3 shows the integration of the prototype with the Brazilian Army Decision Support System "C2 in combat" in laboratory and and in operational vehicle.

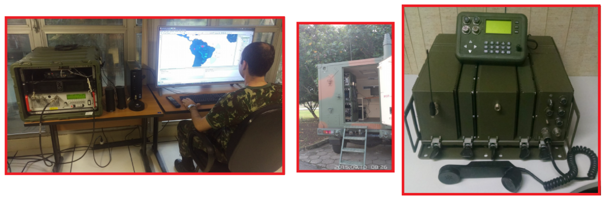


Figure 3: Integration Test

3 Mission Oriented Sensors Array

The MOSA (Mission Oriented Sensor Arrays) architecture discussed in this article has the potential to provide real-time, ready-to-use information made in embedded data processing machines embarked on Unmanned Vehicle Systems (UVS), reducing the need to send video streams or high resolution images. As a consequence, the minimal communication bandwidth to carry on data in real time reduces significantly.

The main characteristic of MOSA architecture is the division of the system into two distinct modules: the vehicle module (the safety-critical UVS element) and the MOSA module (the non-critical UVS part). MOSA systems include a set of embedded sensors that provide raw data for specific applications.

In addition to hardware, a MOSA system also includes the software required to perform a mission, to communicate with all sensors, and send / receive data to / from the vehicle. Integrated processing reduces the complexity of raw data in ready-to-use information [7].

Figure 4 shows a simplified functional diagram of the MOSA architecture and the interconnection between its system components. The topology of the system may change according to the application.

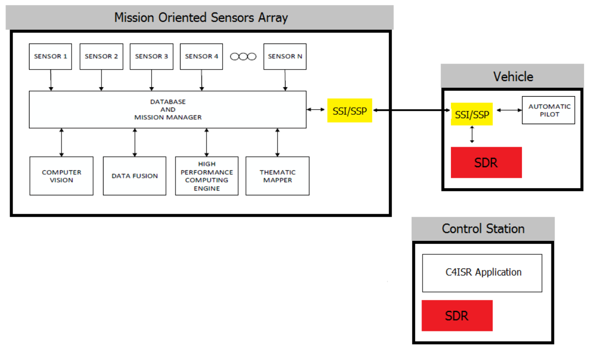


Figure 4: Basic Architecture of MOSA Systems

Model-Driven Development (MDD) is used in the construction of MOSA systems. MDD is a software development methodology where the key elements are models, from which the code is produced. Using MDD, rapid prototyping of complex systems can be achieved through the automatic generation of high-performance code. This code can be embedded in electronic components to be applied in real-time environments.

In order to communicate with the aircraft, MOSA uses a standard interface, called SSP / SSI (Smart Sensor Protocol / Smart Sensor Interface). SSP is the communication protocol, while SSI is the interface that allows the MOSA system to use various services provided by the stand-alone vehicle, particularly the data transmission with the remote ground control station (GCS). MOSA systems can be used in different UVS that have been adapted to communicate through SSI / SSP.

The communication protocol uses a plug-and-play mechanism to verify that the vehicle is capable of performing a specific mission. This possibility is negotiated between the MOSA and the vehicle during the handshake phase of the protocol. In some cases, longer or better vehicle stability may be necessary, among other limiting factors. According to these limitations a MOSA systems will, in whole or in part, execute a planned mission.

The MOSA approach leads to modern autonomous systems that can perform complex missions, presenting decision-making capabilities and optimizing data flow, in real time, within the limits of communication channels.

Although in complex systems, such as medium and large UVS, hardware costs does not present a limitation, the use of MOSA can provide great versatility and flexibility in the process of developing sensor systems for new applications. Different sensors and processing units can be integrated into the best cost / benefit arrangement for a specific application. It is designed to automatically perform missions that can be pre-programmed at the GCS. In addition, missions can be reconfigured in case of events that may compromise mission results or degrade flight safety, such as an unexpected change in atmospheric condition. MOSA can dynamically choose the best sensor arrangement for a given atmospheric condition, mitigating the impact on mission results.

Those basic capabilities, of checking the system capabilities and optimizing the embedded system topology (e.g. number and type of sensors), requires that the system to know the sensors attached to itself and to describe it capabilities. For that end was developed the Smart Sensor Protocol (SSP) and the Smart Sensor Interface (SSI).

The project of the SSP was focused in the simplicity and, once that it is just a mechanism for interfacing, it provides a simple layer and that should not cause overhead in the communication between the entities. It provides connection, verify the viability of the execution of the mission, enables the communication between the entities during the mission execution and ends this cooperation section. It does not interfere in the data been exchanged, so that it should be easy to adapt this protocol for other architectures of autonomous vehicles [3].

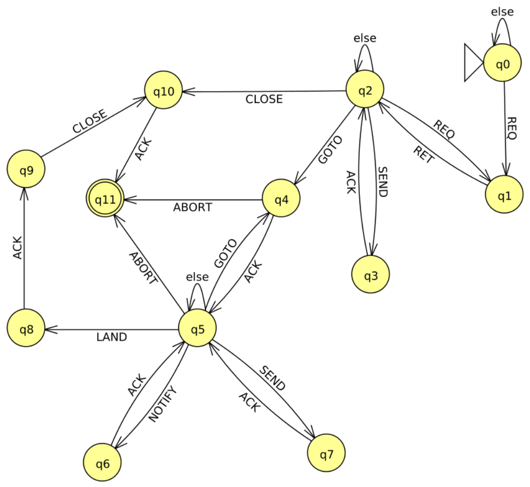


Figure 5: State Machine of the SSP [3]

The protocol it self follows the stages of the mission execution. A state machine of the mission execution with the headers of the messages in each stage can be seen in figure 5. The mission stages and respective states are as follows:

1. Negotiation and Mission viability: q0, q1, q2 and q3;
2. Mission Execution: q4, q5, q6 and q7;
3. Mission Ending: q8, q9, q10 and q11;

Although it can be argued that this structure limits the execution of the mission, this enables the simulation of the mission and assure the behavior of the system in various failures scenarios. For a more detailed description of see [3].

4 MOSA & SDR - Interoperability Proposal

Due-to the integrated characteristics required in the NCW presented in section 1, unmanned-systems must provide real-time data to the C^4ISTAR systems.

As a consequence, due to the amount of data, delays in network and scarce channel bandwidth, it is necessary to control the pre-processing step, waveform and data flow to achieve this real-time requirement. This requires MOSA to change the pre-processing step according to the available waveforms.

The objective is to enable MOSA to consult the waveform capacities and check if they are compatible with the mission. For this end, it is necessary to expand the MDF to add waveform requirements. One approach is to consider the following parameters for tactical operations [2] [8]:

- Number of nodes in the network
- Antenna characteristics
- Velocity of the UVS
- Maximum reach
- Frequency of operation
- Throughput
- Delay
- Jitter
- Bit Error Rate (BER)
- Package Error Rate

This minimal requirements will be generated in the planning of the mission.

All this information must be consulted in the SCA environment so that the MOSA control component can check if the mission requirements are meet. This would be done in the first step depicted in of figure 5 in states $q1$ to $q3$.

For that end, an SCA application will be created to enable MOSA communication with SDR-SCA Applications. As presented in section 3, the MOSA communication with the UVS is performed through UDP or TCP/IP, depending on the system requirements. So, it is visible to the SDR SCA components. If that is not the case for some legacy MOSA system, an RS232 communication can be established and a SCA Device created to enable the integration. This case can be seen in figure 6.

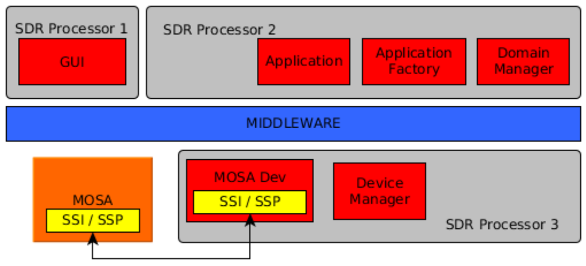


Figure 6: Basic SCA Architecture for Legacy MOSA

If the MOSA system has a network interface, them the hole SCA system can be implemented under the SCA mindset. All sensors can be SCA Devices and the SPA as a SCA Application. This case is presented in figure 7. That would provide more integration and would enable both systems to use computing power for both SPA and WF processing.

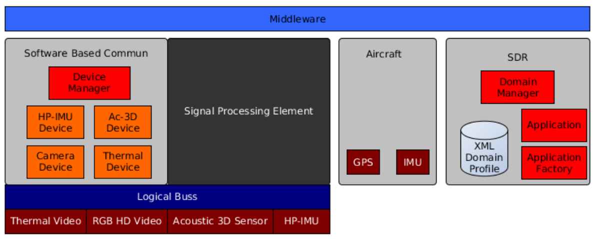


Figure 7: Basic SCA Architecture MOSA

In both cases the operation sequence would proceed as follows: MOSA System check the available WF through **MOSA Device**. It then determines whether there is some that meet the minimum requirements. After, appropriated requests would be done to instantiate and start the respective WF and to configure the SPA accordingly. After that, MOSA would continue if the mission is partially or fully feasible.

That could be useful in a mission such that if the communication bandwidth has more than $1200Kbps$, it would stream pre-processed video, like a smart-frame. Otherwise, if there is at least $2.4Kbps$, then the mission would be partially feasible and would only send important triggers after the SPA data is pre-processed and analyzed.

This represent a major step towards improving the usability of both systems by enabling the update of systems main characteristics through software updates. Also, this improves the resources usage, reducing energy consumption by optimized processing and weight reduction.

5 Concluding Remarks and Future Works

The objective of this work was to present a proposal of integration of the SDR architecture with the MOSA architecture for the production and processing of data in real time. It focused on presenting the main aspects of the system. The flexibility and modularity of an SDR embedded in an autonomous vehicle, coupled with the plasticity of the SCA compliance, can optimize communication by enabling the selection, portability and development of new mission-oriented waveforms.

It is envisaged that the proposed model greatly facilitates the selection and adjustment on-the-fly of the mission waveform using techniques such as Spectrum Sensing and Dynamic Spectrum Access (SS and DSA). It is intended to finalize simulations and field tests with the proposed architecture at the end of this year, allowing the validation of functional and performance aspects.

The preliminary analysis presented indicates that MOSA and SDR-SCA architectures are compatible, so that its integration can represent a major step towards improving the usability and enhancement of autonomous systems in civil and military applications.

6 Acknowledgement

The authors acknowledge the support granted by Studies and Projects Financier (FINEP), Brazilian Army (EB), Army Technological Center (CTEX) and members of the Nucleus of Research and Innova-

tion in Telecommunications Applied to Defense (NIP-CAD).

References

[1] David S. Alberts and Richard E. Hayes. *Power to the Edge: Command and Control in the Information Age.* CCRP, 1942.

[2] Marcio Nascimento Bispo, George Alex Fernandes Gome, José Niuton da Nova, and David Fernandes Cruz Moura. Emprego de análise baseada em cenários em apoio a projeto de sistemas de comunicações militares. *Cadernos CPqD Tecnologia*, 10(2):63–76, 11 2014.

[3] Rayner de Melo Pires. Desenvolvimento de um mecanismo plug-and-play para o arranjo inteligente de sensores em sistemas aéreos não tripulados. Master's thesis, USP - São Carlos, 12 2013.

[4] EB. *Doutrina de Operações Conjuntas.* Minstério da Defesa, 2011.

[5] EB. *Manual de Campanha de Comando E Controle.* Minstério da Defesa, 2015.

[6] N. M. Figueira, I. L. Freire, O. Trindade, and E Simões. Mission-oriented arrays and uavs - a case study on environmental monitoring. *Commission VI, WG VI/4*, 10, 2014.

[7] Nina Machado Figueira. *Arranjos de sensores orientados à missão para a geração automática de mapas temáticos em VANTs.* PhD thesis, USP - São Carlos, 02 2016.

[8] Ronaldo M. Salles, David F. C. Moura, Jeronymo M. A. Carvalho, and Marcelo R. Silva. Novas perspectivas tecnológicas para o emprego das comunicações no exército brasileiro. *RMCT*, pages 69–80, 2008.

Can SCA 4.1 Replace STRS in Space Applications?

Ran Cheng[1], Li Zhou[1*], Qi Tang[1], Dongtang Ma[1], Haitao Zhao[1], Shan Wang[1], Jibo Wei[1]
[1]College of Electronic Science and Engineering, National University of Defense Technology, China
[*]Email: zhouli2035@nudt.edu.cn

***Abstract*—Software Defined Radio (SDR) has gained a great deal of attentions in both civil and military applications. As the most important SDR standard, Software Communications Architecture (SCA) has been accepted and extensively adopted in the development of radio systems since it was first released. A new standard, Space Telecommunications Radio System (STRS), was developed by National Aeronautics and Space Administration (NASA) to align with the mission requirements due to the drawbacks of SCA 2.2.2. In this paper, we explore whether SCA 4.1, the latest release version of SCA, can meet the requirements of space application by making a comparison between SCA 4.1 and STRS in static memory occupation, inter-components communication delay, waveform deployment delay and waveform switching delay. We have tested our realizations of full SCA 4.1, lightweight SCA 4.1 and STRS in our testbed extensively, where full SCA 4.1 is considered as a benchmark. The experiment results show that although lightweight SCA 4.1 has relative larger demand on static memory occupation, longer waveform loading and switching delay, its inter-components communication could be as efficient as STRS.**

Keywords—Software Defined Radio, Software Communications Architecture, Space Telecommunications Radio System, efficiency

I. Introduction

The concept of Software Defined Radio (SDR) is adopted in implementing communication systems in the space environment with three main goals, reducing the development cycle-time, reducing the development cost, and increasing the communication flexibility among space radios and ground stations. However, implementing SDR in space environment comes with some challenges that need special concerns. Firstly, the space communication system is a strict size, weight and power (SWaP) constrained system, which limits its total capability. Secondly, the chips for space use is developing much slower than chips for commercial use. For example, the central processing unit (CPU) and memory chips for space use are quite expensive and have much lower clock speeds, smaller cache and memory size due to the strict reliability requirements in the space environment. Finally, there are urgent demands for the reuse of both the platform and software. The platform reuse allows different applications to be deployed on the same platform, enabling the innovation of the system. The software reuse reduces the cycle-time of reprogramming and speeds up the date-to-market [1][2].

Software has been contributed to the development of communication systems for decades, which brings a lot of advantages compared with hardware-based or firmware-based system development. Among the advantages, modularity, lower development costs and shorter development time are the best benefits and highlights. Based on the software-based development paradigm used in SDR-based system, some hardware problems could be turned into software problems [3].

Quite a few SDR standards have been proposed since the concept of SDR was coined, e.g., Software Communications Architecture (SCA) [4], GNU Radio [5], Abstraction Layer and Operating Environment (ALOE) [6] and Space Telecommunications Radio System (STRS) [7] etc. SCA is the most widely adopted standard in SDR area and has been deployed in thousands of radios worldwide [8], while STRS is dedicated for space SDR development and has been deployed on the space communications and navigation (SCAN) testbed of NASA [9].This paper is designed to verify if appropriate realization of the latest version of SCA can catch up with the efficiency indexes of STRS[11]. Thus, we first give a brief introduction of highlights of SCA 2.2.2, STRS and SCA 4.1 in a chronological sequence.

A. SCA 2.2.2

The final specification of SCA 2.2.2 was proposed in May, 2006. It describes the core framework which provides the infrastructure to allow software components to plug-and-play and also describes set of rules and application program interfaces (APIs) that must be used by developers of SCA-compliant software components. Fig. 1 presents the SCA software execution model. As depicted in the figure, the hardware is separated from the software through the hardware abstraction layer (HAL), making the operation environment (OE) independent of the hardware.

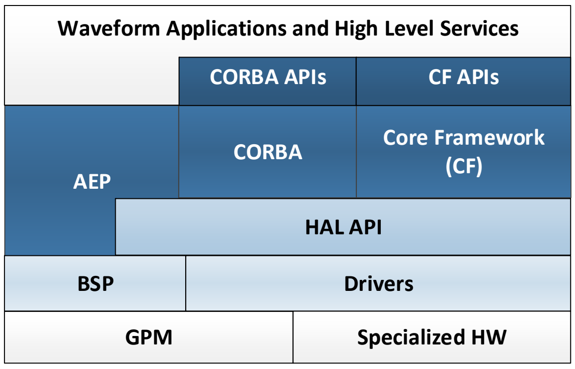


Fig. 1. SCA 2.2.2 software execution model [11].

Due to the stringent portability and interoperability requirements, SCA radios generally contain much more software than previous generations of SDRs. To be more specific, SCA radios use some software modules to process the input and output signals. Most SCA radios contain several millions lines of source code. As a result, SCA radios generally take longer time to boot, use more memory and has lower communication speed [8].

In order to support the ability of multi-platform communication, common object request broker architecture (CORBA) standard is adopted. It brings great benefits to support portability and interoperability. However, its high demands on memory consumption and CPU occupation make it an obstacle of adopting SCA in space applications.

B. STRS

The most recent specification of STRS, NASA-STD-4009, was approved in May, 2014. In order to benefit from the SDR technology in space applications, NASA proposed STRS at a historic moment [12]. Fig. 2 presents the STRS software execution model. As can be seen from the figure, the waveform applications and high level services are abstracted from the hardware by a common, consistent operating environment which includes a set of published APIs, including POSIX APIs, STRS APIs and HAL APIs [12].

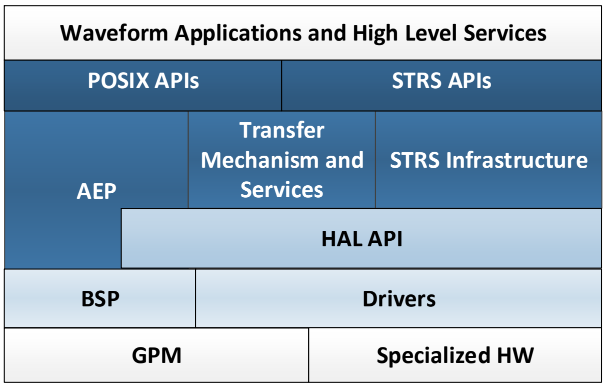

Waveform Applications and High Level Services
POSIX APIs
STRS APIs
AEP
Transfer Mechanism and Services
STRS Infrastructure
HAL API
BSP
Drivers
GPM
Specialized HW

Fig. 2. STRS software execution model [12].

STRS is designed to be able to evolve for each generation of space communication. The standard provides many flexibility to meet a variety of requirements of users and make it possible to reuse components in previous versions, which reduces the cost and the risk of deploying SDRs for space missions.

C. SCA 4.1

The Joint Program Executive Office (JPEO) for the Joint Tactical Radio System (JTRS) put forward SCA 4.0 in 2009. With multiple recommendations incorporated, it is replaced by the latest version, SCA 4.1, which was announced in August, 2015. Thanks to the great contributions from JTRS, JTNC, WInn Wireless and many other companies, the SCA keeps advancing, making it potentially applicable to resource-limited scenarios.

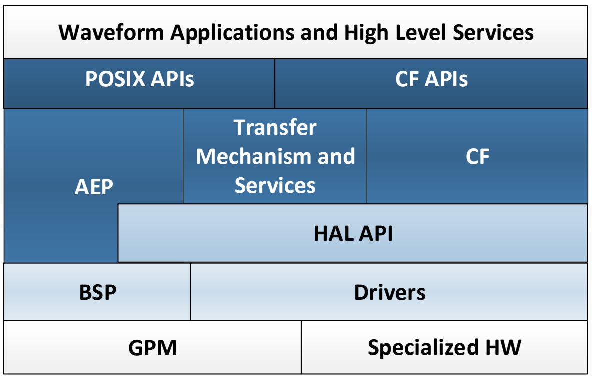

Waveform Applications and High Level Services
POSIX APIs
CF APIs
AEP
Transfer Mechanism and Services
CF
HAL API
BSP
Drivers
GPM
Specialized HW

Fig. 3. SCA 4.1 software execution model [4].

The highlights of evolution from SCA 4.1 to SCA 2.2.2 can be summarized as follows.

Adopt "push model" behavior. The "push model" behavior reduces the total number of calls, which lowers the overall complexity and enhances the system efficiency.

Remove dependence on CORBA. SCA 4.1 has removed the CORBA requirements and replace it by a reconfigurable transfer mechanism with common APIs, so that the architecture can gain more flexibility and efficiency.

Add static registration behavior. SCA 4.1 provides both static registration and dynamic registration. The introduce of static registration can enhance the system performance and reduce overhead.

Provide Units of Functionality (UOF) and SCA Profiles. UOFs are able to omit unnecessary interfaces and modules, so that the object size can be reduced and the system performance can be improved.

Overall, SCA 4.1 has introduced many lightweight designs which enhance the flexibility and efficiency of the architecture. With these new features, it is supposed that SCA 4.1 has the potential to be deployed on space SDRs. Therefore, whether STRS is the only option to be deployed on the radio device in space environment and whether SCA 4.1 can be utilized in space environment still need to be explored.

To make these doubts clear, this paper makes a comparison between SCA 4.1 and STRS by different metrics, including static memory occupation, inter-components communication delay, waveform deployment delay and waveform switching delay.

The reminder of this paper is organized as follows. Section II presents our core framework designs for SCA 4.1 and STRS. Section III introduces our testbed implementation. Section IV presents the experiment results and section V concludes the paper.

II. CORE FRAMEWORK DESIGN

Fig. 4 and Fig. 5 illustrate the layered framework that we designed according to the standard of SCA 4.1 and STRS. In STRS, core framework is also known as STRS infrastructure.

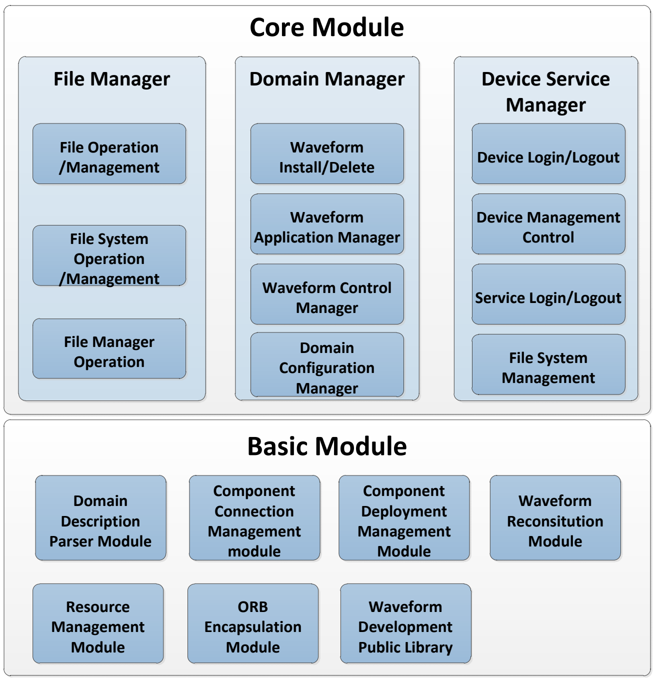

Core Module
File Manager
File Operation /Management
File System Operation /Management
File Manager Operation
Domain Manager
Waveform Install/Delete
Waveform Application Manager
Waveform Control Manager
Domain Configuration Manager
Device Service Manager
Device Login/Logout
Device Management Control
Service Login/Logout
File System Management
Basic Module
Domain Description Parser Module
Component Connection Management module
Component Deployment Management Module
Waveform Reconsitution Module
Resource Management Module
ORB Encapsulation Module
Waveform Development Public Library

Fig. 4. SCA 4.1 layered framework.

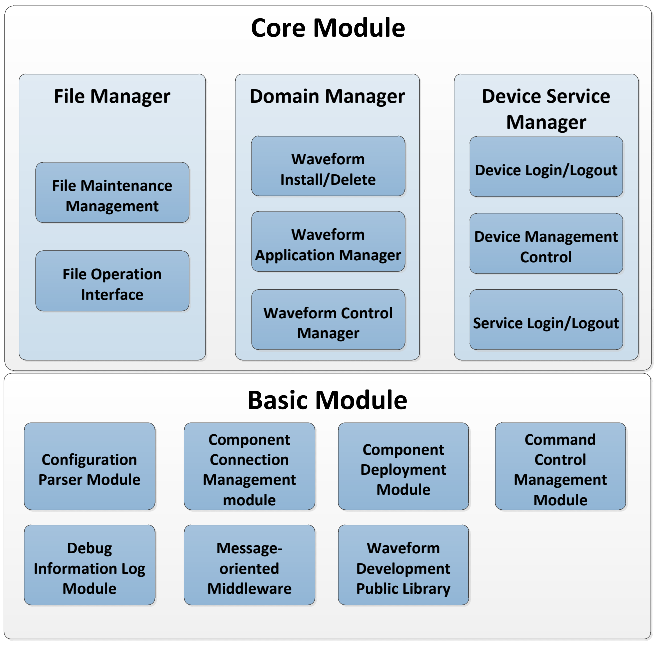

Core Module
File Manager
File Maintenance Management
File Operation Interface
Domain Manager
Waveform Install/Delete
Waveform Application Manager
Waveform Control Manager
Device Service Manager
Device Login/Logout
Device Management Control
Service Login/Logout
Basic Module
Configuration Parser Module
Component Connection Management module
Component Deployment Module
Command Control Management Module
Debug Information Log Module
Message-oriented Middleware
Waveform Development Public Library

Fig. 5. STRS layered framework.

As can be seen from Fig. 4 and Fig. 5, STRS does not include the Domain Configuration Manager, File System Management and ORB encapsulation module. It is obvious that STRS has much less requirements than SCA, so that STRS owns a more lightweight core framework structure.

Fig. 6 and Fig. 7 presents the core framework interfaces of SCA 4.1 and STRS respectively. STRS core framework is written in C language and does not have the complex inheriting relationship as SCA, but only has some common infrastructure interfaces, which simplifies the realization of the core framework.

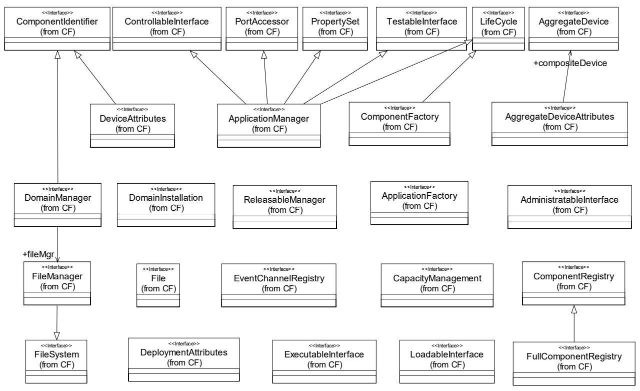

<<Interface>> ComponentIdentifier (from CF)
<<Interface>> ControllableInterface (from CF)
<<Interface>> PortAccessor (from CF)
<<Interface>> PropertySet (from CF)
<<Interface>> TestableInterface (from CF)
<<Interface>> LifeCycle (from CF)
<<Interface>> AggregateDevice (from CF)
+compositeDevice
<<Interface>> DeviceAttributes (from CF)
<<Interface>> ApplicationManager (from CF)
<<Interface>> ComponentFactory (from CF)
<<Interface>> AggregateDeviceAttributes (from CF)
<<Interface>> DomainManager (from CF)
<<Interface>> DomainInstallation (from CF)
<<Interface>> ReleasableManager (from CF)
<<Interface>> ApplicationFactory (from CF)
<<Interface>> AdministratableInterface (from CF)
+fileMgr
<<Interface>> FileManager (from CF)
<<Interface>> File (from CF)
<<Interface>> EventChannelRegistry (from CF)
<<Interface>> CapacityManagement (from CF)
<<Interface>> ComponentRegistry (from CF)
<<Interface>> FileSystem (from CF)
<<Interface>> DeploymentAttributes (from CF)
<<Interface>> ExecutableInterface (from CF)
<<Interface>> LoadableInterface (from CF)
<<Interface>> FullComponentRegistry (from CF)

Fig. 6. SCA 4.1 core framework interfaces.

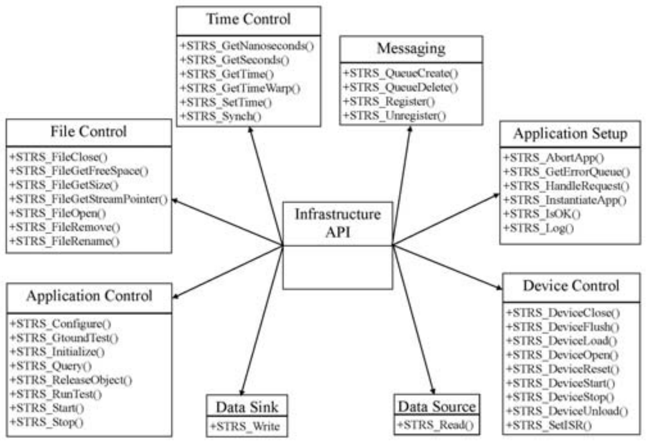


Fig. 7. STRS core framework interfaces.

The purpose of this paper is to compare the efficiency of SCA 4.1 and STRS. The comparison is carried out with respect to full SCA 4.1, lightweight SCA 4.1 and STRS. Lightweight SCA 4.1 is a profile with the lightest selection of interfaces, modules, which is considered as a potential replacement of STRS. Full SCA 4.1 is an realization with full CORBA Profile transfer mechanism and we consider it as a benchmark in our experiments [4][12].

Full SCA 4.1 still adopt the concept of agency, making users easier to access services provided by the component in the system. The agency mechanism is depicted in Fig. 8. The Remote Method Invocation (RMI) system establishes three abstract layers, which contain socket communication, the serialization and deserialization of variables and results, etc. The stub and skeleton combine to form the RMI frame protocol. The remote reference layer is adopted to find the communication object. The transport layer provides the interconnection of client and server based on the TCP/IP protocol.

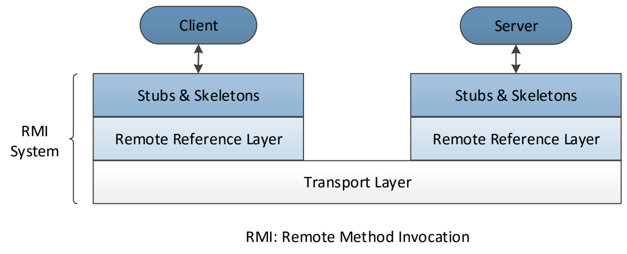


Fig. 8. Agency mechanism.

On the contrary, lightweight SCA 4.1 and STRS do not utilize the mechanism of agency, and launch all GPP components in a process address space. In this way, we could gain higher efficiency without the burden of serialization, deserialization and various protocols, etc.

III. TESTBED INPLEMENTATION

The experiments are carried out on our own testbed, which is shown in Fig. 9. The hardware ZLSDR-1000 is a high performance general SDR platform, which is designed for rapid prototype of waveforms. The main chip inside is ZYNQ 7030 which includes a dual-core of ARM Cortex-A9 (clock speed 667MHz) and a FPGA of Kintex-7 (logic cells 125K, DSP Slices 400). The DDR memory size is 1GB. This platform contains the Linux 3.17 operating system, which is convenient for users to develop applications in a Ubuntu system. All experiments are carried out on this testbed.

Fig. 9. General SDR platform (ZLSDR-1000).

There are several aspects that may affect the performance, e.g., packet size, total amount of packets and number of components, etc. Four metrics are adopted to investigate the performance between STRS, lightweight SCA 4.1 and full SCA 4.1, namely, static memory occupation, inter-components communication delay, waveform deployment delay and waveform switching delay. The delay is captured by adding timestamps at the input or output port of a component and the inter-components communication delay is presented in Fig. 10.

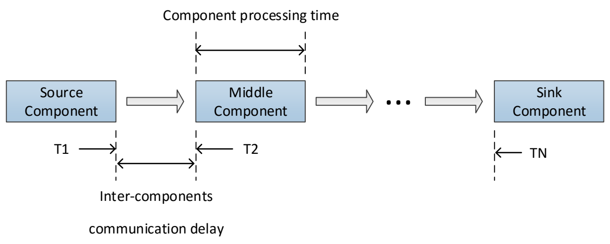

Component processing time
Source Component
Middle Component
Sink Component
T1
T2
TN
Inter-components communication delay

Fig. 10. Inter-components communication delay.

As shown in Fig. 10, the timestamp T1 is recorded before the source component sends a packet and the timestamp T2 is recorded after the middle component receives a packet. The inter-components communication time of a single link (T) can be computed as [T=(T2-T1)]. The experiments only record two timestamps (T1 and TN). The timestamp TN is recorded after the sink component receives the last packet. The difference of TN and T1 (TN-T1) approximately represents the amount time of all links in inter-components communication with the component processing time has been curtailed extremely to approach zero.

The waveform deployment delay and waveform switching delay are displayed in Fig. 11. As can be seen from Fig. 11, the waveform deployment delay is comprised of the delays of uploading and initializing the waveform, and the waveform switching delay includes of delays of waveform stop to waveform launch.

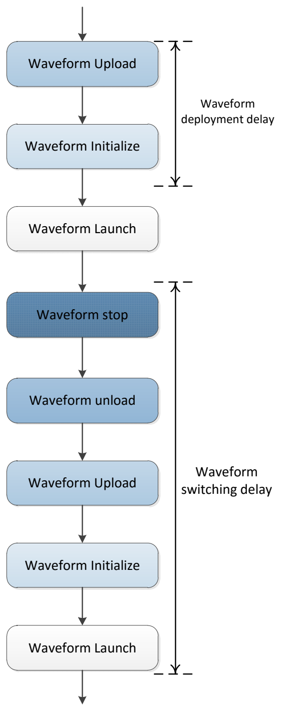


Fig. 11. Waveform deployment delay and waveform switching delay.

The efficiency of inter-components communication is a critical element for evaluating a SDR system and is influenced by several aspects, including packet size, total amount of packets and number of components. The number of links of inter-components, which also affects the delay, is not considered as a metric as our test waveform is a cascaded flow waveform with no processing codes in each middle components [13].

IV. Experiment results and analysis

To compare the performance between Lightweight SCA 4.1 and STRS, we carry out extensive experiment and discuss the results in this section. The results of full SCA 4.1 are presented as benchmarks. The experiment parameters are listed in Table I.

TABLE I. Experiment Parameters

Parameter	Value
Packet size (bytes)	128, 256, 512, 1024, 2048, 4096
Amount of packets (×10^6)	1, 2, ..., 10
Number of components	3, 4, ..., 11

In the experiment, continuous packets are pushed from the source component to the sink component. The purpose is not only to evaluate the delay performance but also to conduct pressure test. The test is to evaluate the long-term performance of the system with different core frameworks and transfer mechanisms.

A. Experiment Results

Table II shows the static memory occupation of STRS, lightweight SCA 4.1 and full SCA 4.1. As shown in Table II, the memory occupation of STRS is much smaller than that of lightweight SCA 4.1 and full SCA 4.1, which is more suitable for memory limited radio systems.

TABLE II. Static Memory Occupation Comparison

Implementation	Static memory occupation (MB)
STRS	4.77
Lightweight SCA 4.1	27.82
Full SCA 4.1	77.20

Table III illustrates the communication delay of each link between components in microsecond with different packet sizes. The number of component is 11. As shown in the table, the communication delay of STRS remains almost the same for different packet sizes. The communication delay of lightweight SCA increases with the packet size when it is small, and finally tends to saturate when the package size raises to 2048 bytes. The communication delay of full SCA 4.1 keeps increasing with the packet size. The delay of STRS and lightweight SCA is close, while that of full SCA 4.1 is over ten times larger than the other two frameworks. It should be noted that, in the experiment one packet is passed through 10 links from the transmitter to the receiver since the component number equals to 11.

TABLE III. Inter-components communication delay comparison with different packet sizes

Package size (bytes)	STRS (us)	Lightweight SCA (us)	Full SCA 4.1 (us)
128	17.30	17.20	189.00
256	17.30	18.40	189.00
512	17.30	19.00	196.00
1024	17.30	20.80	200.00
2048	17.30	21.00	216.00
4096	17.50	21.00	240.00

Table IV presents the total time consumption with different amount of packets. As shown in the table, that of STRS, lightweight SCA and full SCA 4.1 all increase linearly with the amount of data package in almost the same rate.

TABLE IV. Total Time Comsuption Comparison with Different Amount of Packets

Amount of packets	STRS (s)	Lightweight SCA 4.1 (s)	Full SCA 4.1 (s)
100,000	17.50	21.00	240.00
200,000	40.10	42.20	476.00
300,000	66.70	63.60	715.00
400,000	81.60	85.00	946.00
500,000	101.90	105.40	1190.00
600,000	119.20	125.40	1441.00
700,000	143.10	147.80	1669.00
800,000	164.90	162.20	1905.00
900,000	184.10	191.00	2142.00
1,000,000	201.70	207.40	2399.00

Table V shows the communication delay of each link between components in microsecond. As shown in the table, that of STRS, lightweight SCA and full SCA 4.1 increase with the components number.

TABLE V. Inter-components communication time comparison in the component number

Number of components	STRS (us)	Lightweight SCA 4.1 (us)	Full SCA 4.1 (us)
3	15.50	23.00	175.00
4	16.00	19.67	186.67
5	16.75	20.50	197.50
6	16.00	19.80	208.00
7	16.50	21.33	211.67
8	16.43	20.14	218.57
9	17.00	20.75	227.50
10	17.22	20.22	233.33
11	17.50	21.00	240.00

Table VI shows the waveform deployment delay of each SDR architecture in millisecond. As shown in the table, the deployment delay increases with the component number for all of the three SDR architecture used in the experiment.

TABLE VI. WAVEFORM DEPLOYMENT DELAY COMPARISON IN THE COMPONENT NUMBER

Number of components	STRS (ms)	Lightweight SCA 4.1 (ms)	Full SCA 4.1 (ms)
3	12	46.7	843.3
4	13.3	51	1078
5	14.7	55	1318
6	16	58.7	1588
7	17.7	62	1841.8
8	18.7	65.3	2130
9	20	69.7	2385
10	20.5	73	2641.6
11	21	79.6	2903.7

Table VII shows the waveform switching delay of each standard in millisecond. Through the observation, the deployment delay increases with the increasing of the amount of components for all of the three standards.

TABLE VII. WAVEFORM SWITCHING DELAY COMPARISON IN THE COMPONENT NUMBER

Number of components	STRS (ms)	Lightweight SCA 4.1 (ms)	Full SCA 4.1 (ms)
3	13	51.6	880.6
4	14.6	56	1124
5	15.7	60	1376.3
6	17.7	64.4	1661.6
7	19.4	67.7	1930.4
8	20.4	71.3	2234
9	21.6	76.7	2502
10	22.2	79	2769.6
11	23	85.9	3035.1

B. Analysis

According to the experiment results, lightweight SCA 4.1 and STRS have very close transfer efficiency. Their transfer delay is almost 1/10 of full SCA 4.1.

As shown in Fig. 12, each component in full SCA 4.1 creates a new process and keeps "alive" by circling the accepting code inside. Components accept and send packet once available from middleware, as shown in Fig. 11. While Components in STRS and lightweight SCA 4.1 fetch and push packets by transfer mechanism. The creation and communication between processes devote to the lower efficiency in full SCA 4.1.

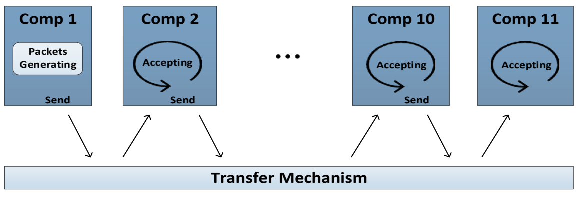


Fig. 12. Circling mission of SCA component

Another difference for full SCA 4.1 is the adoption of object request broker (ORB) mode, which brings more manipulations such as serialization and deserialization, which involuntary cause the decrease of efficiency.

The standards of STRS and SCA both have their advantages and disadvantages. The merits of STRS are its high efficiency and the fact that it can operate on the lightweight platform which is resource-limited. In the earlier versions of VxWorks, e.g., 5.4 and 5.5, that use the sharing address space for all tasks, the efficiency of the inter-task communication is quite high. STRS also has disadvantages in term of portability and interoperability. The portability and interoperability of STRS is lower than other SDR architectures, making it less attractive in implementing ground-based communication systems. Meanwhile, the implementation of STRS does not provide the agency mechanism, placing more difficulty for the application developer. The merits of the SCA are increasing flexibility by adopting self-defined lightweight middleware and reducing the expenditure with thread communication. The drawbacks of the SCA are high waveform deployment delay, high waveform switching delay and large static memory occupation.

V. CONCLUSIONS

This paper evaluate the performance of three SDR implementation, i.e., STRS, lightweight SCA and full SCA 4.1 with four typical metrics. The experiment results demonstrate that the lightweight SCA 4.1 and STRS have similar performance on inter-components transfer efficiency. The experiment results also show that the efficiency of full SCA 4.1 CORBA is much lower than the other two SDR architectures in all aspects. According to the experiment results, it is worth considering adopting lightweight SCA 4.1 in the space platform owing to its high efficiency. However, when use this architecture, one should pay special attention to the waveform deployment, switching delay and static memory occupation. Possible future work include carrying out experiments in platform with limited computing capability and memory size. Besides, the experiments can also be extended to evaluate the power consumption of different architectures, which is also a challenge in space environment.

ACKNOWLEDGEMENT

This research was supported in part by the National Natural Science Foundation of China (Grant No. 61601482).

REFERENCES

[1] Adrat M, Ascheid G. Special Issue on Recent Innovations in Wireless Software-Defined Radio Systems[J]. signal processing systems, 2015, 78(3): 239-241.

[2] Cheng Y, Xu K, Hu Y F, et al. Technology demonstrator of a novel software defined radio-based aeronautical communications system[J]. Iet Science Measurement & Technology, 2015, 8(6): 370-379.

[3] Baranda J, Henarejos P, Gavrincea C G. An SDR implementation of a visible light communication system based on the IEEE 802.15.7 standard[C] International Conference on Telecommunications. IEEE, 2013:1-5.

[4] "Software Communications Architecture (SCA) Specification," Joint Tactical Networking Center (JTNC), Version 4.1, August 2015.

[5] Grahamcumming J. The GNU Make Book[J]. 2015.

[6] Gomez I, Marojevic V, Bracke J, et al. Performance and overhead analysis of the ALOE middleware for SDR[J]. 2010, 49(1):1134-1139.

[7] Reinhart R C, Johnson S, Kacpura T J, et al. Open Architecture Standard for NASA's Software-Defined Space Telecommunications Radio Systems[J]. Proceedings the IEEE, 2007, 95(10): 1986-1993.

[8] Adrat M, Bernier S P, Buchin B, et al. A Technical Review of SCA Based Software Defined Radios: Vision, Reality and Current Status[J]. Journal of Signal Processing Systems, 2016: 1-11.

[9] Reinhart R C, Johnson S K. NASA's SDR Standard: Space Telecommunications Radio System[J]. 2007.

[10] Putthapipat P. Lightweight Middleware for Software Defined Radio (SDR) Inter-Components Communication[J]. 2013.

[11] "Software Communications Architecture (SCA) Specification," JTRS Standards, Joint Program Executive Office (JPEO) Joint Tactical Radio System (JTRS), Version 2.2.2, May 2006.

[12] "Space Telecommunications Radio System (STRS) Architecture Standard, NASA-STD-4009", NASA Technical Standard, June 2014.

[13] Putthapipat P, Andrian J, Liu C, et al. Studies on inter-component communication latency based on variation number of components and packet size in SDR-SCA waveform application[J]. International Journal of Computational Science and Engineering, 2016, 12(1): 65-72.

Investigation of High-Efficient Transfer Mechanisms for SCA 4.1

Yu Zhao[1], Li Zhou[1*], Qi Tang[1], Dongtang Ma[1], Haitao Zhao[1], Shan Wang[1], Jibo Wei[1]
[1]College of Electronic Science and Engineering, National University of Defense Technology, China
[*]Email: zhouli2035@nudt.edu.cn

***Abstract*—SCA 4.1 proposes the concept of transfer mechanism, which eliminates the dependence on CORBA and enables to use reconfigurable transfer mechanisms. Transfer mechanism is the combination of series of low-level transfer functions. Depending on the core framework design, it could be in the form of a middleware or realized by directly encapsulating several low-level transfer functions. In this paper, we compared the efficiency of three typical middlewares, namely, omniORB, Binder and ZeroMQ and explored the loss of efficiency due to the encapsulation of low-level transfer functions. Finally, we proposed a transfer mechanism named KD-RPC, which embeds TCP socket, POSIX message queue, shared memory and ZeroMQ with common application program interfaces (APIs). The experiment results show that KD-RPC achieves better performance than omniORB and Binder.**

Keywords—Software Defined Radio; Software Communication Architecture; transfer mechanism; Common Object Request Broker Architecture; efficiency

I. Introduction

As a defacto Software Defined Radio (SDR) standard, SCA is rapidly advancing in recent years in terms of scalability, flexibility, extensibility and adaptability. In former SCA standard versions, CORBA is adopted to improve portability and interoperability. However, the latest version, SCA 4.1, eliminates the dependence on CORBA by introducing the concept of transfer mechanism [1]. SCA 4.1 does not prohibit the usage of CORBA, meanwhile, it enables to use other transfer mechanisms with common application program interfaces (APIs), thus providing more flexibility. With the reconfigurable feature, SCA 4.1 is expected to provide better performance in the size, weight and power (SWaP) constrained conditions. In this paper, we introduce a reconfigurable transfer framework and investigate the performance of different transfer mechanisms.

In recent years, the research on middleware has become very popular due to the widespread deployment of distributed systems. Basically, middlewares can be classified into three categories, namely, Object Request Broker (ORB), Remote Procedure Call (RPC) and Message-Oriented Middleware (MOM) [2]. As representative examples, CORBA, Binder and ZeroMQ are selected in this paper.

ORB is the core component of Common Object Request Broker Architecture (CORBA) and was published in 1990 by Object Management Group (OMG). It enables the interaction between the client and the server in a transparent way. Owing to its advantages, such as function completeness, implementation independence and programming language adaptability, CORBA is followed by many distributed computing platform companies and has become the standard of distributed computing techniques.

RPC realizes the function of conversation layer — it builds a logical channel between two processes that trying to communicate, and releases the channel when the communication is over, no matter that the process is in a local or remote node. The communication model of RPC is simple and clear: the server waits the request issued by the client all the time, calls the appointed process when the request arrives and then returns the result. In Android platform, RPC is realized by Binder, an IPC mechanism based on Binder Driver and Service Manager [3].

MOM can provide the data transfer cross platforms by efficient and reliable message transfer mechanism. By providing message transfer and queue model, MOM can expand the inter-process communication in distributed environment and support multiple protocols, languages, hardware and software platforms. ZeroMQ is a concurrent framework based on sockets that carry atomic messages across various transports like intra-process, inter-process, TCP, and multicast. It can achieve N-to-N connections with patterns like fan-out, pub-sub, task distribution, and request-reply. Its asynchronous I/O model gives the scalable multicore applications, built as asynchronous message-processing tasks [4]. It has a score of language APIs and runs on most operating systems.

The reminder of the paper is organized as follows. Section II introduces the implementation of omniORB, Binder and ZeroMQ. In section 3, we compare the efficiency of three middlewares mentioned above. In section 4, we propose a self-defined transfer mechanism by encapsulating low-level POSIX functions in standard APIs.

II. Implementation of Transfer Mechanisms

In this section, we will further introduce the transfer mechanisms mentioned above and explain the usage of them in our experiment.

A. OmniORB

OmniORB is an ORB that implements version 2.6 of the CORBA specification [5]. The goal of omniORB is to provide the most generally useful parts of the specification in a clean and efficient manner but it does not slavishly implement every last part. The main missing features of omniORB are:

- omniORB does not have its own Interface Repository.
- omniORB supports interceptors, but not the standard Portable Interceptor API.

The communication architecture of omniORB is the same as CORBA. Figure 1 depicts the architecture of omniORB.

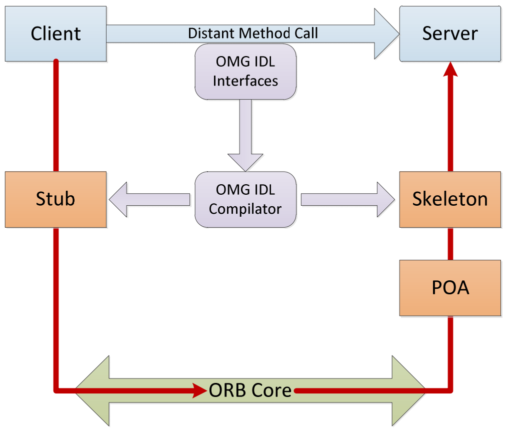


Figure 1. CORBA communication architecture[5].

In addition, CORBA defines two message transfer methods, i.e. one-way and two-way method. In one-way method, the client does not wait for completion of the request, nor does it intend to accept results; while the two-way method would block the client until the request is completed in the form of a return value, an exception, or an "out" parameter from the server back to the client [6].

Data transfer is realized by the Client's calling of PushPacket() function of object reference of the Server.

B. *Binder*

Binder, namely Android RPC, is used for inter-process communication in Android systems. Binder consists of four parts: Client, Server, Service Manager and Binder Driver [3], as shown in Figure 2.

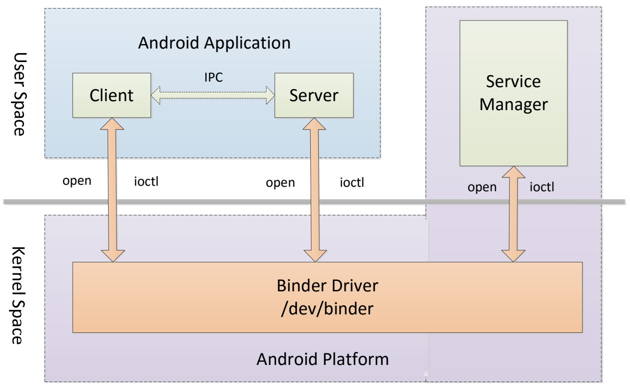


Figure 2. Binder mechanism [3].

Service Manager, Server and Client runs in separate processes respectively. Service Manager is protective process of Binder mechanism, and it manages Servers created by user and enables Client to inquire the remote interfaces of Server [7]. After Server registering its service in Service Manager, Client can communicate with Server via Proxy of Server, the detailed execution process is shown in Figure 3. The communication between Client and server is synchronous.

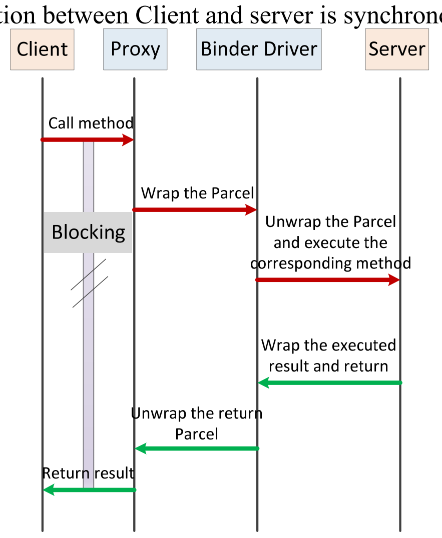


Figure 3. Execution process of Binder.

C. *ZeroMQ*

The architecture of ZeroMQ is shown in Figure 4. ZeroMQ create the I/O threads according to the parameter of function zmq_init() called by user, and the number of I/O threads corresponds to the input parameter. Each I/O thread has a Poller bound to itself. Poller is implemented by Reactor mode and it adopts different I/O modes (such as select, poll, epoll, kequeue, etc.) depends on the operating system. The main thread communicates with the I/O thread via Mail Box. When the server starts to listen, the main thread would create zmq_listener and bind it to the I/O thread, and then, the I/O thread would add zmq_listener to Poller to listen write/read events. When the client initiates a connection, the process is the same except zmp_listener is changed with zmq_connecter.

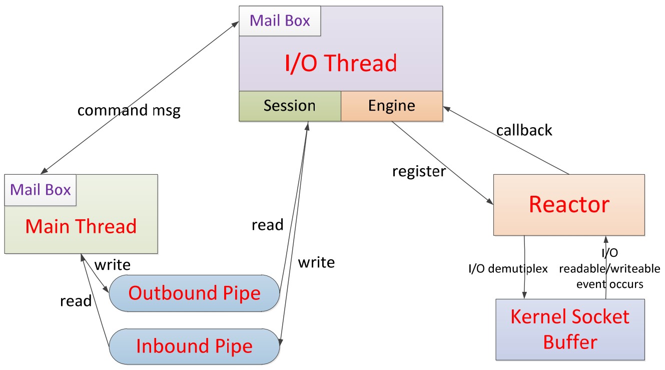


Figure 4. The architecture of ZeroMQ [8].

An authentication between the server and the client is asked for the first time they communicate with each other. The authentication is realized by sending identity from zmq_init. Afterwards, a session is created to support later communications. Each session is associated with a write/read pipe.

ZeroMQ provides multiple transfer protocols (such as one-to-one, request/reply, publish/subscribe and survey) and transport mechanisms (such as in-process, inter-process, TCP and WebSocket), enabling users to adapt it to satisfy various requirements.

Pair protocol is the simplest and least scalable scalability protocol. It allows scaling by breaking the application into exactly two pieces. The drawback of this protocol lies in limitation of scalability.

Inter-process transport allows for sending messages between processes within a single central processing unit (CPU) core. The implementation uses native IPC mechanisms provided by the local operating system and the IPC addresses are thus OS-specific. On POSIX-compliant systems, UNIX domain sockets are used and IPC addresses are file references. This transport mechanism is not reliable. TCP transport allows for passing messages over the network using simple reliable one-to-one connections.

The interfaces provided by ZeroMQ are very simple and clean. Figure 5 shows the function execution process of the communication between the client and the server.

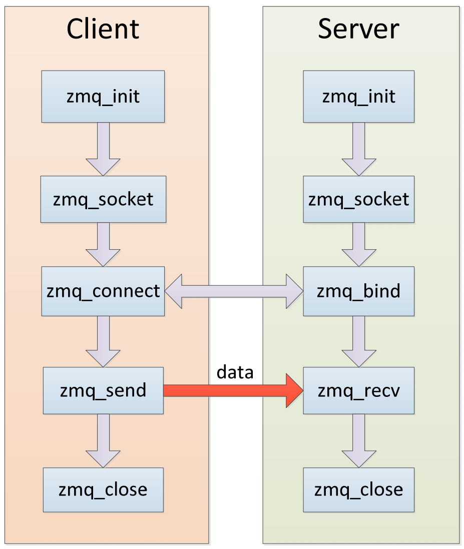


Figure 5. Function execution process of ZeroMQ.

III. Efficiency Comparison

A. *Transfer Delay Analysis*

In SCA, the transfer mechanism plays two roles. The first is to facilitate the communication between software components, and the second is to simplify the implementation of the software components and core framework. Between these two roles, the communication role is of the most importance. The transfer delay is the critical measure for evaluating the performance of a transfer mechanism.

Transfer delays between components, shown in Figure 6, consists of the following parts:

I. Delays of encoding and decoding;

II. Transfer delays of low-level transfer functions specified by middleware;

III. Other delays.

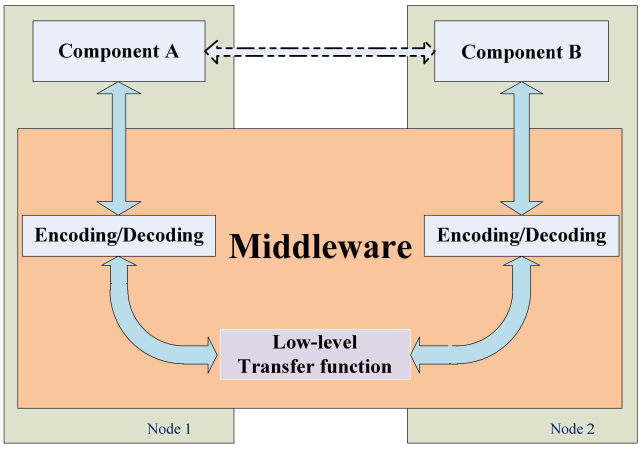


Figure 6. Inter-component communication.

Encoding includes data formatting, serialization and data checking, etc. Decoding is the reverse process of encoding. Generally, Part I depends on the structure and design of the middleware. Part II is determined by the efficiency of specified transfer functions, such as message queue, shared memory and TCP socket. Part III includes delays in system schedule, thread management and the others determined by design, strategy and type of middleware.

B. *Test Environment and Methods*

All the experiments run on our general SDR Platform ZLSDR-1000. The baseband chip is ZYNQ 7030 which includes a dual-core CPU and an FPGA. Each core is an ARM Cortex-A9 core whose clock speed is 667MHz. The FPGA is Kintex-7 125K logic cells and 400 DSP Slices. The memory size is 1GB. The operating system on the platform is Linux 3.17.

As can be seen from Figure 7, the test waveform is made up of several components which are connected in a cascade flow. The SourceComp generates packets and pass them to a middle component, until the SinkComp receive the packets. There could be more than one middle components and there are no processing code in the middle component, which means the received packets are send out directly. T1 and T2 are obtained by calling the clock_gettime() function, which provides a precision of nanosecond. We take the results from Monte Carlo simulation of 500 trials.

The detailed steps are as follows.

1) Set the packet size as 1024 bytes.

2) Set the number of components as 2.

3) Record the first time SourceComp sends packet as T1.

4) SourceComp continually sends 1 million packets.

5) Count the amount of packets that SinkComp receives and record the time T2 when it reaches 1 million.

6) Calculate the interval of T2 and T1 and record it.

7) Set the number of components as 3,4,5,6 and 10 respectively, repeat step3 to 6.

8) Set the packet size as 256, 512, 1024, 4096 and 8192 bytes respectively, repeat step 3 to 6.

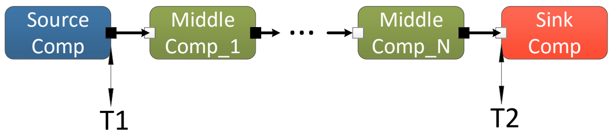


Figure 7. Waveform created for measurement.

In the experiment, all middlewares and transfer functions are configured with default settings. As an exception, ZeroMQ and pure TCP socket are configured with the sending buffer and receiving buffer both set as 128KB. ZeroMQ-TCP and ZeroMQ-IPC mean the protocol of ZeroMQ is set to TCP and IPC respectively.

C. *Results*

A large number of measurements were carried out in the experiment. Firstly, three typical middlewares, omniORB, Binder and ZeroMQ are compared. Figure 8 shows the delay of single packet transfer when packet size is 1024 bytes. The component number varies from 2 to 10, as the horizontal axis shows.

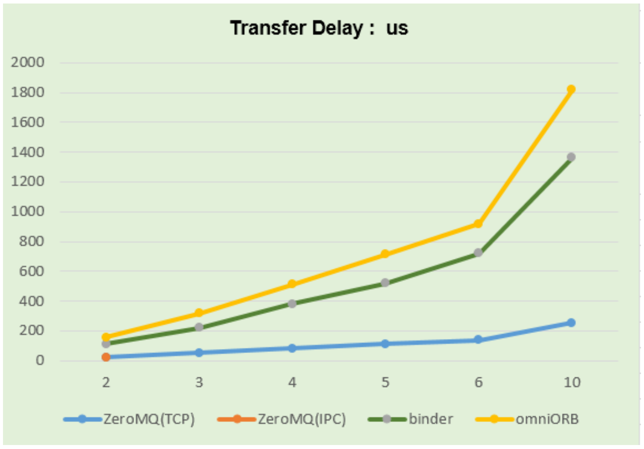


Figure 8: Delay measurements with different number of components , the packet size is 1024 Bytes.

As shown in the figure, omniORB consumes the longest time while ZeroMQ-TCP consumes the least time. Especially when the component number is 2, delay of omniORB reaches 7.22 times of ZeroMQ-TCP. Averagely, the delay of ZeroMQ-TCP is 1/5 of Binder and 1/7 of omniORB. It should also be noted that three middlewares have approximately the same behavior—transfer delay increases almost linearly with the number of components. Notice that ZeroMQ-IPC has only one point because Unix domain socket, the low-level transfer function specified by "IPC", is not reliable and packet loss occurs when the component number is larger than 2.

Figure 9 illustrates the transfer delay of different middlewares when packet size varies from 128 bytes to 8192 bytes. In this experiment, the number of components is set to 2. The general results are the same as the former experiment, omniORB is the slowest one and ZeroMQ is the fastest one. This diagram can be divided into two parts: when the packet size is smaller than 1024 bytes (4096 bytes for omniORB), as the packet size decreases, the transfer efficiency doesn't improve remarkably. Besides, in this region, ZeroMQ shows more advantage: its transfer delay is about 1/5 of Binder and 1/7 of omniORB. In the second part, the transfer delay increases almost linearly as the packet size grows. What's more, this trend holds when the packet size is bigger than 8192 bytes. In this region, the delay of ZeroMQ-TCP is approximately 1/3 of Binder and omniORB. Also notice that ZeroMQ-IPC has only 4 points in the figure because of packet loss when the packet size is larger than 1024 bytes.

From the experiment, we can observe that if we ignore the packet loss, there are almost no efficiency difference between ZeroMQ-IPC and ZeroMQ-TCP.

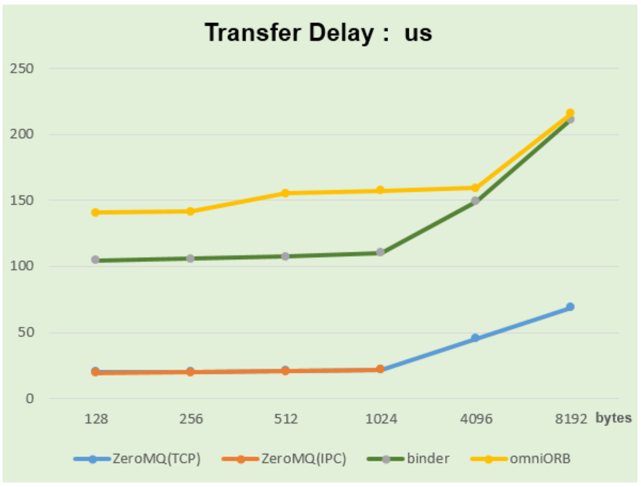


Figure 9. Delay measurements with different packet sizes, the number of components is 2.

The second part of the experiment is to explore the loss of efficiency due to the encapsulation of low-level transfer functions. As a high-efficient middleware, ZeroMQ-TCP is chosen as a benchmark to be compared with two low-level transfer functions—TCP socket and POSIX message queue. Figure 10 shows the delay of single packet transfer with component number varying from 2 to 10 when packet size is 1024 bytes.

As a Message-Oriented Middleware, ZeroMQ is different from omniORB and Binder: it simply transfers the data without any extra processing. But the impact on the efficiency due to encapsulation is still significant. Averagely, the time ZeroMQ-TCP spends is 5.46 times of pure TCP socket. This gap can be up to 38.20 when compared with omniORB (not shown in this

Figure). Beside, as component number increases, this gap remains almost constant.

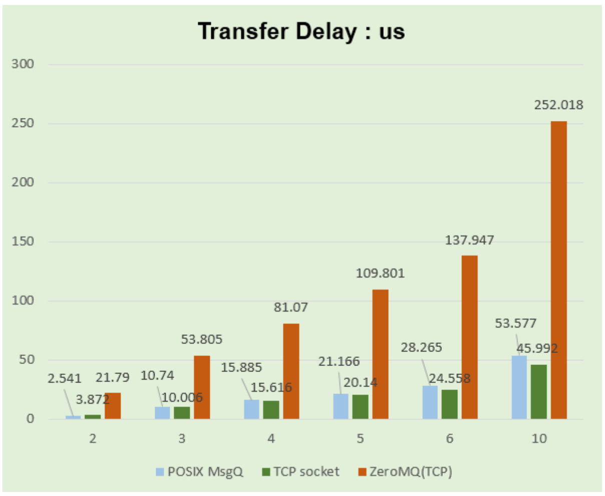


Figure 10. Delay comparison among ZeroMQ-TCP and low-level transfer functions with different number of components, the packet size is 1024 bytes.

Figure 11 presents the delay with different packet sizes when component number is 2. When data size is 128 bytes, the time ZeroMQ-TCP consumes is 8.57 times of pure TCP and it drops to 3.42 times as the packet size reaches 8192 bytes. At the same time, the absolute difference increases much (from 17.79 µs to 48.67 µs).

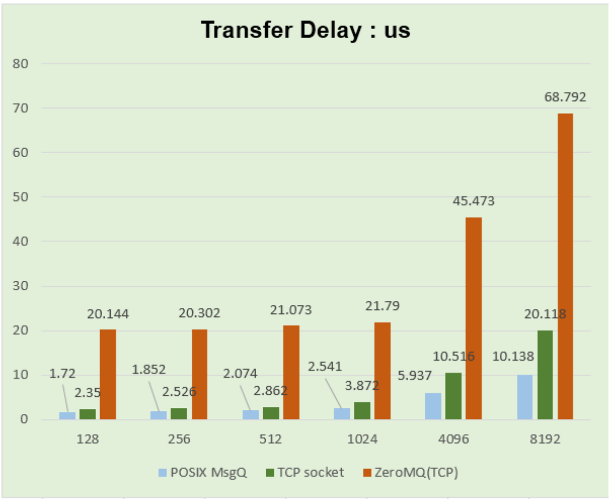


Figure 11. Delay comparison among ZeroMQ-TCP and low-level transfer functions with different packet sizes, the number of component is 2.

IV. KD-RPC

We propose a portable, OS-independent and high-efficient transfer mechanism named KD-RPC in this section. The main features are as follows.

- Based on remote process call (RPC);
- Self-adaptive and pluggable transfer functions;
- Self-defined frame structure and serialization approach.

The hierarchical structure of the KD-RPC is shown in Figure 12.

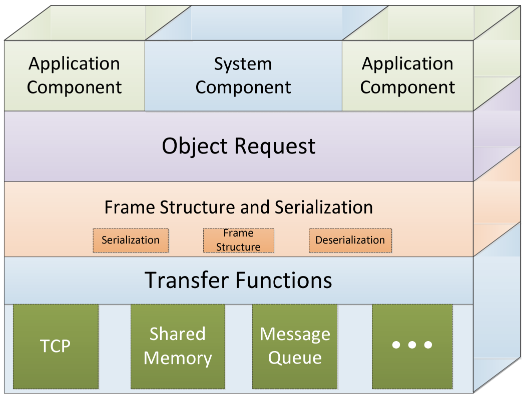


Figure 12. Hierarchical structure of KD-RPC.

A. *Self-Adaptive and Pluggable Transfer Functions*

KD-RPC comprises several modules that simplify the application implementation, so that the application developer does not have to understand much about the platform-specific details and focuses on the implementation. One key module the KD-RPC provided is the module that provides the connection among different components. KD-RPC is self-adaptive by encapsulating several low-level inter-process and network communication methods using a pluggable transfer protocol. users do not need to know which low-level transfer function is used when they call a remote method. Rather, the transfer mechanism can automatically choose the efficient transfer function according to the type of object request. For example, if the call is local, then the transfer mechanism uses the shared memory or POSIX message queue for communication; otherwise, it uses socket . Furthermore, any transfer functions can be added to or removed from KD-RPC with very limited modifications. Currently, KD-RPC has embedded TCP socket, POSIX message queue, shared memory and ZeroMQ.

Besides, the static configuration is also possible for our transfer mechanism by using the configuration file, which is useful when the application requirements are stable or known in advance.

B. *Frame Structure and Serialization*

Section III. A analyzed the transfer delay of the middleware and encoding and decoding are significant parts of it. Here, we focus on two factors: frame structure and serialization.

To enable the access of a remote object transparently from the client, a frame structure is necessary to encapsulate the request information and the return information. The frame structure should provide the interface name, the object, and function along with relative parameters that the client calls.

To improve the transfer efficiency, we design two kinds of frame structures—one is used for in-board communication and the other is for inter-board communication. Assuming that a 128 bytes packet is to be sent, when using the GIOP defined in CORBA, the size of whole message will be 176 bytes, however, the message size is reduced to 140 bytes using the self-defined frame structure.

Serialization transforms message to stream so that data can be transferred in a continuous form. Deserialization is the reverse process of serialization that makes sure processors in distributed platform can retrieve the data structure.

Currently, The serialization implemented by KD-RPC is compatible with Boost library and supports multiple data types, such as standard container, self-defined type and so on. Besides, data serialized by KD-RPC is binary stream, which shortens the length of stream and further improves the efficiency.

C. *Performance Tests*

Table I compares the transfer delay of KD-RPC, TCP socket and omniORB with different packet sizes varying from 128 bytes to 8192 bytes. Tests are also carried out on a virtual machine with an Intel 3.2 GHz dual-core processor and 1 GB RAM. The operating system is Ubuntu Linux 4.2.

TABLE I. Delay Comparison among KD-RPC, TCP and omniORB

Type \ Packet size (B)		128	256	512	1024	4096	8192
TCP socket	VM[a]	0.344	0.349	0.379	0.477	0.825	1.660
	ARM	2.350	2.526	2.862	3.872	10.52	20.12
KD-RPC	VM	3.877	4.558	4.457	4.969	7.046	11.62
	ARM	69.05	72.76	79.01	90.20	157.6	257.4
omniORB	VM	48.49	49.65	47.46	48.26	48.15	55.43
	ARM	140.6	141.4	155.1	157.4	159.3	215.9

[a] VM: Virtual Machine.

KD-RPC adopts TCP socket as its low-level transfer function. On the virtual machine, KD-RPC performs much better than omniORB. The delay of KD-RPC is about 10% of omniORB when the packet size is smaller than 4096 bytes. When the packet size reaches 8192 bytes, the delay of KD-RPC is 1/5 of omniORB. Meanwhile, we observe that, due to the encapsulation of TCP socket, the delay of KD-RPC is averagely 10 times of the pure TCP socket.

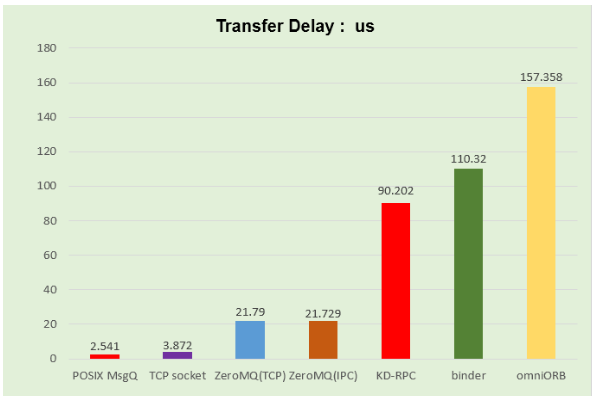


Figure 13. General transfer delay comparison, the number of components is 2 and the packet size is 1024 bytes.

On ZLSDR-1000, when packet size is smaller than 4096 bytes, the transfer delay of KD-RPC is around 50% of omniORB. However, the performance of KD-RPC is quite close to omniORB when packet size is from 4096 or 8192 bytes. We suppose that the performance degradation might be related to the CPU, driver and other hardware architecture. Future optimization is necessary to enhance its efficiency.

Figure 13 shows the delay comparison between KD-RPC and all the middleware and low-level transfer functions tested in this paper when the number of components is 2 and packet size is 1024 bytes.

V. Conclutding Remarks

All experiments in this paper confirm one fact, i.e. there is a tradeoff between universality and efficiency. As an open source version of CORBA, omniORB implements various functions and services defined in CORBA, providing cross-platform communication and interoperability among different OSs and programming languages. However, it does not has much advantage in terms of efficiency compared with other transfer mechanisms. Binder is an RPC realized in Android platform. This mechanism improves the efficiency by adopting one-time copy technique. Nonetheless, the biggest drawback of Binder is that it does not support network communication. ZeroMQ highly encapsulated low-level protocols and transfer functions and is provided with very clean and simple interfaces. Users can easily realize N-to-N communication in multiple network topology with ZeroMQ. However, as a Message-Oriented Middleware, ZeroMQ does not support more advanced functions and services but merely data transfer.

SCA 4.1 provides the possibility of further improving the communication efficiency between components, and KD-RPC is just the attempt at this issue. KD-RPC can flexibly load and use various low-level transfer functions and exploit their advantages to the full. Besides, the improvement of efficiency is partly at the price of sacrificing universality and portability.

Acknowledgement

This research was supported in part by the National Natural Science Foundation of China (Grant No. 61601482).

References

[1] Software Communication Architecture Specification User's Guide, Joint Tactical Networking Center, Version: 4.1, 23 February 2016.

[2] Noha Ibrahim, "Orthogonal Classification of Middleware Technologies," *Ubiquitous computing systems*, pp. 46-51, 2009.

[3] Zhongliang Li, "Analysis of Android source code," p242, first ed., April 2015.

[4] Pieter Hintjens, "ØMQ - The Guide," November, 2010.

[5] Duncan Grisby, "The omniORB version 4.2 Users' Guide,".

[6] The Common Object Request Broker:Architecture and Specification, Object Management Group, Inc. (OMG), February, 1998.

[7] Coding_glacier, "The Mechanism of Android Binder," 2012. [Online]. Available: http://www.CSDN.net.

[8] Kaka11, "Source Code Analysis of ZeroMQ," 2011. [Online]. Available: http://www.CSDN.net.

Joint Optimization of Spectrum Sensing Time and Threshold in Multichannel Cognitive Radio

Cai Zhuoran
Department of Space Integrated Electronics
Shandong Institute of Space Electronic Technology
Yantai, China
qingdaogancai@126.com

Qu Zhichao
Department of Space Integrated Electronics
Shandong Institute of Space Electronic Technology
Yantai, China
moyuertalang@163.com

***Abstract*—In order to improve the throughput of multichannel CR, a joint optimization of spectrum sensing time and sub channel sensing threshold based on the alternating direction optimization is proposed. Based on the SU's listen-before-transmit frame structure, an optimized allocation model is built to maximize the aggregate throughput of the SU over all the sub channels, providing that the communication demand of the PU and the performance of the sub channel spectrum sensing are guaranteed. The joint optimization algorithm is proposed to obtain the optimal solutions to the model through alternatively optimizing sensing threshold and time. The simulation results show that there exits the optimal sensing time and threshold that maximize the SU's throughput, and compared to the previous schemes, the joint allocation can improve the SU's throughput.**

Keywords—cognitive radio; spectrum sensing; joint optimization; throughput.

I. Introduction

Based on the conventional fixed spectrum allocation policy, most available radio spectra have been assigned to registered users, which lead to a serious waste of spectrum utilization. In fact, recent reports from Federal Communications Commission (FCC) have shown that only 30% of the allocated spectrum in US is fully utilized [1]. Cognitive radio, which enables secondary users to utilize the spectrum when primary users are not occupying it, has been proposed as a promising technology to improve spectrum utilization efficiency [2–4], and has three essential components: (1) Spectrum sensing: the secondary users sense the radio spectrum environment within their operating range to detect the frequency bands which are not occupied by primary users; (2) Dynamic spectrum management: cognitive radio networks dynamically select the best available bands for communication; (3) Adaptive communications: a cognitive radio device can configure its transmission parameters (e.g., carrier frequency, transmission power) to opportunistically make best use of the ever changing available spectrum [5].

Spectrum sensing is a fundamental task for cognitive radio, and the most common method for spectrum sensing is energy detection, which does not require any primary signal information, and it is robust to unknown dispersed channels and fading. Based on the SU's listen-before-transmit frame structure, at the beginning of each frame, the SU senses the PU, if the PU is not detected, the SU starts to transmit the data. Multichannel CR allows the SUs to use multiple idle sub channels simultaneously to improve the overall throughput. The performance of the CR is determined by sensing time and sensing threshold [5-6], the longer sensing time the better sensing performance, but the longer sensing time means the shorter transmit time for SU, and smaller detection threshold will increase the probability of detection, but it also increase the probability of the false alarm and decrease the spectrum utilization of SU. So, the appropriate sensing time and threshold is important for increasing the performance of CR. In [7], the authors optimized the sensing threshold, providing that the increase of the throughput of SU and the performance of spectrum sensing is guaranteed. In [5], the authors designed the sensing slot duration to maximize the achievable throughput for the SUs under the constraint that the PUs are sufficiently protected with a fixed sensing threshold.

In this paper, we propose a joint optimization algorithm to obtain the optimal solutions to the model through alternatively optimizing sensing threshold and time. The simulation results show that there exits the optimal sensing time and threshold to maximize the SU's throughput, compared to the previous schemes, the joint allocation can improve the SU's throughput.

This paper is organized as follows. Section II presents the model of SU's throughput and reviews the energy detection scheme. In Section III optimization of sensing time and threshold are proposed. Simulation results are given in Section IV. Conclusions are finally drawn in Section V.

II. Energy Detection and System Model

Suppose that we are interested in the frequency band with the number of sub channels is L, the discrete received signal at the secondary user in sub channel l can be represented as

$$y_l(t)=\begin{cases} n_l(t), H_l^0 \\ h_l s_l(t)+n_l(t), H_l^1 \end{cases} \qquad (1)$$
$$t=(1,2,3,\dots M)$$

Where t is the sample sequence, s_l is the PU signal with power p_s^l, n_l is the white Gaussian noise with power σ_l^2, h_l is the channel gain on sub channel l between PU and SU, M is number of samples, H_l^1 is the hypothesis that the PU is active, and the H_l^0 is the hypothesis that the PU is inactive. Let τ be the sensing time and f_s be the sampling frequency, the number of samples can be represented as

$$M = \tau f_s \tag{2}$$

The test statistic for energy detector is given by

$$Y_l = \frac{1}{M}\sum_{t=1}^{M}\left|y_l(t)\right|^2 \tag{3}$$

Where $y_l(1), y_l(2), \ldots y_l(M)$ are independent and identically distributed random process, If M is large enough, we use Gaussian approximations to the probability density functions of the test statistic Y_l

$$Y_l = \begin{cases} N\left(\sigma_l^2, \frac{1}{M}\sigma_l^4\right) \\ N\left((1+\gamma_l)\sigma_l^2, \frac{(1+2\gamma_l)}{M}\sigma_l^4\right) \end{cases} \tag{4}$$

$N(a,b)$ is a Gaussian independent and identically distributed (iid) random process with mean a and variance b, $\gamma_l = h_l^2 p_s^l / \sigma_l^2$ is the signal to noise ratio of spectrum sensing on sub channel l.

If we set the detection threshold as ε_l, the probability of false alarm and the probability of detection are given by

$$\begin{cases} P_l^f(\tau,\varepsilon_l) = Q\left(\frac{\varepsilon_l - \sigma_l^2}{\sqrt{\sigma_l^4/(\tau f_s)}}\right) \\ P_l^d(\tau,\varepsilon_l) = Q\left(\frac{\varepsilon_l - (1+\gamma_l)\sigma_l^2}{\sqrt{(1+2\gamma_l)\sigma_l^4/(\tau f_s)}}\right) \end{cases} \tag{5}$$

Where $Q(\bullet)$ is the complementary distribution function of the standard Gaussian,

$$Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^{\infty}\exp\left(-\frac{t^2}{2}\right)dt \tag{6}$$

Fig.1 shows the listen-before-transmit frame structure designed for a cognitive radio network with where each frame consists of one sensing slot and one data transmission slot.

Suppose the sensing duration is τ, and the frame duration is T. Denote $c_l^0 = \log_2\left(1+\gamma_l^{SU}\right)$ as the throughput of the SU when it operates in the absence of PU, and $c_l^1 = \log_2\left(1+\frac{\gamma_l^{SU}}{1+\gamma_l}\right)$ is the throughput when it operates in the presence of primary users. Where γ_l^{SU} is the signal to noise ratio of SU on sub channel l.

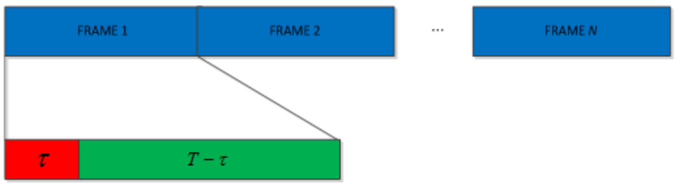


Fig.1. listen-before-transmit frame structure for cognitive radio networks

For a given frequency band of interest, let us define $P\left(H_l^1\right)$ as the probability for which the primary user is active, and $P\left(H_l^0\right)$ as the probability for which the primary user is inactive, and $P\left(H_l^1\right)+P\left(H_l^0\right)=1$. The aggregate throughput of the SU over all the sub channels is given by

$$C(\tau,\varepsilon) = \left(\frac{T-\tau}{T}\right)\sum_{l=1}^{L}\left(P\left(H_l^0\right)c_l^0\left(1-P_l^f(\tau,\varepsilon_l)\right) + P\left(H_l^1\right)c_l^1\left(1-P_l^d(\tau,\varepsilon_l)\right)\right) \tag{7}$$

We denote r_l^0 as the throughput of the SU when it operates in the absence of PU, and r_l^1 as the throughput when it operates in the presence of primary users. Then $r_l^0 = \log_2\left(1+\gamma_l^{PU}\right)$ and $r_l^1 = \log_2\left(1+\frac{\gamma_l^{PU}}{1+\gamma^{PS}}\right)$, where γ_l^{PU} is the signal to noise ratio of PU, and γ^{PS} is the signal to noise ratio from SU to PU. Then the aggregate throughput of the PU over all the sub channels is given by

$$R(\tau,\varepsilon) = \frac{P\left(H_l^1\right)}{T}\tau\sum_{l=1}^{L}r_l^0 + (T-\tau)\sum_{l=1}^{L}\left(r_l^0 P_l^d(\tau,\varepsilon_l) + r_l^1\left(1-P_l^d(\tau,\varepsilon_l)\right)\right) \tag{8}$$

III. OPTIMIZATION OF OVERALL THROUGHPUT OF SU

In this paper we propose a joint optimization algorithm to maximize the throughput of the SUs through alternatively optimizing sensing threshold and time, providing that the communication demand of the PU and the performance of the sub channel spectrum sensing are guaranteed. The optimization model can be represented by

$$\max_{\tau,\varepsilon} C(\tau,\varepsilon)$$
$$s.t. \begin{cases} R(\tau,\varepsilon) \ge \xi, \\ P_l^f(\tau,\varepsilon_l) \le \alpha, \\ P_l^d(\tau,\varepsilon_l) \ge \beta, l = 1,2,\cdots L. \end{cases} \tag{9}$$

Where $\beta \ge 0.5$ is the target probability of detection which the PU are defined as being sufficiently protected, $\alpha \le 0.5$ is the maximal probability of false detection which the SU can sufficiently use the sub channel l and the $R(\tau,\varepsilon)$ is the minimal throughput required by the PU. Equation (6) is a multiple variable optimization problem, and we use alternative optimization method to solve this problem. Firstly, we set an initiate value for sensing time τ, then we search an optimal sensing threshold ε^* to maximize the C. Secondly, we set $\varepsilon=\varepsilon^*$ to search an optimal sensing time τ^*. After some iterations, we will find the global optimal value of sensing time and sensing threshold to maximize the C.

A. Optimization of Sensing Threshold

We set the sensing time $\tau=\tau_0$ and substituting (9) into (3) gives:

$$\begin{aligned}&\varepsilon_l^{\min}<\varepsilon_l<\varepsilon_l^{\max}\\ &\varepsilon_{\min}=\left(\frac{Q^{-1}(\alpha)}{\sqrt{\tau_0 f_s}}+1\right)\sigma_l^2\\ &\varepsilon_{\max}=\left(Q^{-1}(\beta)\sqrt{\frac{2\gamma_l+1}{\tau_0 f_s}}+\gamma_l+1\right)\sigma_l^2\end{aligned}\tag{10}$$

Substituting $R(\tau,\varepsilon)\geq\xi$ into (8) gives: $\sum_{l=1}^{L}\Delta r_l P_l^d=g(\tau_0)$ where

$$\begin{aligned}&\Delta r_l=r_l^0-r_l^1\\ &g(\tau_0)=\frac{\xi}{\eta_l^1(T-\tau_0)}-\frac{\tau_0}{T-\tau_0}\sum_{l=1}^{L}r_l^0-\sum_{l=1}^{L}r_l^1\end{aligned}\tag{11}$$

Thus, (9) can be converted to

$$\begin{aligned}\max_{\varepsilon} C(\varepsilon)=&\left(\frac{T-\tau_0}{T}\right)\sum_{l=1}^{L}P\left(H_l^0\right)c_l^0\left(1-P_l^f(\varepsilon_l)\right)\\ &+P\left(H_l^1\right)c_l^1\left(1-P_l^d(\varepsilon_l)\right)\\ s.t.&\begin{cases}\sum_{l=1}^{L}\Delta r_l P_l^d(\varepsilon_l)\geq g(\tau_0)\\ \varepsilon_l^{\min}<\varepsilon_l<\varepsilon_l^{\max}\ (l=1,2,\ldots L)\end{cases}\end{aligned}\tag{12}$$

For equality constraints in (12), they can be modified to unconstrained by integrating positive Lagrange multiplier, leading to:

$$\begin{aligned}L(\varepsilon)=&\sum_{l=1}^{L}\left(P\left(H_l^0\right)c_l^0\left(1-P_l^f(\varepsilon_l)\right)+P\left(H_l^1\right)c_l^1\left(1-P_l^d(\varepsilon_l)\right)\right)\\ &+\lambda\left(\sum_{l=1}^{L}\Delta r_l P_l^d(\varepsilon_l)-g(\tau_0)\right)\end{aligned}\tag{13}$$

To maximize $L(\varepsilon)$ by requiring the $\frac{\partial L(\varepsilon)}{\partial\varepsilon}=0$, thus the threshold could be given by

$$\begin{aligned}\varepsilon_l^0=&\left(\sqrt{\frac{1}{4}+\frac{1}{2}\gamma_l+\frac{(1+2\gamma_l)}{(\tau_0 f_s)\gamma_l}\ln\left(\frac{P\left(H_l^0\right)c_l^0\sqrt{(1+2\gamma_l)}}{\lambda\Delta r_l-P\left(H_l^1\right)c_l^1}\right)}\right.\\ &\left.+\frac{1}{2}\right)\sigma_l^2\end{aligned}\tag{14}$$

Substituting (14) into $\sum_{l=1}^{L}\Delta r_l P_l^d(\varepsilon_l)=g(\tau_0)$ we can obtain the value of λ. The optimal value of sensing threshold could be given by

$$\begin{aligned}&\varepsilon_l^*=\begin{cases}\varepsilon_l^{\min},\varepsilon_l^0<\varepsilon_l^{\min}\\ \varepsilon_l^0,\varepsilon_l^{\min}<\varepsilon_l^0<\varepsilon_l^{\max}\\ \varepsilon_l^{\max},\varepsilon_l^0>\varepsilon_l^{\max}\end{cases}\\ &(l=1,2,\ldots L)\end{aligned}\tag{15}$$

B. Optimization of Sensing Time

In section III.A we set the initial sensing time as $\tau=\tau_0$ to obtain the optimal sensing threshold $\varepsilon^*=\left[\varepsilon_1^*,\varepsilon_1^*,\ldots\varepsilon_L^*\right]$, the associate probability of detection is $P_l^d(\tau_0,\varepsilon_l^*)$. In this section, we fix the sensing threshold as $\varepsilon=\varepsilon^*$, then search the optimal sensing time $\tau=\tau^*$ to maximize the throughput C. Based on (5) the probability of false alarm is represented as

$$P_l^f=Q\left(Q^{-1}\left(P_l^d\right)\sqrt{1+2\gamma_l}+\gamma_l\sqrt{\tau f_s}\right)\tag{16}$$

To ensure $R(\tau,\varepsilon)\geq\xi$ and $P_d^f(\tau,\varepsilon_l)\geq\beta$, it is obviously that $P_l^d(\tau,\varepsilon_l)>P_l^d(\tau_0,\varepsilon_l^*)$, since $Q(x)$ is a monotonically decreasing function of x, from (16), $P_l^f(\tau,\varepsilon_l)\geq P_l^f(\tau_0,\varepsilon_l^*)$, from (7) $C(\tau,\varepsilon)\leq C(\tau_0,\varepsilon^*)$. So, if $P_l^d(\tau,\varepsilon_l)=P_l^d(\tau_0,\varepsilon_l^*)=\mathrm{P}_l$ throughput of SU C will be maximal. Substituting (14) and $P_l^d=\mathrm{P}_l$ into (9) it gives

$$\begin{aligned}&\max_{\tau} C(\tau)=\left(\frac{T-\tau}{T}\right)\sum_{l=1}^{L}P\left(H_l^0\right)c_l^0(1-\\ &Q\left(Q^{-1}(\mathrm{P}_l)\sqrt{1+2\gamma_l}+\gamma_l\sqrt{\tau f_s}\right))+P\left(H_l^1\right)c_l^1(1-\mathrm{P}_l)\\ &s.t.P_l^f(\tau,\varepsilon_l)\leq\alpha,l=1,2,\cdots L.\end{aligned}\tag{17}$$

Substituting $P_l^d=\mathrm{P}_l$ into $s.t.P_l^f(\tau,\varepsilon_l)\leq\alpha,l=1,2,\cdots L.$ it gives $\tau\geq\max(\tau_1,\tau_2,\ldots\tau_L)$ and (17) can be converted to

$$\begin{aligned}&\max_{\tau} C(\tau)=\left(\frac{T-\tau}{T}\right)\sum_{l=1}^{L}P\left(H_l^0\right)c_l^0(1-\\ &Q\left(Q^{-1}(\mathrm{P}_l)\sqrt{1+2\gamma_l}+\gamma_l\sqrt{\tau f_s}\right))+P\left(H_l^1\right)c_l^1(1-\mathrm{P}_l)\\ &s.t.\tau\geq\max(\tau_1,\tau_2,\ldots\tau_L).\end{aligned}\tag{18}$$

Firstly, we must proof that when $\tau\in[0,T]$ $C(\tau)$ is a convex function of τ. $C(\tau)$ can be converted to

$$\begin{aligned}C(\tau)=&\left(\frac{T-\tau}{T}\right)\sum_{l=1}^{L}(P\left(H_l^0\right)c_l^0\left(1-P_l^f(\tau)\right)\\ &+P\left(H_l^1\right)c_l^1(1-\mathrm{P}_l))\end{aligned}\tag{19}$$

$\nabla^2 C(\tau)$ can be represented by

$$\nabla^2 C(\tau) = 2P(H_l^0)c_l^0 \nabla P_l^f(\tau) - (T-\tau)P(H_l^0)c_l^0 \nabla^2 P_l^f(\tau) \tag{20}$$

The first derivative of $P_l^f(\tau)$ is negative and second derivative of $P_l^f(\tau)$ is positive, so $\nabla^2 C(\tau) < 0$ and the $C(\tau_{\max})$ will be maximal if $\nabla C(\tau_{\max}) = 0$. The optimal sensing time is

$$\tau^* = \max(\tau_{\max}, \max(\tau_1, \tau_2, \ldots \tau_L)) \tag{21}$$

C. Joint Optimization of Sensing Time and Sensing Threshold

The process of joint optimization of sensing time and threshold is shown as below:

1) Set the initial parameter: $k=1$, $\tau^k = 0, \varepsilon^k = \{0\}$, and estimation error ξ;

2) Set $\tau^k = \tau_0, \tau_0 \in [0,T]$;

3) Substituting τ_0 into (12) to obtain the ε^*;

4) Set $\varepsilon^{k+1} = \varepsilon^*$;

5) Substituting $\varepsilon^{k+1} = \varepsilon^*$ into (15) to obtain the τ^*;

6) Set $\tau^{k+1} = \tau^*, k = k+1$;

7) Repeat 3) ~ 6) until $|\tau^k - \tau^{k-1}| \le \xi, |\varepsilon^k - \varepsilon^{k-1}| \le \xi$.

IV. Results and Discussion

In simulations, we set $T = 1s, \sigma_l^2 = 1mW$, $L = 10, \alpha = 0.5$, $\beta = 0.9$ and $P(H_l^0) = P(H_l^1) = 0.5$. The transmit power of SU and PU is $10mW$, the multi-channel gain is Rayleigh distribution with mean $-10dB$, and estimation error $\xi = 0.01$.

In Fig.2 we show the achievable throughput versus the sensing time allocated to each frame for the secondary network. It is seen that there exist a sensing time to maximize the throughput of SU. In Fig.3 we show the achievable throughput versus the average sensing threshold allocated to all sub channels for the secondary network. It is also seen that there exist a sensing threshold to maximize the throughput of SU.

In Fig.4, we compare the throughput of SU and PU, in this figure four results are shown: fixed sensing threshold and time, optimization of sensing time, compromise of sensing time and threshold, joint optimization of sensing time and threshold. It is seen that, the proposed joint optimization of sensing time and threshold leads to more throughput than other three methods.

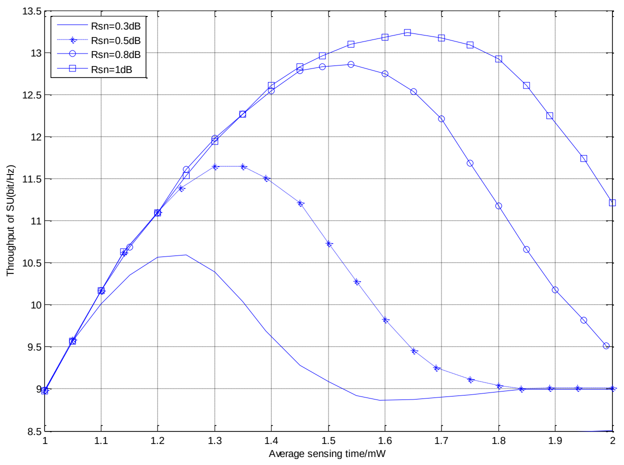


Fig.2. SU throughput versus sensing time

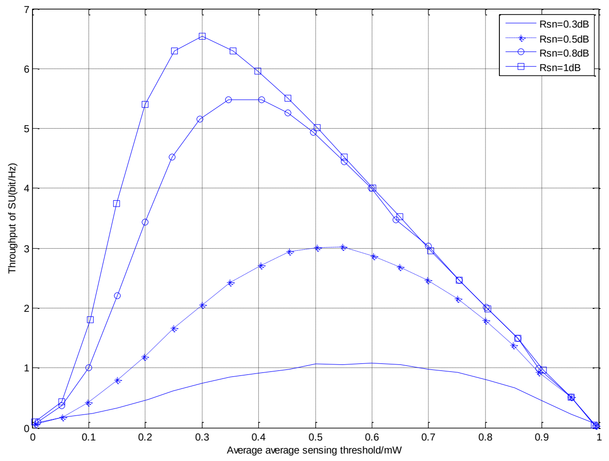


Fig.3. SU throughput versus average sensing threshold

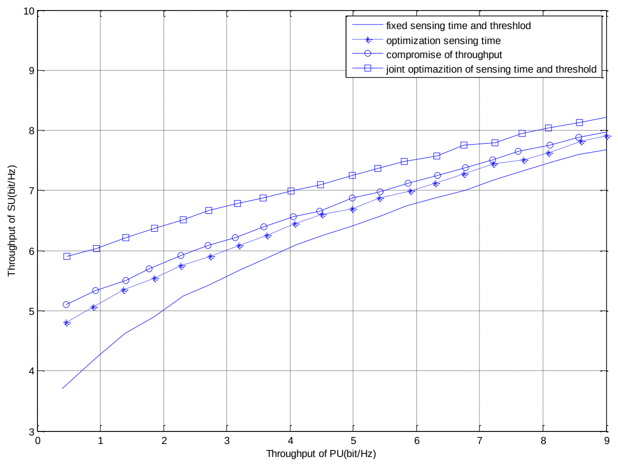


Fig.4. SU throughput versus PU throughput

V. Conclusions

Based on the SU's listen-before-transmit frame structure, a joint optimization of spectrum sensing time and threshold model to maximize the aggregate throughput of the SU over all the sub channels is proposed in this paper. The joint optimization algorithm alternatively optimizes sensing threshold and time to obtain the optimal solutions to the model.

The proposed method actually is optimization problems for sensing time and sensing threshold respectively. Theoretical analysis and simulation results show that these optimization problems have a joint global optimal solution to maximize the throughput of SU. At a given throughput of PU, the proposed joint optimization method will obtain more throughput of SU.

REFERENCES

[1] Spectrum Policy Task Force Report; Technical Report TR 02-155; Federal Communications Commission: Washington, DC, USA, 2002.

[2] Mitola, J.; Maguire, G.Q. Cognitive radio: Making software radios more personal. IEEE Personal Commun. 1999, 6, 13–18.

[3] Mitola, J. Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio. Ph.D. Thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, 2000.

[4] Haykin, S. Cognitive radio: Brain-Empowered wireless communications. IEEE J. Sel. Area Commun. 2005, 23, 201–220.

[5] Liang, Y.C.; Zeng, Y.H.; Peh, E.C.Y.; Hoang, A.T. Sensing-throughput tradeoff for cognitive radio networks. IEEE Trans. Wirel. Commun. 2008, 7, 1326–1337.

[6] Liu Xin, Jia Min, Tan Xuezhi. Threshold optimization of cooperative spectrum sensing in cognitive radio network [J]. Radio Science, 2013, 48(1): 23-32.

[7] Zhi Quan, Sayed AH, Poor HV. Optimal multiband joint sensing in cognitive radio networks [J]. IEEE Transactions on Signal Processing, 2009, 57(3): 1128-1140.

Deterministic and cognitive wireless communication system with jamming-resistant capabilities for tactical or industrial communications

Raul Torrego, Ander Etxabe, Pedro M.Rodriguez, Iñaki Val
Information and Communication Technologies
IK4-IKERLAN
Arrasate-Mondragon, Spain
{rtorrrego, aetxabe, pmrodriguez, ival}@ikerlan.es

***Abstract*—This work presents a jamming-resistant and deterministic wireless communication system, which is intended to be used in tactical or industrial communications. These kind of applications require data communication to be bounded in the time and reliability domains, no matter which is the harshness of the environment or the presence of malicious interferences. In harsh propagation environments, communication systems suffer from severe signal degradation, including delay spread, deep fading and Doppler spread. Besides, they must also deal with other system's interference and jammer attacks. The aim of this work is to propose a new communication system, which is based on the IEEE 802.11a/g physical layer, implemented on a FPGA. On top of this physical layer, a cognitive Time Division Multiple Access (TDMA) MAC layer is proposed, fulfilling real-time requirements. The radio cognitive capabilities implemented in the MAC layer allow the communication system to detect the interference generated by a jammer or other spectrum users, switching the communication to a safe frequency band. Both, simulations in OPNET network simulator and measurements on real hardware are provided in order to characterize the performance of the presented system. The results prove the capability of the system to guarantee delay bounds in tactical or industrial environments with interference.**

Keywords—Wireless communications; Deterministic, Real-time; Jamming-resistant; OFDM; TDMA; MAC; Cognitive Radio; Tactical radio; Industrial environment; FPGA

I. INTRODUCTION

The replacement of wired communication systems by wireless ones is a common trend nowadays. The benefits in terms of lower costs in materials, deployment and maintenance that wireless systems provide over their wired counterparts are highly appreciated in many fields. Specifically, industrial applications, in which the aforementioned costs are highly elevated, or tactical communications are one of the main beneficiaries of this technology.

Time-critical and mission-critical applications (automotive, aerospace, military…) require data communication to be bounded in the time and reliability domains. Communications carried out in such applications demand short latency, minimal jitter, deterministic data delivery time and high reliability. Due to the nature of these applications, these requirements must be imperatively met in order to avoid material damages or even personal injuries.

However, communications in industrial or tactical environments have to deal with harsh environments that complicate this goal. The presence of severe multipath due to reflections in metallic structures or surfaces, or the presence of multiple sources of electro-magnetic interference (high power electric motors, other wireless communication systems present nearby…) are two of the difficulties that have to be get over. Besides, a new threat has arisen in the last years: signal jammers. These devices and their users are able to generate malicious interference in order to disrupt wireless communications on purpose.

Unfortunately, traditional wireless communication systems are not able to overcome all these difficulties and, at the same time, fulfill with the aforementioned requirements needed by control application or tactical communications. As a consequence, it is necessary to deploy new wireless communication systems like the one presented in this work, based on cognitive radio technology [1].

The proposed wireless communication system, shown in Fig. 1, is based on a custom Orthogonal Frequency Division Multiplexing (OFDM) modem design which has been implemented on the programmable logic of a Xilinx Zynq Field Programmable Gate Array (FPGA). The modem is fully customizable, in case it is needed to add new features, and it is compliant with the IEEE 802.11a/g physical layer standard. On top of this modem, and being the main contribution of this paper, a deterministic, real-time and cognitive Medium Access Control (MAC) layer has been implemented and evaluated. Based on a Time Division Multiple Access (TDMA) MAC, which ensures deterministic communications in the absence of interference, cognitive capabilities have been added. Unlike traditional cognitive radios, which are used in order to enhance spectrum utilization, the presented wireless communication system is able to detect interference (malicious or coming from other wireless communication systems) and switch the communication to an unoccupied and safe frequency band. Simulations in OPNET and measurements on real hardware are

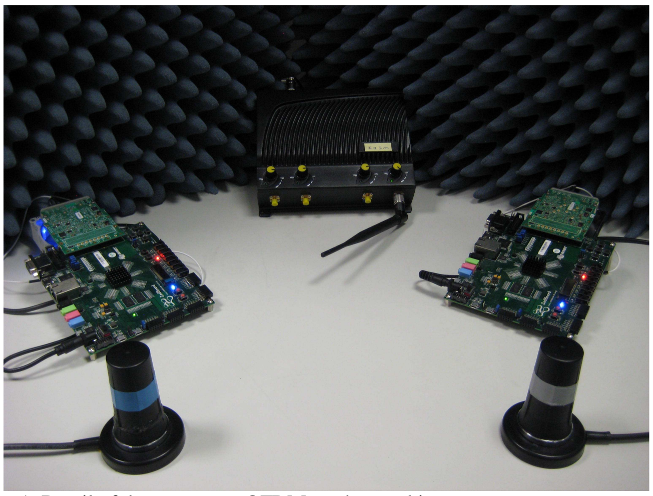


Fig. 1. Detail of the test setup: OFDM modem and jammer.

also presented, which demonstrate the capability of the system to guarantee deterministic data delivery time even in the presence of interference.

The remainder of the paper is organized as follows: related work is presented in Section II. In Section III the FPGA implementation of the OFDM physical layer, base of the presented wireless communication system, is described. The deterministic, real-time and cognitive MAC, main contribution of this paper, is explained in Section IV. In Section V, system performance results of the implementation are presented; all of them in terms of delay and system recovery time against interference. Finally, concluding remarks and future work are summarized in Section VI.

II. Related work

There are several wireless communication systems in the literature which aim to fulfill some of the aforementioned requirements of industrial applications or tactical communications. Unfortunately, at the best of authors' knowledge, they do not cover all of them in the way the presented communication system does. This section aims to gather and explain some of these wireless communication systems in order to contextualize and give value to the presented work.

In the field of customizable radio modems, the WARP project [2] aims to develop a scalable and extensible programmable wireless platform. It offers a real-time FPGA implementation of the IEEE 802.11 OFDM PHY and Distributed Coordination Function (DCF) MAC. Unfortunately, this project does not go further in the implementation of higher Open System Interconnection (OSI) layers. Besides, it does not provide at the moment the necessary hardware blocks so as to implement the time synchronization mechanisms needed for deterministic and real-time communications.

The MAC is one of the most important layers in a communication system for ensuring data reliability and time bounds. Several MACs have been proposed in the literature from both cognitive and non-cognitive approaches. P. Suriyachai, et al. present a MAC layer called GinMAC [3]. It is based on a TDMA and includes mechanisms to obtain the required end-to-end reliability and delay bounds. Unfortunately, during some tests in an industrial environment in Portugal, functional problems related with the presence of interference arose. W. Shen, et al. propose another relevant critical MAC: PriorityMAC [4]. Also based in a TDMA access, it handles four priorities for data to be sent: data for emergency safety actions, extremely critical control, critical control and periodic monitoring. Despite implementing mechanisms to prioritize a particular type of traffic, no time bound is ensured using this MAC. Besides, it does not have mechanisms to overcome the effects of interference. Among the cognitive radio approaches, which do take into account the presence of interference, the MAC presented by K. Kunert, et al. [5] is the most interesting. It is based on a TDMA scheme, and it reserves time within a frame to exchange control/spectrum sensing information. This MAC aims to select the most suitable frequency for the communication at any given moment. Unfortunately, it does not foresee the possibility of a destructive interference that blocks the complete transmission of data. Consequently, in case of interference, the network has to be formed again which leads to recovery times unsuitable for time-critical applications.

With regard to jamming-resistant wireless communication systems, several approaches have been presented. [6], [7] and [8] address the jamming problem from a signal processing point of view. They present several strategies, based on spread spectrum techniques, which do not try to avoid the interference but mitigate its effects. Unfortunately, none of the systems deal with real-time and deterministic data delivery time. Other approaches take advantage of cognitive radio capabilities. Q. Wang et al. [9] present a system which, unlike other Cognitive Radio implementations, assumes that malicious interference may not be predictable and may completely disrupt communication. On this basis, the proposed system implements an algorithm that continuously keeps exploring the best set of channels for transmission. Available channels are sorted based on both, spectrum sensing and a reward policy applied each time a successful communication is achieved in a particular channel. In case of a disruptive interference, the whole system switches to the next frequency band in the set. Although this algorithm is able to avoid interference, no further information is given with regard to system recovery time or medium access strategy, thus not being able to determine if the system ensures deterministic data delivery time and real-time characteristics.

Communications in tactical or industrial close-loop control applications require short latency, minimal jitter, deterministic data delivery time and high reliability, even in the presence of malicious interference. Therefore, our proposed wireless communication system implements a MAC layer that fulfills all these requirements as a whole, unlike previous systems that address them individually.

III. OFDM modem over FPGA

The physical layer of the proposed communication system is an OFDM transmitter and receiver implemented in FPGA

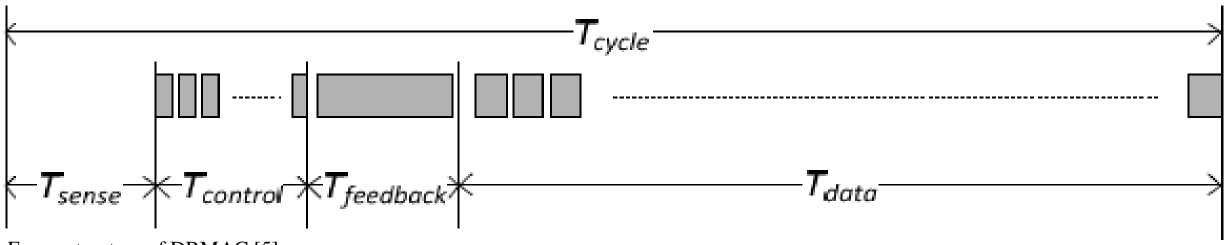


Fig. 2. Frame structure of DRMAC [5].

programmable logic. This type of implementation takes advantage of the parallelization possibilities of the FPGA, thus achieving low latency in the transmission and reception processes. Besides, the reconfigurable nature of FPGAs allows a complete customization of any of the signal processing algorithms present in the modem and the possibility of adding new features in case it is necessary. The modem has been designed, tested and implemented using the rapid prototyping tool named Xilinx System Generator.

The OFDM modem is compatible with the IEEE 802.11a/g standard at the physical level. It implements a 52 subcarrier OFDM scheme (48 data subcarriers and 4 pilot subcarriers) which can be modulated with BPSK, QPSK, 16-QAM or 64-QAM constellations. The modem in configurable to the following data rates: 6, 12, 24, 36 and 54 Mbits/s. These rates are high enough for the small amount of data that close-loop control applications demand (short messages of several bytes length).

Harnessing the aforementioned customization characteristics that FPGAs offer, and in order to enable cognitive features in the MAC layer, an energy detector (ED) has been included in the OFDM modem. The detector analyses the energy present in the current channel. If the energy is greater than a certain threshold, the ED decides that the sensing is under the hypothesis that a transmission is taking place in that channel. The performance of an ED is reduced under uncertainty noise, so an estimator of the noise power has been implemented. The noise information is estimated using recursive averaging.

The presented wireless communication system has been implemented in the Nutaq ZeptoSDR platform [10]. The ZeptoSDR consists of a ZedBoard, a low-cost development board for the Xilinx Zynq-7000 all programmable SoC, and a Radio420S module, a reconfigurable RF front-end which is able to select a frequency band between 300 MHz and 3 GHz. The OFDM modem has been implemented in the programmable logic of the Zynq SoC while the upper communication layers, i.e. the real-time, deterministic and cognitive MAC layer, have been software-implemented in the ARM core present in the Zynq SoC.

Despite the reconfigurable front-end is able to work in any frequency between 300 MHz and 3 GHz, the current implementation is limited to work only in 4 different frequencies: 2.4 GHz, 1.2 GHz, 1.1 GHz and 868 MHz. It should be noted that the communication system has been tested in an anechoic chamber, without interfering with the licensed bands of 1.2 and 1.1 GHz.

IV. Deterministic, Real-Time and Cognitive MAC

The proposed MAC is based on the MAC presented by K. Kunert, et al. [5] to which a novel handoff algorithm has been added. This improvement allows interference avoidance in a deterministic way. This MAC, also known as DRMAC, follows the general TDMA frame structure depicted in Fig. 2. At the beginning of the frame, each node senses the spectrum using the ED. During the control period, each node sends the spectrum sensing results to the coordinator of the network in its corresponding control slot. Besides, the coordinator sends an acknowledgement (ACK) to each node to confirm the reception of the ED results. Then, the coordinator generates a Sorted Channel List (SCL) using the channel information gathered by the nodes, and transmits it during the feedback period. In this list, the channels are sorted in function of its occupation, in a way that the least busy channel is the first channel in the list. If the occupation difference between the first and second channel exceeds a given threshold, the system changes its transmission band. Finally, a normal TDMA slot is used to transmit data traffic. Moreover, a frame start slot has been added at the beginning, to synchronize the network and a service slot at the end to send configuration packets.

In order to avoid a possible jammer, in our implementation the control period is used to monitor whether the channel is interfered or not as well as to generate the SCL. During this period, the coordinator receives in each control slot a packet from a node, while the node receives an ACK packet if the transmission was carried out correctly. Therefore, if the coordinator does not receive a packet, an error is considered. In the same way, nodes consider an error if no ACK packet is received. As a jammer causes bursty losses, it is considered that a jammer is interfering the band after a certain number of consecutive errors (N). When this number of consecutive errors is detected, every node in the networks hops to another band. The SCL is used to decide the band to hop, therefore, every node in the network hops to the same frequency band.

To characterize the real-time behavior of the MAC, a theoretical analysis using Network Calculus [11] has been carried out. This analysis establishes a theoretical delay bound considering the input traffic and the traffic which the network is capable of handling. A periodic input traffic and a maximum number of retransmissions M have been considered. Each packet is transmitted M times and, after this time, it is discarded because a transmission timeout is considered as an error in real-time applications. Therefore, the maximum delay d is obtained taking into account these conditions is

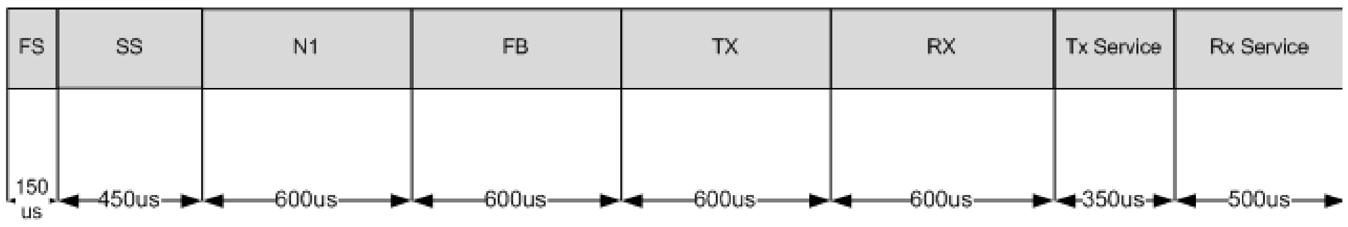


Fig. 3. Frame structure of the implemented MAC.

$$d \leq M \cdot c + t_{tx} \tag{1}$$

where M is the maximum number of retransmissions, c is the frame length and t_{tx} the transmission time which depends on the packet length and the bit rate. This bound will be ensured if the closed-loop control cycle time is higher than d.

The MAC layer and the wireless network have been modeled in the OPNET network simulator. A simple network composed by a coordinator and a node has been deployed. Note that the network design is scalable for several network devices in a star topology, but this example serves to represent the backbone link of a train communication network (link between two train consists). Concerning the TDMA scheduler configuration, the number of consecutive errors to hop to another channel has been set up to 3 and the maximum number of retransmissions of a packet to 4. This way, in case of the appearance of jamming interference in the transmission band, after the system hops to an interference free band, still a packet retransmission would be left, thus guaranteeing its deliver.

With this setup, the resulting TDMA frame structure and timing can be seen in Fig. 3, with a total frame length of 3850 us. Besides, with the OFDM modem configured with a data rate of 12 Mbps and sending 50 Byte messages, the maximum theoretical delay obtained with network calculus, even in the presence of a jamming interference in the transmission band, is

$$d \leq 4 \cdot 3850 + 600 \xrightarrow{yields} d \leq \ 16\,ms \tag{2}$$

The situation in which more than one band is being interfered has not been considered in the above analysis due to the life-cycle that data in closed-loop control application has. Packets arriving after two or more consecutive frequency hops would be discarded for this reason. However, it is interesting to know the time in which the system recovers after the appearance of interference. This analysis has also been carried out with network calculus. This recovery time *rt* is given by

$$rt \leq N \cdot l \cdot c + t_{tx} \tag{3}$$

where N is the numbers of bands that are simultaneously being interfered and l the number of consecutive communication errors that the network accepts after hopping to a new band.

For example, with the aforementioned setup, and considering that 2 out of the 4 available bands are being interfered, the maximum system recovery time is:

$$rt \leq 2 \cdot 3 \cdot 3850 + 600 \xrightarrow{yields} rt \leq \ 23.7\,ms \tag{4}$$

This theoretical delay bounds will be verified via simulations and measurements in real hardware in the next section.

V. Implementation, simulations and Measurements

This section presents the FPGA implementation results, and the simulations and measurements carried out that demonstrate the deterministic, real-time and jamming-resistant capacities of the presented wireless communication system.

A. *FPGA implementation*

Table I shows the FPGA logical resources consumed by the OFDM modem implementation. It can be seen how the modem fits in the available resources of the low-cost Xilinx XC7Z020 FPGA that has been used.

The physical layer implementation of the modem achieves a minimum input sensitivity level of -79 dBm, for a QPSK modulation per each OFDM subcarrier.

B. *Deterministic and real-time behaviour*

Fig. 4 shows the simulation and measurement results that demonstrate the deterministic behavior of the proposed system. The figures depict the probability Density Function (PDF) of the end-to-end time latency from the generation of a data packet to be transmitted (control cycle time of 30 ms) to its reception in the receiver. Two scenarios have been measured: the first scenario is an ideal one, with no interference, in which no packets are lost. Thus, no retransmissions happen. In the second one, two antennas have been plugged to the system and a controlled sporadic interference has been introduced in order

TABLE I: FPGA resource utilization

	SLICE	**REGISTER**	**LUT**	**DSP**	**BRAM 18 kbits**
Number	11612	38637	33805	180	140
Percentage	87%	36%	63%	81%	25%

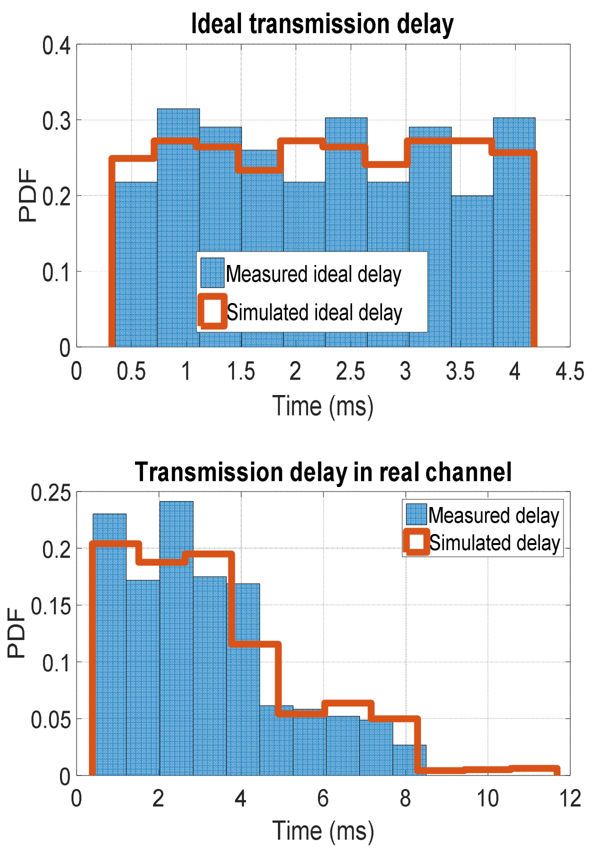


Fig. 4. Transmission delay simulations and measurements.

to generate packet losses and retransmissions. This interference could emulate the WiFi traffic environment that can be found in a train station. The aim of this test is to measure the deterministic behavior of the MAC against sporadic packet losses. Therefore, the handoff algorithm has been disabled so that the system does not hop to an interference free band where no retransmissions would happen.

As it can be seen in Fig. 4, in the ideal scenario, the transmission delay is bounded to the length of a single frame plus a small processing time. In the real scenario, as some packets are lost and retransmissions take place, the delay increases. Most of data packets are received during the first frame but some of them arrive during the second one after one retransmission. In any case, the system behaves in a real-time and deterministic way, fulfilling the control cycle time requirements and remaining below the theoretical maximum delay calculated with network calculus of 16 ms.

C. *Jamming-resistant behaviour*

Table II shows the time needed by the system to change its transmission/reception frequency. Due to the RF front-end characteristics, two different times are considered. "Channel change time" if the frequency hop is smaller than 50 MHz and "Band change time" if the frequency hop is greater. As it can be seen, the achieved frequency reconfiguration times are smaller than a single TDMA frame. Thus, it is possible to implement agile cognitive algorithms.

Fig. 5 presents the simulations and measurements carried out in order to test the behavior of the jamming-resistant feature. In the appearance of interference, the time needed to hop to a safe frequency band and recover the communication is measured. With this purpose, a commercial jammer (TX4CA from Projammers) has been added to the real scenario presented in the previous section. Three different jamming scenarios have been tested:

- *Scenario A*: Only the frequency in which the communication system is working is interfered. A singe frequency hop is forced. Under these circumstances it can be seen, both in simulations and in the measurements, how the recovery time is bounded between 8.5 and 12.5 ms. This time corresponds to the loss of 2 to 3 frames plus a slot of the next one. That is, the necessary time so as to lose 3 consecutive feedback messages or ACKs.
- *Scenario B*: The frequency in which the communication system is working and the next one in the SCL are interfered. Two frequency hops are forced. In this case, recovery time is bounded between 20 and 23.7 ms. This time corresponds to two consecutive recovery processes. That is, to the loss of 5 to 6 consecutive frames.
- *Scenario C*: The frequency in which the communication system is working and another band are interfered. Randomly, depending on the state of the SCL, one or two frequency hops are forced. In this last scenario, a mixture of the results of the above two scenarios can be observed. Taking into account that the state of the SCL table is not controlled in this test, after the first frequency change, the system can randomly hop into a free or interfered band. Therefore, system recovery times are distributed between this two possibilities, achieving the same time boundaries of the aforementioned two scenarios. It should be noted that in the simulations, the SCL status is completely random. Therefore, after the first frequency change, the probability of hopping to the second interfered band is 1/3, as can be seen in Fig. 5. However, in the real scenario, the SCL is updated with the spectrum sensing data, thus penalizing the interfered band. Accordingly, the probability of hopping to this band gets reduced.

It should be noted that the obtained maximum system recovery times match the maximum time of 23.7 ms estimated via network calculus theoretical analysis.

VI. CONCLUSIONS

TABLE II: FREQUENCY CHANGE TIME

	Change Time
Band change	191 us
Channel change	50 us

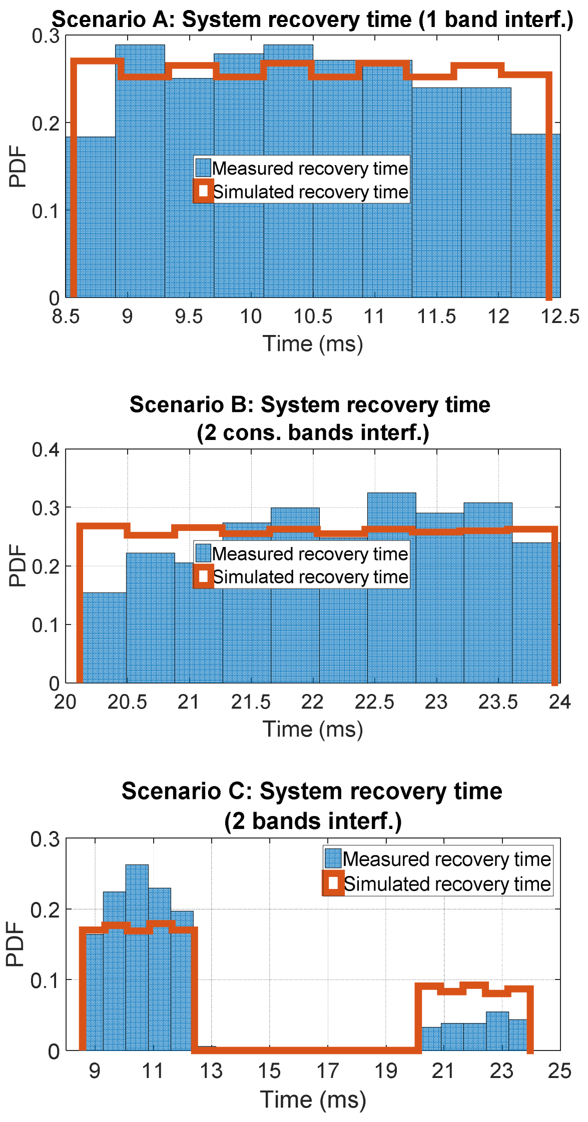


Fig. 5. System recovery time simulations and measurements.

In this paper a jamming-resistant and deterministic wireless communication system has been presented. A custom implementation of an OFDM modem design has been carried out into the low-cost Xilinx XC7Z020 FPGA. The proposed MAC layer, which includes a novel spectrum handoff algorithm for interference avoidance, has proven to behave in a real-time and deterministic way, achieving bonded data delivery times. Besides, the cognitive features added to the MAC, provide the system with jamming-resistant features that allow the wireless communication system to recover from a malicious interference in a short and deterministic time. All these features allow the presented system to be used in time-critical and mission critical industrial or tactical applications, in which data delivery has to be bonded in the reliability and time domains. With regard to future work, the improvement of the spectrum sensing algorithm is planned. The integration of cyclostationary signal detectors would allow the system to distinguish different interference types (users and jammers) and act accordingly.

ACKNOWLEDGMENT

This work was partly supported by TIPOTRANS (ETORTEK) and CIIRCOS (PC2013-68) projects of the Basque Government (Spain), and by COWITRACC (TEC2014-59490-C2-1-P/2-P) project of the Spanish Ministry of Economy and Competitiveness.

REFERENCES

[1] Rodriguez, P., Torrego, R., Casado, F., Fernandez, Z., Mendicute, M., Arriola, A., Val, I.: 'Wideband cognitive wireless communication system: implementation of an RF-Ethernet bridge for control applications'. Proc. Wireless Innovation Forum European Conference on Communication Technologies and Software Defined Radio (SDR'14 – WInnComm – Europe), Rome (Italy), Year 2014

[2] https://warpproject.org/, accessed 2016

[3] P. Suriyachai, et al., "Time-critical data delivery in wireless sensor networks," in Distributed Computing in Sensor Systems, 2010, pp. 216-229

[4] W. Shen, et al., "PriorityMAC: A priority-enhanced MAC protocol for critical traffic in industrial wireless sensor and actuator networks," IEEE Transactions on Industrial Informatics, vol. 10, pp. 824-835, 2014

[5] K. Kunert, et al., "Deterministic real-time medium access for cognitive industrial radio networks," in IEEE International Workshop on Factory Communication Systems - Proceedings, WFCS , art. no. 6242549 , pp. 91-94, 2012.

[6] Q. Dong, et al., "Adaptive Jamming-Resistant Broadcast Systems with Partial Channel Sharing" in IEEE 30th International Conference on Distributed Computing Systems (ICDCS), 2010

[7] M. Liechti et al., "Jamming Mitigation by Randomized Bandwidth Hopping" in The 11th International Conference on emerging Networking EXperiments and Technologies (ACM CoNEXT 2015), 2015

[8] C. Pöpper et al., "Jamming-resistant Broadcast Communication without Shared Keys" in Proceedings of the 18th conference on USENIX security symposium (SSYM'09), 2009

[9] Q. Wang et al., "Anti-jamming Communication in Cognitive Radio Networks with Unknown Channel Statistics" in 19th IEEE International Conference on Network Protocols (ICNP), 2011

[10] http://www.nutaq.com/blog/zeptosdr-architecture-and-api, accessed 2017

[11] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Berlin, Heidelberg: Springer-Verlag, 2001.

Reconfigurable Antenna based System for Spectrum Monitoring and Radio Direction Finding

Hassan El-Sallabi, Abdulaziz Aldosari, and Saad Alkaabi
Emri Signal and Information Technology Corps
Qatar Armed Forces
Doha, Qatar

Abstract

A reconfigurable antenna based system for spectrum monitoring and radio direction finding is proposed. The proposed system provides lower complexity and smaller physical size compared to existing direction finding systems based on phase difference of multiple antenna elements or systems based on mechanically rotating antenna. Instead of using multi-element antenna arrays or rotatable directional antenna, we use reconfigurable antenna that has the capability to reconfigure its characteristics to match frequency and minimize polarization loss of incoming signals. The other main feature of the reconfigurable radio direction finder is the use of multiple directional antenna pattern states with multiple pointing directions that cover the azimuthal 360 degrees.

I. Introduction

Spectrum range dedicated for radio communications is scarce. Communications industry worldwide and new wireless services offered in market increased the demand for the RF spectrum. These demands have a major consequences on frequency allocations to meet no electromagnetic interference requirements. Spectrum management is helps to maintain interference free environment. Part of spectrum management process is spectrum monitoring to ensure proper operation of radio communication systems without harmful interference as required by spectrum management activities. Spectrum monitoring has the following the goals: 1) find out any interference on local, regional and global scale; 2) ensuring acceptable quality in operating systems; 3) provides information on actual use of spectrum bands; 4) provide measurement based inputs to programs organized by ITU to eliminate harmful interference. Hence, it is a process of inspection and meeting compliance set my spectrum management authority that enables to identification of accidental or intentional interference sources, verification of proper characteristics in terms of transmission power etc of radiated signals in addition to identification of illegal transmitters. Spectrum planners use reports from spectrum monitoring to understand level of spectrum usage compared to assignments and provides actual information on meeting the set polices [1]. Radio direction finder (RDF) [2] is a part of the monitoring system. It can operate in fixed, mobile and homing modes. RDF is needed to locate and position sources of interferences and illegal transmitters. RDF system has a vast range of applications in military and civilian commercial applications. These include emergency aid, which is usually deployed on civil aircraft, remotely guiding navigation system, locating and tracking interference sources (both accidental and malicious), locating illegal radio frequency transmitters, jammers, etc. It would have applications in search and rescue operations, when RF signals are transmitted. The system can be adopted for ships, small boats and aircrafts to find directions of defined beacon signals.

In this work, we propose a spectrum monitoring and RDF system based on reconfigurable antennas. The main advantage of the proposed system lies in providing lower complexity and smaller physical size compared to existing techniques based on phase difference of multiple antenna elements or systems based on mechanically rotating antenna.

II. Reconfigurable Antennas

Antennas have usually been considered to have a fixed radiation pattern at a specific operating frequency. Reconfigurable antennas [3] have introduced in literature as they have the capabilities to dynamically change their characteristics such as radiation pattern, polarization, and operating frequency. Reconfigurability aspect of antennas can be achieved via different techniques such as altering physical structure of the antenna, altering feeding methods, controlling current density, etc. The design requirement and performance level decides the reconfigurability method. The distribution of current in antenna and its geometry determines how the antenna radiates its energy into the radio channel, or how it captures energy from it. The complex far field radiation pattern of an antenna can be mathematically be expressed as [4]

$$\mathbf{F}(\theta,\phi) = \hat{\mathbf{r}} \times \left[\hat{\mathbf{r}} \times \left[\int_{\mathrm{v}'} \mathbf{J}_{\mathrm{v}'}(\mathbf{r}') e^{-j\beta\hat{\mathbf{r}}\mathbf{r}'} dv \right] \right]$$

where $\mathbf{F}(\theta,\phi) = \left(\mathrm{F}_\theta(\theta,\phi), \mathrm{F}_\phi(\theta,\phi)\right) \in C^{2\times 1}$, $\mathbf{J}_{\mathrm{v}'}(\mathbf{r}')$ is the current distribution in the antenna, $\hat{\mathbf{r}}$ is the unit vector in direction of propagation to observation point, $\mathbf{r}'$ is a vector from coordinate system's origin to any point on the antenna. From the above equation, it is clear that by changing the antennas' physical configuration, $\mathbf{r}'$, then the current distribution $\mathbf{J}_{\mathrm{v}'}(\mathbf{r}')$ changes, which is reflected in altering the complex far field radiation pattern $\mathbf{F}(\theta,\phi)$. Hence, controlling distribution of current in antenna $\mathbf{J}_{\mathrm{v}'}(\mathbf{r}')$ leads to controlling the radiation pattern $\mathbf{F}(\theta,\phi)$. This current control process can be achieved by using micro-electromechanical system switches (MEMS) or active switches such as field effect transistor (FET) or diodes. The distribution of current in an antenna can be controlled by placing these switches in strategic locations to control current paths to generate a specific antenna characteristics. Implementation of reconfigurable antennas can be accomplished in three different processes; which 1) Design Stage, 2) Simulation Stage, 3) Optimization Stage [5]. Antenna design stage include selection of radiating structure of the antenna and reconfigurable aspects. Selection process is based on several performance parameters such as power consumption, directivity, bandwidth operating frequency, etc, in addition to design constraints such as antenna size, fabrication, cost, etc. These antenna performance parameters have to be fulfilled for every antenna state to make sure that the reconfigurable antenna has to work optimally as expected when it switches from antenna state to other. After selection of antenna structure, it is important to select reconfigurablility function and how it takes place. Antenna reconfigurability can be categorized into 4 different reconfigurability functions; which are 1) Reconfiguring resonance frequency [7], which usually takes place by changing physical planar that alters surface current distribution, 2) Reconfiguring radiation pattern [7,8], which usually takes place by changing radiating edges, slots, or feeding network, 3) Reconfiguring polarization state, which usually takes place via changing surface structure or the feeding network, 4) Combinations of reconfiguring the above listed characteristics, which takes place by using some of the aforementioned different techniques simultaneously. It is very difficult to configure frequency, radiation pattern and polarization independent of each other. Changing one characteristic will change others as a consequence. Therefore, careful design and analysis are important. Due to lack of space, the Simulation Stage and Optimization Stage are not discussed here.

III. Proposed System

The system would have a wideband omni-directional antenna to detect energy in electromagnetic spectrum of interest. The frequency and bandwidth of signal of interest are used as information to configure the resonance frequency of reconfigurable antenna. Then, the system switches to next mode of operation. The core idea in reconfigurable antennas is to control changing current distribution around the antenna, which leads to altering the radiated far field. These changes can be achieved by modifying antenna geometry or its material properties. The switching process between different antenna modes via different activated and deactivated switches distinguishes reconfigurable antennas from phased array antennas. Different antenna patterns with different pointing directions are configured to detect arrived radio energy and weight them differently based on different pointing directions. The other main feature of the reconfigurable radio direction finder is the use of multiple directional antenna pattern states with multiple pointing directions that cover the azimuthal 360 degrees. The re-configurability of pointing direction of antenna patterns may also take place in polar plane to find elevation angle of direction of radio signal. These multiple pointing directions of antenna pattern states have a predefined reference to determine the direction of incoming signal through processing in signal processing unit of the system. Reconfigurable antenna with multiple antenna pattern states and configurability of beam-width of antenna patterns improve the accuracy of the estimated direction that make the system self-dynamic with the environment and direction of arrival. These weighted energy power received from every antenna stated are used with their corresponding pointing direction to estimate direction of arrival of the signal of interest.

The proposed system works on three mode stages. The first mode is to detect spectrum energy and its bandwidth via wideband antenna detection system. The detected signal of interest is defined initially in terms of its frequency range, bandwidth and polarization. This information is fed to the reconfiguring unit of the system, which starts the second stage of mode of operation whereby, the reconfigurable antenna direction finder system dynamically reconfigure the reconfigurable antenna to resonate its operating frequency on the frequency and bandwidth of signal of interest and then reconfigure multiple-antenna patterns for best detection at the polarization state. The directional multiple antenna patterns have different pointing directions relative to a pre-defined reference direction. The signals from different antenna states are processed in a processing unit to estimate the direction of arrival based on their detected signal power and pointing direction of multiple directional antennas pattern states. The third stage of mode of operation is to reconfigure beam-widths [9] of the reconfigurable antennas pointing close to the estimated direction in stage of operation mode. This last stage of re-configurability refines the estimated direction by using estimated signals from refined third stage antenna patterns and their pointing direction. The more the multiple antenna patterns with narrower beam-width, the more accurate the system achieves better estimation the direction of arrival. Block diagram of the proposed system is shown in Figure 1.

The proposed system is based on estimating the direction of arrival of RF signal to the reconfigurable antenna at the receiver. Instead of using multi-element antenna arrays or rotatable directional antenna, we use reconfigurable antenna that has the capability to reconfigure its antenna characteristics, such as radiation pattern, pointing direction, resonance frequency, and polarization to match frequency and minimize polarization loss of incoming signals.

IV. Direction Finding Algorithm

The direction of arrival of signal propagation from a radio source can be defined by two angles; the azimuth and elevation angles. Our approach can be adopted in systems that use reconfigurable antennas that can use multiple antenna patterns sequentially. These antenna patterns are called antenna states. The reconfigurable antenna systems divides the angular range to *N* angular sectors. Each angular sector is centered with a pointing direction of a particular antenna state. The control unit sequentially sends different weights for optimum antenna states, their corresponding radiation patterns scan particular spherical sector. The principal of algorithm operation is shown in Figure 1. The control unit sends ON and OFF commands to switches that correspond to antenna state *k*. It stays there for a while and radio receiver makes received signal strength (RSS) measurements. Then, it sequentially sends other set of ON and OFF commands to the switches that correspond to other antenna state (say *k*+1), and stays there for a short while and make measurements. This process continues until it goes through all antenna states and start the operation again. For illustration as shown in Figure 2, based on direction of incoming signal, the highest signal level will be what is received from antenna state 1 and then, the next level is what is received from antenna state 4. Measurement data from antenna state 2 and 3, will be minimum. Each antenna state interact with electric field of the incoming signal differently due to their different antenna characteristics. The receiver measures the RSS of the incoming signal with every antenna state. The RSS is related to time dependent complex signal, $V_{oc}(t)$, which is given by the voltage induced at the local port of the antenna, which can be formulated as

$$V_{oc}(t) = \int \Im(\Omega) \cdot E_i(\Omega)\, e^{-jk(r_i - \mathbf{V}\cdot\Psi_i(\Omega))} d\,\Omega$$

where $E_i(\Omega)$ is the electric field of the plane wave incident of DF antenna from direction of solid angle $\Omega = (\theta, \phi)$, θ is elevation angle, ϕ is azimuthal angle, $\Im(\Omega)$ is the far field amplitude of the antenna pattern, k is the wave number (i.e., $k = \frac{2\pi}{\lambda}$), $\Psi_i(\Omega)$ is for the arrival direction vector defined for incident ray for Cartesian coordinate as follows

$$\boldsymbol{\Psi_i} = \cos(\phi_i)\sin(\theta_i)\,\vec{x} + \sin(\phi_i)\sin(\theta_i)\,\vec{y} + \cos(\theta_i)\,\vec{z}$$

$\mathbf{V}$ is the velocity (speed and direction) of the RF source terminal, which is assumed as the receiver in this notation, and defined by

$$\mathbf{V} = v_x\vec{x} + v_y\vec{y} + v_z\vec{z}$$

The DF algorithm is based on reconfigurable antenna that can periodically switch pointing directions of its directional antenna pattern ($\Im_n(\Omega)$) of antenna state n that cover section of spherical angular space. There are *N* different antenna states that each have its own pointing direction that can be labeled as $\Im_n^{(\theta_n,\phi_n)}(\Omega)$, where $\Im_n^{(\theta_n,\phi_n)}(\Omega)$ is the far field amplitude of antenna patter of state *n*, whose point direction is (θ_n, ϕ_n). The DF receiver makes several measurements for every antenna state and then switches to the next and so on. Then, it repeats the cycle. The measurement vector of all *N* antenna states can be written as

$$M = [V_1\ V_2\ V_3\ \ldots.V_N\,]'$$

where V_N is the average of measurement time series of amplitude of signal measured with DF receiver while its antenna state is $\Im_n^{(\theta_n,\phi_n)}(\Omega)$. In order to estimate the direction or arrival of signal of interest, the measurement vector

M is processed while considering spherical properties of the data. Spherical properties invokes spherical statics, which is different from linear statistics. Spherical statics is also called directional statistics. Directional statistics [6] are concerned mainly with observations of unit vectors in a plane or three dimensional space to cover circle or sphere spaces, respectively. In this application, we have *N* directional measurements individuals. At a particular time, we have measurement $V_n(t)$ and associated with a unit direction vector that correspond to pointing direction of this particular antenna state *n*. This unit vector can be written as

$$\Upsilon_n = [\cos(\phi_n)\sin(\theta_n) \qquad \sin(\phi_n)\sin(\theta_n) \qquad \cos(\theta_n)\ ]'$$

The measured data that correspond to each antenna state can be combined with the known its pointing direction as follows

$$\chi_n = V_n \Upsilon_n$$

The χ_n vector may be described by direction cosines, which represent its components along the three coordinate axis, x-, y-, and z-axes. The χ_n vector contains the amount of projection of measured amplitude in the three principal axes The components of all $\chi_1, \chi_2, \ldots, \chi_z$ on each axis $(\mathrm{X}, \mathrm{Y}, \mathrm{Z})$ in Cartesian coordinates combined together allows us to compute the centroid of measured data along each principal axis. The center of gravity of set of projected measurement data has their mean of the x coordinate, mean of y coordinate, and mean of z coordinates as follows:

$$\mathrm{X}_{DF} = \frac{1}{N}\sum_{n=1}^{N} V_n \cos(\phi_n)\sin(\theta_n)$$

$$\mathrm{Y}_{DF} = \frac{1}{N}\sum_{n=1}^{N} V_n \sin(\phi_n)\sin(\theta_n)$$

$$\mathrm{Z}_{DF} = \frac{1}{N}\sum_{n=1}^{N} V_n \sin(\theta_n)$$

Then, we can estimate both the azimuthal and elevation angles of direction of arrival of signal of interest via the direction of the centroid of the measured data with multiple antenna states of reconfigurable antenna. The elevation angle (Ξ_{DF}) and azimuthal angle (Φ_{DF}) of signal of interest a can be estimated via conversion of Cartesian coordinates to spherical coordinates as follows:

$$\Xi_{DF} = \cos^{-1}\left(\frac{\mathrm{Z}_{DF}}{\sqrt{X_{DF}^2 + Y_{DF}^2 + Z_{DF}^2}}\right)$$

$$\Phi_{DF} = \tan^{-1}\left(\frac{\mathrm{Y}_{DF}}{\mathrm{X}_{DF}}\right)$$

These angles are based on conventional spherical coordinate. The range of elevation angle is from 0 to π and range of azimuth angle is from 0 to 2π. The resultant length that comes out from the three centroids in x, y, z-axis is related to dispersion of the measured data estimate. The mean resultant length (*R*) can be computed from its direction cosines as follows:

$$\overline{R} = \sqrt{X_{DF}^2 + Y_{DF}^2 + Z_{DF}^2}$$

The spherical variance of the estimated directional centroids can be computed with

$$\Lambda_{DF} = 1 - \overline{R}$$

The value of Λ_{DF} is always between zero and one inclusive. In order to increase the accuracy of the DF system, the system can be figured into two levels of DF estimation. In the first level, the Z_1 antennas states have wide beam-width that can determine the direction in coarse fashion, see Figure 2. The second level, the control unit of the reconfigurable antenna DF system send weights that optimize Z_2 antenna states that have narrow beam-widths that allows determining the direction more precisely than in first level antenna states, see Figure 3. The two levels approach takes longer processing time but it DF estimation is higher.

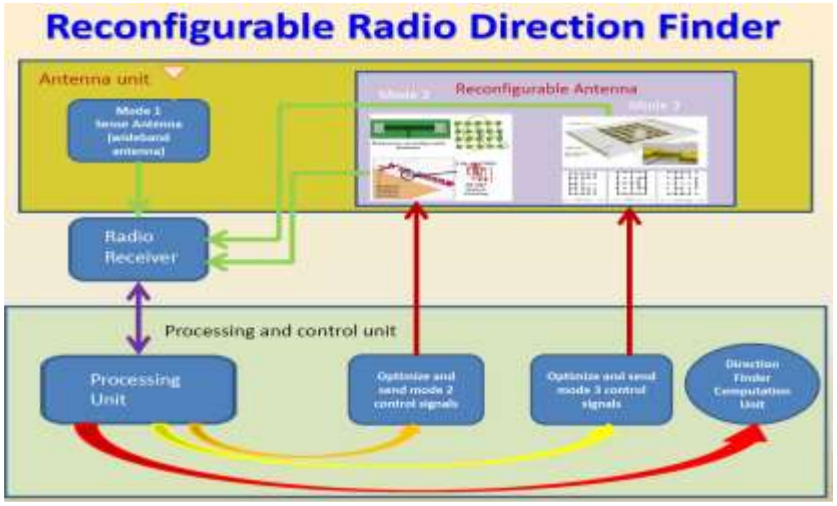


Figure 1. Block diagram of the proposed Reconfigurable Radio Direction Finder System.

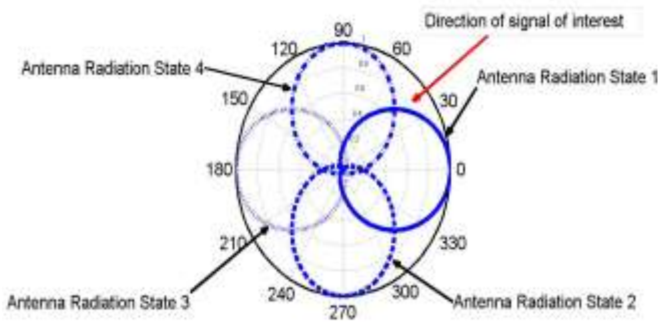


Figure 2. Sample of four antenna states pointing in different direction with a possible direction of incoming signal.

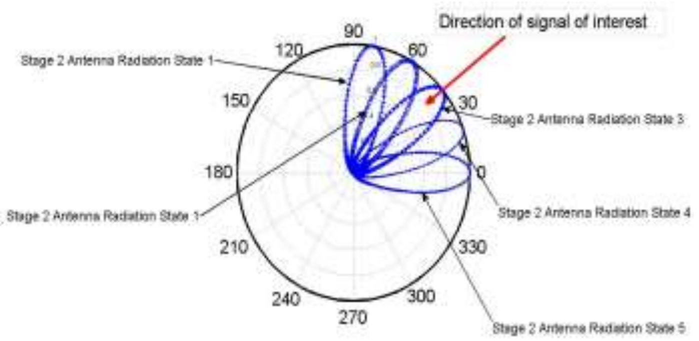


Figure 3. Sample of five narrow beamwdith antenna states pointing in different directions with a possible direction of incoming signal.

Conclusion

This work proposed spectrum monitoring and direction finder system based on reconfigurable antenna. The reconfigurable antenna can be reconfigured to work as wideband antenna for spectrum monitoring. Then, it can be reconfigured to a narrowband antenna to match a particular frequency of interest. The configurability also includes directional radiation pattern states, which each antenna state has a particular pointing direction. The system cycles all antenna states to scan the angular domain and find the RSS for every antenna state. This information is used to estimate the direction of arrival of signal of interest.

References

[1] International Telecommunication Union (ITU, *Spectrum Monitoring Handbook Radiocommunication Handbook*. Switzerland, 2011.
[2] ITU Radio Regulations, Section IV. Radio Stations and Systems – Article 1.91, definition: *radio direction-finding station*
[3] J.T. Bernhard. (2007). *Reconfigurable Antennas*. Morgan & Claypool Publishers
[4] Constantine A. Balanis. Modern Antenna Handbook. Wiley-Interscience, New York, NY, USA, 2008.
[5] D. Langoni, M. H. Weatherspoon, E. Ogunti, S. Y. Foo, "An overview of reconfigurable antennas: Design simulation optimization", *Proc. IEEE 10th Annu. Wireless Microw. Technol. Conf.*, pp. 1-5.
[6] Fisher, NI., *Statistical Analysis of Circular Data*, Cambridge University Press, 1993.
[7] Majid, H. A., M. K. A. Rahim, M. R. Hamid, and M. F. Ismail, "Frequency reconfigurable microstrip patch slot antenna with directional radiation pattern," Progress In Electromagnetics Research, Vol. 144, 319–328, 2014
[8] Aboufoul, T.; Chen, X.; Parini, C.; Alomainy, A. (2014). "Multiple-parameter reconfiguration in a single planar ultra-wideband antenna for advanced wireless communication systems". *IET Microwaves, Antennas & Propagation*. 8 (11).
[9] A. Khidre, F. Yang, and A. Z. Elsherbeni, "Reconfigurable microstrip antenna with tunable radiation beamwidth," in Proceedings of the IEEE Antennas and Propagation Society International Symposium (APSURSI '13), pp. 1444–1445, Orlando, Fla, USA, July 2013.

Proceedings of WInnComm Europe 2017, Copyright © 2017 Wireless Innovation Forum All Rights Reserved

Experimenting with Flexible D2D Communications in Current and Future LTE networks: A D2D Radio Technology Primer & Software Modem Implementation

A. Gotsis, K. Maliatsos, P. Vasileiou, S.Stefanatos
Feron Technologies P.C.
Contact Author: antonis.gotsis@feron-tech.com

M. Poulakis, A. Alexiou
Department of Digital Systems, University of Piraeus
Contact Author: mpoulak@unipi.gr

***Abstract*[1]—The current work aims at providing a comprehensive analysis of recent and ongoing activities on enabling device-to-device communications within traditional LTE radio access networks, as well as proposing a new software library and a software modem prototype for 3GPP D2D implementation and experimentation. A systematic investigation of the D2D-related 3GPP specifications is first performed and the key radio protocol implementation aspects are identified and extensively discussed. Focus is on D2D-specific physical layer processing and radio resource allocation peculiarities. The detailed D2D radio protocol description is followed by a brief presentation of two relevant software/hardware products developed within the context of EU-funded project FLEX-D. The first is an open-source software library developed in MATLAB that implements almost all the functionalities of the D2D radio protocol up to 3GPP release 14, including latest V2X specifications. The second is a real-time standard-compliant software modem prototype running on general purpose processor hosts and interfacing with SDR boards. Both contributions may aid the R&D community in understanding the performance limitations of D2D radio protocol and providing the foundations for creating end-to-end D2D solutions for vertical markets, such as public safety and vehicular communications.**

I. INTRODUCTION

Device-to-device communications or simply D2D allow for two user equipment devices (UEs) to communicate directly, hence allowing single-hop communication instead of the conventional two-hop cellular communication, where the base station is always the one end of each communication pair. D2D communications is a disruptive paradigm in the cellular world, going hand-in-hand with LTE evolution and 5G rise in the context of LTE-A, LTE-A Pro, and NR radio technologies [1],[2]. The D2D system concept within the 3GPP standardization body has both an evolutionary flavor as it benefits the performance of existing services, by leveraging latency reduction and traffic offloading features, and a revolutionary flavor as it enables new applications, such as public safety, social networking apps, vehicular communications, and wearables/IoT. The current work has a two-fold scope: to provide a comprehensive summary of the recent outcomes and ongoing activities around D2D support in 4G/5G, as well as present a D2D protocol software library and a software modem implementation based on the host-SDR model.

In the first part of the paper, we provide a concise technology introduction to the 3GPP D2D radio technology, as it has evolved from its first appearance in the 3GPP standard Release 12, its minor enhancement in Release 13, its V2X version planned to appear in the upcoming Release 14, and the application to wearables envisioned for Release 15. In particular, we will focus on the defined D2D operation modes as well as on the system information/timing reference transmission/acquisition procedure which is necessary for setting up communication at layer 1 (L1). For each mode, we will present the key D2D radio technology challenges and aspects, including the related signals, channels and procedures. We will also cover the D2D-specific radio resources allocation introduced in the standard for managing intra-D2D and inter D2D-LTE operation. The latter is necessary, since D2D has been decided to operate within the underlying LTE network, specifically reuse part of cellular uplink (UL) resources. We will also report recent D2D extensions in order to cater for the emerging V2X paradigm.

In the second part of the paper we describe our ongoing activities on developing an open-source software library implementing the D2D radio protocol standard and a real-time software modem using SDR. First versions of the library have been already published in Github repository. A first modem prototype is also available, comprising: i) a transceiver implementation of the basic 3GPP D2D radio functionalities running in general purpose processor (GPP) based hosts; ii) an interface with a USRP board for over-the-air signal transmission/reception; iii) a basic interface with higher layers for enabling end-user applications. We describe the software modem architecture along with a detailing of the basic building blocks. The radio transceiver components development is based on the in-house open software library. We also provide an evaluation of the real-time capabilities of the prototype modem, based on extensive benchmarking carried out in various typical GPP hosts, and conclude with a set of over-the-air link-level experiments performend in a EU Future Internet Research Experimentation (FIRE) platform.

The remaining of the paper is structured as follows. Section II provides a tutorial-style introduction to the LTE D2D radio protocol, while Section III includes a brief overview of our software library implementing the respective transceiver functionalities. Section IV builds on previous Sections and reports the D2D real-time software modem prototype development aspects, in particular, the architecture specification, design, implementation, runtime benchmarking and over-the-air link-level evaluation. Finally, Section IV.E summarizes the key contributions and presents an outlook of the work.

[1] The content of this contribution is largely based on material also included in the EU-funded public project deliverable: "FLEX D5.24 (FLEX-D D1), FLEX-D Innovations & Experimental Activities: Final Report", which will be shortly available in http://www.flex-project.eu.

II. A 3GPP D2D Radio Primer

A. *Background – Standardization*

The provided material is by no means exhaustive. Instead it covers issues related to intra-D2D and inter D2D/legacy control signaling (both L1 and higher-layer), as well as D2D L1 processing specifications. These are the fundamental pillars for designing and implementing a D2D software library as well as a real-time software modem. Issues related to the Core network or to the end-user application (e.g. security) are beyond the scope of the current paper.

3GPP started considering the support of direct communication among devices in the context of LTE Release 12 [3], [5], under the code name "ProSe" (proximity services). Although at an initial stage ProSe was centered around public safety use-cases, soon other commercial services, namely user-oriented (social applications) and network-oriented (offloading) as well as vertical ones (vehicle-to-vehicle/infrastructure communications) emerged. Initial work conducted within 3GPP resulted in a feasibility report published in 2013 [6], which highlighted the requirements for introducing communication among proximal UEs under LTE coverage or in the absence of it (public safety). In the following two years, 2013 and 2014, the D2D radio aspects were extensively studied in a series of 3GPP RAN1 meetings. A high-level presentation of the envisioned D2D radio architecture, technologies and protocols, summarizing the main outcomes of the corresponding RAN1 work, was published in [7] and [8], in 2014. **The term "sidelink" (SL) has been introduced for explicitly referring to the direct communication link enabled by D2D** (and to differentiate direct-access from typical cellular access), and since then it is considered an integral part of evolving 3GPP releases, together with regular uplink and downlink.

First D2D radio technical specifications (Physical Channels and Modulation, Multiplexing and Channel Coding, MAC, Physical-Layer Procedures, and RRC) appeared in the 3GPP standards, version 12.5.0 in June 2015. Sidelink defines the procedures for realizing a single-hop UE-UE communication, similarly to Uplink and Downlink which define the procedures for UE-BS and BS-UE access respectively. Along the same lines PC5 was introduced as the new direct UE interface, similarly to the Uu (UE-BS/BS-UE) interface. Sidelink enhancements in Release 13, published in March 2016, focused in network coverage extension based on L3 relaying provided by a ProSe-enabled UE, the latter supporting both cellular and direct-access connectivity. In June 2017, Release 14 is expected to be completed, and will include a set of sidelink enhancements for supporting V2X communications[2]. Finally, in the context of working Release 15, which is expected to freeze by September 2018, further D2D enhancements related to UE-to-network relays for IoT and wearables, are investigated [9].

B. *Radio Protocol Overview*

The D2D radio protocol design has been impacted by two important decisions:

- **Sidelink has been decided to operate in the same resources used for cellular LTE access**, and in particular in the LTE uplink spectrum. In this sense some kind of coordination between sidelink and regular uplink communication should be employed to avoid harmful interference on any of both sides. Mechanisms for network-controlled D2D access have been devised, at least when this is possible (i.e. D2D UEs are within full or partial coverage of an LTE eNB). These are primarily based on L3 signaling (RRC and SIB18/19 configuration messages) and secondarily on L1 signaling (DCI Format 5/5A messages).
- **Sidelink was decided to operate using new** physical **signals** (synchronization preambles and channel estimation pilots), control signaling/data physical and transport **channels**, and MAC channels/**structures**. Although distinct from typical cellular-access signals, channels and structures, their design is heavily based on uplink/downlink design.

3GPP has defined two D2D operation modes:

- **Discovery**, where D2D UEs announce their presence to other proximal UEs, sending a very short data message using a light (in essence L1) protocol stack. The corresponding "inverse" functionalities are also defined, which specify how a D2D UE could monitor and recover announcements sent by proximal UEs.
- **Communication**, facilitating typical real-time and non-real time applications (e.g. VoIP, on-demand video-streaming, V2V communication applications), where D2D UEs send/receive data to/from proximal UEs, using a complete protocol stack, including L1, L2, L3 and above.

A procedure including timing and system-information acquisition is also triggered by both modes. This is necessary for acquiring L1 synchronization at the receiving D2D UEs' side as well as inform partial/out-of-coverage D2D UEs about L1 system configuration (bandwidth mode, sidelink physical layer identity, sidelink subframe/frame numbering)[3]. Towards this purpose, a protocol for D2D-specific timing synchronization and system information acquisition is also defined. For in-coverage D2D operation, where both transmitting and receiving UEs reside in the same cell, time synchronization is provided by the legacy LTE cell and there is no need to perform D2D-specific synchronization. However, there are several scenarios where a D2D-specific procedure is necessary: (i) in multi-cell in-coverage, where the receiving D2D UE resides in a different asynchronous cell with respect to the transmitting D2D UE; (ii) in partial-coverage, where the receiving D2D UE is out of coverage and needs to acquire synchronization from the in-coverage transmitting D2D UE; (iii) out of coverage, where both UEs are outside the coverage of a legacy LTE cell and the transmitting UE acts as a reference synchronization source.

[2] http://www.3gpp.org/news-events/3gpp-news/1798-v2x_r14

[3] In this work we implicitly consider timing and system-information acquisition to be a "standalone" mode as well for a baseline D2D link.

Sidelink specifications for L1, L2, L3, including the LTE eNB control signaling may be found in the following documents:

- 36.211 Physical channels and modulation (Section 9)
- 36.212 Multiplexing and channel coding (Section 5.4)
- 36.213 Physical layer procedures (Sections 5.2.2.25, 5.2.2.26, 5.10, 14)
- 36.321 Medium Access Control (MAC) Protocol specification (Sections 5.14, 5.15, 5.16)
- 36.331 Radio Resource Control (RRC); Protocol specification (Sections 6.5.2, 6.3.8)

C. Radio Protocol Detailed Description

The sidelink physical layer is actually a tweaked version of the conventional LTE UL/DL specifications. Relevant transport channels, physical channels, control-signaling, and physical signals/sequences are introduced. The structure of the SL processing (in particular the transmitter side) is illustrated in Figure 1.

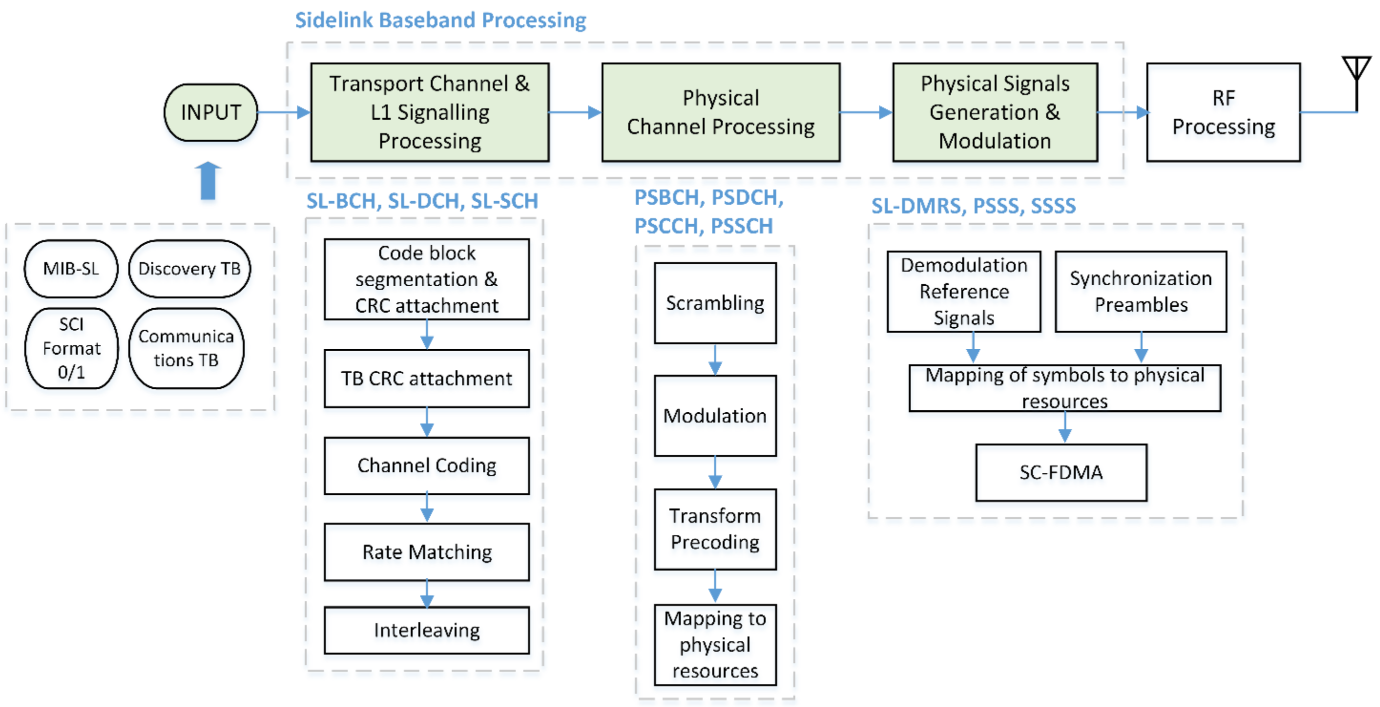

Sidelink Baseband Processing
INPUT
Transport Channel & L1 Signalling Processing
Physical Channel Processing
Physical Signals Generation & Modulation
RF Processing
MIB-SL
Discovery TB
SCI Format 0/1
Communications TB
SL-BCH, SL-DCH, SL-SCH
Code block segmentation & CRC attachment
TB CRC attachment
Channel Coding
Rate Matching
Interleaving
PSBCH, PSDCH, PSCCH, PSSCH
Scrambling
Modulation
Transform Precoding
Mapping to physical resources
SL-DMRS, PSSS, SSSS
Demodulation Reference Signals
Synchronization Preambles
Mapping of symbols to physical resources
SC-FDMA

Figure 1: 3GPP Sidelink L1 Processing Chain

The main highlights, with emphasis on differentiation aspects from regular uplink/downlink design, are reported in the following points[4]:

- *Timing synchronization and system information acquisition* is facilitated by a broadcast transport channel, SL-BCH, and its physical counterpart, PSBCH. These channels are similar to the BCH/PBCH broadcast channel used in LTE DL for cell and system acquisition support. They are used for broadcasting a set of pre-ambles and basic system information within a certain region. A set of primary and secondary preambles, PSSS and SSSS, are used for synchronization purposes, similar to legacy PSS and SSS sequences. An SL Master Information Block, MIB-SL, similar to the legacy LTE MIB carries the sidelink system information. Regarding the V2X specifications introduced in Rel.14, slight modifications are defined for SL-BCH, SSSS and the MIB-SL transmission period.
- *Sidelink Discovery* is facilitated through a transport channel, SL-DCH and its physical counterpart, PSDCH. SL-DCH follows the Downlink Shared Channel structure. Higher-layer specifications are actually absent in the discovery mode, since the announcement messages sent by the D2D UEs are PHY Transport Blocks formed with zero MAC overhead. Filling the TB payload is left open and depends on the ProSe application. For the discovery mode no modifications are defined for V2X.
- *Sidelink Communication* is facilitated using a transport channel, SL-SCH, and its physical counterpart, PSSCH. These resemble the DL-SCH and PDSCH channels used for legacy DL data transmission. Data arrives from MAC, which in turn has arrived from higher layers. Minor modifications are defined for V2X.
- In order for the receiving D2D UE to successfully decode the physical communication channels, information regarding the specific resources assigned for transmission and the transmission configuration is needed. These are carried in a newly introduced *sidelink control channel*, called SCI, which resembles the downlink DCI concept. The SCI is carried

[4] Specifications for V2V will be finalized in June 2017. Here we present information based on non-frozen versions.

in the PSCCH channel, which is similar to the legacy cellular PDCCH/PUCCH channels. For D2D, SCI Format 0 is introduced, whereas for V2V a new format, SCI Format 1, is defined (from Rel.14 and on).

- *Physical channels estimation* is enabled by the introduction of SL demodulation reference signals (SL-DMRS) which are highly similar to UL reference sequences. SL-DMRSs are multiplexed with the payload of the PSBCH, PSDCH, PSCCH, and PSSCH. For D2D, two DMRS symbols per subframe are used for PSBCH, PSDCH, PSCCH, PSSCH. For V2X, three DMRS symbols are used for PSBCH and four symbols for PSCCH and PSSCH.
- *Single-layer transmission* (no transmit diversity/MIMO) is only allowed, as per the latest Release. For Release 15 and on, at least *transmit diversity* will be considered.
- All physical channels are modulated following UL SC-FDMA before transmission. The only difference compared to UL is that the last symbol is zeroed, although taken into account during L1 processing.
- Sidelink communication is *half-duplex* and *no feedback channels* (e.g. for reporting back channel state information) are defined.

1) Timing & System Information Acquisition

Timing & system information acquisition is achieved using a special subframe type called broadcast/synchronization (or simply reference) subframe. Reference subframes are transmitted periodically, i.e. every 40 msec for D2D and 160 msec for V2V, or at-once, in UL subframes configured by the coordinating LTE eNB[5]. Transmissions occur at a fixed subframe offset with respect to the LTE cell subframe timing. The offset is determined through the *syncOffsetIndicator* L3 parameter, which is part of the *SL-SyncConfig* Information Element (IE) transmitted within the SIB18/19. In the standard, reference subframes are triggered by the discovery and communication modes, hence there is no “standalone” broadcast/synchronization procedure. With respect to frequency-domain allocation, reference subframes are always loaded at the central 72 subcarriers of the LTE time-frequency grid, irrespective of the sidelink bandwidth mode. The reference subframe carries two kinds of “information”, synchronization pre-ambles and system configuration information.

Synchronization preambles: These are constructed based on the physical layer “identity” of the reference UE, N_{ID}^{SL}. This, similarly to the Cellular LTE PCI, determines the synchronization pre-amble sequences, PSSS and SSSS, and their mapping to the time/frequency resources of the subframe [10] (§ 9.7). Through these preambles any proximal D2D UE may acquire time synchronization with the reference D2D UE and obtain its physical identity. Note that a D2D reference UE is informed about the sidelink ID through common L3 signaling sent by the legacy network, and in particular by reading out the *slssid* field which is part of the *SL-SyncConfig* IE. The latter IE is transmitted in SIB18 (SIB19) if the reference subframe transmission is triggered by the communication (discovery) mode.

System information: This is a 40-bit sequence, called MIB-SL, which is passed through transport and physical processing blocks and mapped to the reference subframe frequency/time resources following [5] §5.4.1 and [4] §9.6. The specific message structure is shown in Table 1:

Table 1: Sidelink Master Information Block (MIB-SL) Structure

sl-Bandwidth (3 bits)	tdd-ConfigSL (3 bits)	inCoverage (1 bit)	reserved (19 bits)	directFrameNumber (10 bits)	directSubframeNumber (4 bits)

where[6]:

- the *sl-Bandwidth* field provides the bandwidth mode, set equal to the UL bandwidth mode (1.4, 3, 5, 10, 15, 20 MHz). N_{RB}^{SL} expresses accordingly the number of sidelink RBs (6, 15, 25, 50, 75, 100). Note that a UE is informed about the UL bandwidth through an RRC DL broadcast signaling structure sent by the legacy network, called SIB2, and in particular through the *ul-Bandwidth* field.
- the *directFrameNumber* (DFN) and *directSubframeNumber* fields support timing reference in the frame and subframe time-scales respectively. These fields are used if subframe/SFN information from the LTE cell is not available.
- the *inCoverage* field informs the receiving D2D UEs about the coverage status of the reference D2D UE.

Regarding the base-band time-domain signal generation the cyclic prefix should be also known for applying SC-FDMA. This is also communicated to the D2D reference UE through SIB18/19, using the *syncCP-Len* field. This is set as normal or extended, whereas for V2V only normal cyclic prefix is allowed. To aid PSBCH decoding, a sequence of demodulation reference signals (2 for D2D, 3 for V2V) is generated and mapped across the whole bandwidth of specific SC-FDMA symbols of the reference subframe following [6] §9.8.

2) Sidelink Discovery

This mode is a “broadcast” like procedure for sending short/light piece of information. Specifically, it enables a D2D UE:

[5] In case that the underlying D2D UEs are outside the coverage of an LTE eNB, a pre-defined configuration (stored for example in the SIM card) may be used.

[6] *tdd-ConfigSL* is irrelevant for FDD

- to announce its presence to potentially interested proximal UEs through sending a message containing its application identity or other useful information fields (e.g. GPS coordinates, time, etc.);
- to monitor the presence of other proximal UEs by detecting and decoding the corresponding discovery messages, and under conditions, also respond to them using similar discovery messages.

From a signaling point of view, direct discovery is controlled by the LTE network and in particular by a set of ProSe functionalities running in the LTE Core Network [11]. These functions provide the interface of the D2D radio with the relevant D2D applications. In this analysis we do not consider any core-level association procedure (authentication, discovery request and associated responses), but assume that D2D UEs have been granted access to perform radio-level discovery announcement and monitoring. Thus, the focus is on the PC5 discovery procedures. The specification of the discovery mode transmission/reception involves two key functionalities, subframe generation/recovery & resources allocation, both detailed below.

Sidelink discovery subframe generation/recovery:

This includes the transmitter-specific steps, i.e. the preparation of the discovery message transport block, the application of L1 transport and physical channel encoding/processing, the calculation and attachment of DMRSs and the triggering of reference subframe(s), as well as the "reverse" receiver-specific steps. The discovery subframe carries a PC5 discovery message. Following [11] § 11.2.5 there are 10 types of PC5 discovery messages, depending on the scope of the procedure. For Open ProSe discovery, a 232-bit message structured as in Table 2 ([11], Table 11.2.5.1.1), is defined:

Table 2: Structure of Open ProSe Discovery Message (Discovery PHY Transport Block)

Message Type (4 bits)	ProSe Application Code = PLMN ID (MCC/MNC) + ProSe App ID (184 bits)	Message Integrity Check (32 bits)	UTC-based Counter LSB (8 bits)

where:

- the message type field indicates the discovery type (e.g. open, restricted, etc.) and the content type (e.g. announce, query, etc.) [11] § 12.2.2.10
- the ProSe application code field [11] § 12.2.2.6 includes the legacy network PLMN identifier (MCC and MNC) as well as an identifier called ProSe Application ID [11]§ 24.2. The latter is defined during the D2D authentication, is assigned by the ProSe core functionalities, and is associated with the D2D application triggering D2D discovery.
- the message integrity check (MIC) is used to validate the ProSe application code field setting [11] § 12.2.2.11
- the UTC-based Counter LSB indicates the timing of the discovery transmission opportunity [11] § 12.2.2.22

MAC leaves the discovery message intact (see [15] § 6.1.4) as 3GPP specifies transparent MAC implementation. MAC forwards this 232-bit sequence (discovery transport block) directly to the PHY for transport processing, following [13] §5.4.4 and physical channel processing, following [10] §9.5. To assist PSDCH decoding at the receiving side, a sequence of demodulation reference signals is generated and mapped to the discovery D2D subframe following [14]§9.8. Finally, the time-domain waveform is created using SC-FDMA modulation. The cyclic prefix is configured using L3 signaling, and in particular through the *cp-Len* field (set as normal or extended), which is a member of the *SL-DiscConfig IE*. Observe that the cyclic prefix of the discovery channel could be set independently from that of the broadcast (and as it will be seen later that of the communication channel as well).

We conclude by mentioning that when the discovery mode triggers the reference subframe transmission, then the latter will be transmitted in the subframe(s) indicated by the *syncOffsetIndicator* L3 parameter as follows: (i) if *syncOffsetIndicator* indicator coincides with the first LTE subframe assigned for sidelink discovery transmissions, then the specific subframe will be used; (ii) otherwise the closest (in-time) subframe indicated by *syncOffsetIndicator,* which precedes the first LTE subframe assigned for sidelink discovery transmissions, will be selected.

Sidelink discovery mode resources allocation:

Recall that D2D reuses the legacy UL radio resources, and in this respect if we let D2D UEs to blindly select arbitrary LTE subframes and PRBs then it is highly possible that uncoordinated interference may corrupt both UL and SL reception. Coordinated allocation of sidelink resources is realized in two levels: i) in the inter sidelink-uplink level, which is responsible for the determination of sidelink resource pools to avoid conflict with resources used for regular uplink transmissions; ii) in the intra-sidelink level, responsible for the determination of UE-specific resources, based on the configured sidelink resource pool(s). This is used for minimizing/avoiding inter D2D interference. For receiving D2D UEs, multiple resources may be monitored in order to "listen" for discovery announcements coming from multiple ProSe UEs.

The configuration of the subframe/PRB pools is defined for a certain cell period. The starting subframe and the duration of the period are defined with respect to cell timing through the following L3 parameters [14] § 14.3.3:

- *discPeriod*, which is allowed to be configured as 32, 64, 128, 256, 512, and 1024 radio frames. The *discPeriod* field is included in the SL-DiscResourcePool IE, member of the SL-DiscConfig IE, where the latter is transmitted within the SIB19 or UE-specific RRCConnectionReconfiguration messages.
- *offsetIndicator*, which is an integer value ranging from 0 to 10239 and indicating the offset of the subframe pool with respect to SFN #0.

Time and Frequency Resource Pools Configuration ([14] § 14.3.3)

Sidelink transmissions only occur on certain subframes, forming **a *discovery subframe pool*, which may carry both sidelink and uplink control signaling and data**, whereas the rest subframes carry only uplink signaling and data. The pool is formed based on the following L3 parameters (members of SL-DiscResourcePool IEs):

- *subframeBitmap*, a 40-bit (for FDD) binary vector, indicating the sidelink/uplink ('1') and the regular uplink ('0') subframe indices, within a 40-subframes length period;
- *numRepetition*, an integer value ranging from 1 to 5, indicating the number of times the *subframeBitmap* is repeated within a discovery period.

A sidelink discovery PRB pool contains two equal length subsets of contiguous PRBs. PRB indexing is considered with respect to PRB #0 of the legacy LTE cell. The subsets are defined based on three numerical L3 parameters (*prb-Start* , *prb-End* , *prb-Num*) as follows:

- Subset #1 includes PRBs with indexes greater than or equal to *prb-Start* and less than *prb-Start* + *prb-Num*.
- Subset #2 includes PRBs with indexes greater than *prb-End* - *prb-Num* and less than or equal to *prb-End*.

UE-specific discovery resources allocation ([14] § 14.3.1)

UEs are assigned time and frequency resources for sending/monitoring discovery messages to neighbor UEs from the respective resources pool. **Each unique discovery message is loaded into 2 consecutive PRBs of a single subframe**. Retransmissions of the same PSDCH in different PRB-pair/subframe combinations are also allowed. In particular, by setting the L3 parameter *numRetx* appropriately (*numRetx* range is 0 – 3), up to 4 transmissions per discovery TB, thus in total 5 transmissions, are supported.

There are two types of UE-specific resource allocation:

- Type 2B or "Scheduled", which is centrally configured for all D2D UEs at the LTE eNB.
- Type 1 or "UE-selected", which allows UEs to autonomously select the resources;

For both resource allocation types, the objective is to avoid, as much as possible, the assignment of common time/frequency resources to different discovery TBs.

For the "Scheduled" type, conflicts may be fully prevented, as the eNB is fully responsible for the allocation decisions. Given a discovery subframe/PRB pool, the starting indices of the time and frequency resources within the pool are determined through L3 parameters *discSF-Index an discPRB-Index* respectively. In case hopping resource allocation is applied, three additional L3 parameters must be configured, a, b, and c, in order to determine the exact hopping pattern. All these parameters are part of the *SL-DiscConfig* IE.

For the "UE-selected" resource allocation type, UEs select autonomously the exact time and frequency resources from the pool. This is done using a randomization pattern based on a MAC configuration parameter called "resource index" (n_{PSDCH}), from which the actual subframe and PRB indexes (within the pool) for carrying the discovery message are extracted. Different UEs should select different n_{PSDCH} to avoid interference. Although there is no guarantee that resource allocation conflicts are avoided, the random nature of the resource index provides an acceptable level of protection. At the receiver side, a UE may "blindly" monitor for multiple discovery messages by investigating resources corresponding to different n_{PSDCH} settings.

We provide a simple numerical example for clarifying the interpretation of the resource index configuration: Assume a subframe pool containing 40 subframes and 16 PRBs, and that *numRetx* = 1, i.e. 2 transmissions per discovery TB are configured. Given that each unique TB transmission occupies a single subframe and two PRBs, then there are $N_t = 40/2 = 20$ unique time resources and $N_f = 16/2 = 8$ unique frequency resources. Thus, in total there are $N_t \cdot N_t = 160$ non-conflicting UE-specific resources. Any resource index value n_{PSDCH} selected from 0 to 159 will lead to disjoint resource assignments. The mapping of each resource index to the specific subframe/PRB-pair combination is done based on the formulas given in [14] § 14.3.1

3) Sidelink Communication

The sidelink communication mode was initially introduced in 3GPP D2D within the context of public safety applications (e.g. VoIP, MCPTT). Commercial D2D communication services are also envisaged for LTE Evolution and 5G. Special attention has been paid to V2X applications, starting from Release 14. Similar to the discovery mode, resource allocation configuration is based on the concept of sidelink pools for mode-specific allocation and centralized/autonomous decisions for UE-specific allocation. L1 processing and subframe generation on the other hand resembles the approach followed for the LTE downlink. In particular, each transmission involves: i) a "control" part, which is used for informing the receiving UEs about the exact resources used for carrying the data as well as the transmission configuration (PRB allocation, modulation and coding scheme, re-transmission opportunity) which should be known at the receiver for data recovery; ii) a "data" part which is used to carry the payload bits. In the standard, D2D communication is distinguished from V2X using a parameter called sidelink transmission mode. Modes 1 and 2 refer to D2D while modes 3 and 4 to V2X.

Sidelink Communication control & data subframe generation/recovery

Control Channel for D2D Communication

Control information is expressed in the form of SCI messages, which are similar to DL control messages, known as DCI. SCI Format 0 was introduced in Rel.12 ([15] §5.4.3.1.1) for D2D communications. SCI-0 contains 37 – 45 bits, depending on the SL bandwidth mode. The SCI fields are populated as follows:

- for sidelink transmission mode 1 ("fully controlled" mode) using higher layer information carried by L3 control signalling, i.e. RRC, and L1 control signalling configured at the eNB as DCI messages, in particular DCI Format 5 ([15] §5.3.3.1.9),
- for sidelink transmission mode 2 ("autonomous" mode) based on autonomous decisions taken by each transmitting UE.

The structure of SCI Format 0 and DCI Format 5 messages are given in Table 3 and Table 4 respectively.

Table 3: Sidelink Control Information Format 0 (Standard D2D) Message Structure

Frequency Hopping flag (1 bit)	Resource block assignment and hopping resource allocation (5,7,9,11,12,13 bits)	Time Resource Pattern (7 bits)	Modulation and coding scheme (5 bits)	Timing advance indication (11 bits)	Group Destination ID (8 bits)

Table 4: Downlink Control Information Format 5 (DCI Format 5) Message Structure

Resource for PSSCH (6 bits)	TPC for PSCCH, PSSCH (1 bits)	Frequency Hopping flag (1 bit)	Resource block assignment and hopping resource allocation (5 – 13 bits)	Time Resource Pattern (7 bits)

SCI Format 0 message is constructed as follows:

- *Frequency hopping flag* and *Resource block assignment and hopping resource allocation* fields provide the necessary information for the receiving UEs to identify the PRBs where the data channel (PSSCH) resides. Sidelink PRB resource allocation is based on the principles of uplink resource allocation. For the non-hopping case only type-0 is supported, and for hopping both type 1 and type 2 (implicit and explicit hopping pattern definition) [14] § 8.1, § 8.4. For sidelink transmission mode 1 both fields are defined in the DCI 5 message and are just copied to SCI. For mode 2 the fields are configured autonomously by the transmitting D2D UE. For both modes, the selected PRBs should belong to the sidelink communication PRB pool.
- *Time resource pattern (TRP)* provides the time-domain resource allocation for the data channel (PSSCH), and in particular the potential subframes used for PSSCH transmission. In particular, a TRP indicator ranging from 0 to 127, determines a repeated PSSCH subframe pattern, where the subframe mapping process is defined as follows: i) for mode 1 through [14] Tables 14.1.1.1.1-1, where the indicator is copied from the corresponding DCI field ; ii) for mode 2 through [14] § 14.1.1.3 with the help of the L3 parameter *trpt-Subset*, which is transmitted as part of the *CommResourcePool* IE.
- *Modulation and coding scheme*, provides the MCS used for PSSCH. Regarding modulation, QPSK and 16-QAM are supported. For mode 1 the MCS is configured by L3 through the *mcs* field, part of the *SL-CommConfig IE*. For mode 2 it is selected autonomously by the UE.
- *Timing advance indication*, provides a recent uplink-downlink time adjustment value as per [14] § 14.2.1 for mode 1 or set to 0 for mode 2.
- *Group ID* (n_{ID}^{SA}) is an 8-bit code provided by higher layers, indicating the group of UEs which are potentially interested for the transmitted message. This is used at the receiving side for ignoring messages destined to other groups.

The generated SCI message undergoes transport channel encoding following [13] § 5.4.3, and then the generated block is split into two parts, each of which undergoes physical channel encoding following [14] § 9.4. **The generated PSCCHs are loaded into a single PRB of two distinct subframes**, together with DMRS sequences as of [14] § 9.8. Hence, **each control message spans two PRB/subframe resource units.** Notice that this is different from downlink control signaling, where a PDCCH is loaded into a single subframe. At the receiver side the UE should combine both resource units (subframe/PRB pairs) to recover the control signaling information, and thus extract the data channel allocation and transmission configuration.

Control Channel for V2X Communication

A new SCI type, SCI Format 1, has been introduced in Rel.14 to cater for V2X control channel configuration transmission [15] §5.4.3.1.2. As in D2D, a new DCI message type, DCI Format 5A, has been also introduced for carrying eNB control information [15] §5.3.3.1.9A. The structure of SCI Format 1 and DCI Format 5A messages are given in Table 5 and Table 6 respectively.

Table 5: Sidelink Control Information Format 1 (V2V) Message Structure

Priority (3 bits)	Resource Reservation (4 bits)	Frequency Resource Location (0,3,6,7,8 bits)	Time Gap (4 bits)	Modulation and Coding Scheme (5 bits)	Retransmission Index (1 bits)	Reserved Bits (15,12,9,8,7 bits)

Table 6: Downlink Control Information Format 5A (DCI Format 5A – V2V) Message Structure

Carrier Indicator (3 bits)	Lowest Index of the Subchannel allocation (0,2,3,4,5 bits)	Frequency Resource Location (0,3,6,7,8 bits)	Time Gap (4 bits)

SCI Format 1 message is constructed as follows:

- *Priority*, includes one of 8 possible values, corresponding to the *ProSe Per-Packet Priority* (PPPP) configured by the ProSe application-layer and used for QoS purposes ([16], § 5.4.6);
- *Resource Reservation*, is a field used only in the autonomous V2X mode (mode-4), for announcing resources to be used based on sensing decisions ([14] § 14.2.1).
- *Frequency Resource Location*, is a bit pattern used to define PSSCH PRB resources ([14] § 14.1.1.4C);
- *Time Gap*, is the subframe gap between the first and (optional) second PSSCH transmission (re-transmission) ([14] § 14.1.1.4C).
- *Modulation and Coding Scheme*, determines the MCS used for PSSCH;
- *Retransmission Index*, is a boolean field, denoting if the corresponding PSSCH refers to the first transmission or the (optional) second transmission, i.e. the re-transmission ([14] § 14.1.1.4C).
- *Reserved bits*, are dummy zero bits, added for assuring that the SCI message size is 32 bits.

The generated SCI message undergoes transport channel encoding following [13] § 5.4.3, as in the D2D control channel case, with a slight difference in the sequence used for PUSCH interleaving. An identifier notated as n_{ID}^{X} which corresponds to the SCI CRC checksum is also generated; this is going to be used in data encoding (PSSCH). The encoded block then undergoes physical channel encoding following [14] § 9.4, as in the D2D control channel case. **The PSCCH is loaded (together with DMRSs) in two consecutive PRBs of a single subframe, differently from the D2D control channel case**, where a single PRB and two subframes are used. In case a retransmission is configured, the same PSCCH content is loaded into another subframe/PRB-pair from the pool.

Data Channel for D2D Communication

Differently from broadcast, discovery and communication control channel, where the transport block size is fixed, in the sidelink communication data channel, the size is not pre-determined. Instead it is dynamically formed, based on the number of assigned PRBs and the configured MCS. Following the mapping methodology in [14], and specifically Table 8.6.1-1, the SL-SCH transport block size is first calculated and then filled with bits from higher layers. The transport block then undergoes transport channel encoding following [13] § 5.4.2. **The generated block is then split into 4 parts, and each part undergoes physical channel processing** according to [10] § 9.3. **Each PSSCH**, along with relevant DMRSs ([10] § 9.8) **is loaded into distinct subframes/PRB subsets**, belonging to the sidelink communication pool. At the receiver side all 4 PSSCHs should be combined to recover the data transport block.

Data Channel for V2X Communication

As in the D2D communication, the transport block size is initially determined based on the assigned PRB bandwidth and the MCS configuration. The transport block undergoes transport channel encoding following [13] § 5.4.2. The only difference compared to D2D communication is the scrambling sequence definition; in D2D it is based on the group id (n_{ID}^{SA}) , whereas for V2X on the parameter n_{ID}^{X} , calculated during the SCI transport channel encoding. Then, the **encoded block undergoes physical channel processing** as in [10] § 9.3 **and loaded into a single subframe and a subset of sidelink pool PRBs**. If re-transmission is configured in PSCCH, another subframe transmits an identical PSSCH.

Sidelink Communication Mode Resource Allocation

Sidelink communication mode resources allocation follows the same principles applied for the discovery mode. First, mode-specific time and frequency resource pools are formed, and at a second stage D2D UEs are allocated specific resources for control and data from these pools.

Time and frequency resource pools configuration for D2D communication [14] § 14.2.3, 14.1.3, 14.1.4

The configuration of the subframe/PRB pools is defined for a certain cell period. The starting subframe and the duration of the period are defined with respect to cell timing through the following L3 parameters:

- *offsetIndicator*, which is an integer value with range 0 to 319, indicating the offset of the subframe pool with respect to SFN #0.
- *sc-Period*, which is allowed to be configured as 40, 80, 160, 320 subframes. The *sc-Period* field is included in the SL-CommResourcePool IE, transmitted within the SIB18 or the RRCConnectionReconfiguration messages.

PSCCH and PSSCH subframe pools are constructed based on L3 parameter, *subframeBitmap*, which defines a 40-bit length pattern for uplink/sidelink resources distribution. PSCCH and PSSCH pools are separated. Therefore, a UEs' control and data may not be multiplexed in the same subframe. The PSCCH pool precedes. For transmission mode 1 the PSSCH pool starts exactly after the end of the PSCCH pool, whereas for mode 2 it starts at a fixed time-offset with respect to the PSCCH pool.

The PRB pool is defined exactly as in the discovery mode case, using three L3 parameters, *prb-Start*, *prb-End*, *prb-Num*, which are part of an IE dedicated to the communication mode configuration (SL-CommConfig IE). A common PRB pool for PSCCH and PSSCH is defined (since control and data are loaded in different subframes anyway).

UE-specific resources allocation for D2D communication [14] § 14.1.1.1, 14.1.1.2, 14.2.1.1, 14.2.1.2:

For the control channel (PSCCH) a set of two subframe/PRB pairs belonging to the PSCCH resource pool is determined based on a randomization resource index parameter, n_{PSCCH}. This allows to define non-conflicting resource assignments using a single scalar. For mode 1, the transmitting UE is informed about this value from the eNB, in particular through the DCI Format 5 message. For mode 2 the resource index is configured autonomously by the transmitting UE. The receiving UE could look at resources corresponding to specific n_{PSCCH}s or blindly search in a n_{PSCCH}-specific range.

For the data channel (PSSCH), a set of four subframes from the PSSCH pool is picked, based on a UE-specific subframe bitmap, I_{TRP}, which indicates the subframe indices within the pool. The standard defines 108 available patterns ([14] Table 14.1.1.1.1-1). A scalar parameter, *TRP*, is used to indicate the UE-specific pattern. *TRP* is carried in the DCI Format 5 message for mode-1 or autonomously selected by the UE in mode-2, using an additional higher layer parameter, *trpt-subset*. For both modes, the *TRP* field is also copied at the SCI Format 0 message, in order for the receiving D2D UE to know where to look for PSSCH data. The PRBs used for the PSSCH are determined based on the *Frequency hopping flag* and *Resource block assignment and hopping resource allocation* fields (carried in DCI-5 and again copied in SCI-0) for mode-1 or determined autonomously for mode-2.

Time and frequency resource pools configuration for V2X communication [14] § 14.2.4, 14.1.5:

Configuration of time and frequency pools is not employed on a period basis as in the sidelink discovery and standard communication modes. Instead it applies over the whole SFN cycle ([14] § 14.1.5), where an L3-defined pattern, *subframeBitmap-r14*[7], together with a set of excluded subframes (used for synchronization, downlink or other special purposes), determine the available subframes. The specific bitmap length is 10-100, and is repeated over the SFN cycle. A subframe offset indicator parameter, *offsetIndicator-r14,* indicates the 1st available V2X subframe. Frequency-domain resources are organized as follows:

- PRBs are grouped in subchannels.
- The subchannel size is determined by the L3 parameter *sizeSubchannel-r14*, part of SL-CommResourcePool IE. For up to 5 MHz bandwidth mode, the following subchannel sizes are allowed: 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20.
- The number of subchannels allocated to a V2X PSCCH/PSSCH resource pool is determined by another L3 parameter, *numSubchannel-r14*, which may be configured as 1, 3, 5, 10, 15, 20.
- The pool contains a contiguous set of PRBs. The lowest PRB index of the subchannel with the lowest index is given by the L3 parameter, *startRB-Subchannel*[8].
- PSCCH and PSSCH PRB pools may be adjacent or not. This is determined by the L3 boolean parameter *adjacencyPSCCH-PSSCH-r14*. For the adjacent mode, PSCCH and PSSCH PRB pools are identical, otherwise the PSCCH pool starting PRB index is given by the L3 parameter *startRB-PSCCH-Pool-r14*.

UE-specific resources allocation for V2X communication [14] § 14.1.1.4A, 14.1.1.4B, 14.1.1.4C, 14.2.1

Each transmission opportunity for the control channel, i.e. the regular transmission and the optional re-transmission, **occupies a single subframe and two PRBs**. The DCI Format 5A field SF_{gap} determines the time-offset (in # subframes) for the two transmission opportunities. Regarding the PRB pair: i) for the adjacent mode, the first two PRBs of the PSSCH/PSCCH UE-specific PRB subsets are used; ii) for the non-adjacent mode, two PRBs from the dedicated PSSCH pool, starting from PRB index given by L3 parameter, *nsubCHstart*, are used.

For the data channel (PSSCH), exactly the same subframes assigned for PSCCH are used. The PRB set assigned to each UE is based on the parameters *nsubCHstart* (lowest index of allocated subchannel) and *LsubCH* (number of allocated subchannels), which are derived from the *Frequency Resource Location* field of the DCI Format 5A message (and copied to SCI Format 1 messages in order for the receiver to recover the correct resources). For the adjacent PSCCH-PSSCH mode, care is taken such that the first two PRBs of the lowest index subchannel are not allocated for PSSCH, since these are used for PSCCH. Consequently, control and data for a specific V2X communication transmission are located in the same time resource and potentially in adjacent frequency resources, which is not the case for standard D2D.

We conclude the current Section with Table 7, which summarizes the physical signals, physical channels and transport channels relevant to sidelink for both D2D and V2X modes.

[7] r14 is added to distinguish it from the corresponding bitmap parameter defined for standard D2D communication.

[8] An example configuration is as follows: Assume *sizeSubchannel-r14 = 4, numSubchannel-r14 = 3*, and *startRB-Subchannel = 2,* then the PRB pool contains i) Subchannel #0, including PRBs #2,3,4,5; ii) Subchannel #1, including PRBs #6,7,8,9 and iii) Subchannel #3, including PRBs #10,11,12,13.

Table 7: Transport, Signalling, Physical Channel, and Physical Signals for Sidelink D2D/V2X

OPERATION MODE	**Discovery**	**Communication**		**Timing & System Information Acquisition**
PHY Transport Block				
Size/Message	232-bits (arriving from application)	L1 signalling (37-45 bits, D2D; 32 bits, V2X)	Variable size (arriving from L2)	40-bits MIB-SL
Transport/Signalling Channel Processing				
Transport/Signalling Channel	**SL-DCH**	**SCI Format 0 (D2D) Format 1 (V2V)**	**SL-SCH**	**SL-BCH**
Segmentation & CRC attachment	*DL-SCH*	-	*DL-SCH*	-
TB CRC attachment	g_{CRC24A}	g_{CRC16} (no scrambling)	g_{CRC24A}	g_{CRC16}
Channel Coding	TC	CC	TC	CC
Rate Matching	TC	CC	TC	CC
Interleaving (c_{mux})	$2 \cdot (N_{SL}^{symb} - 1)$	$2 \cdot (N_{SL}^{symb} - 1)$, D2D $2 \cdot (N_{SL}^{symb} - 2)$,V2X	$2 \cdot (N_{SL}^{symb} - 3)$, D2D $2 \cdot (N_{SL}^{symb} - 2) - 3$, V2X	$2 \cdot (N_{SL}^{symb} - 3)$, D2D $2 \cdot (N_{SL}^{symb} - 2) - 3$, V2X
Physical Channel & Signals Processing				
Physical Channel	**PSDCH**	**PSCCH**	**PSSCH**	**PSBCH**
Scrambling (c_{init})	510	510	$n_{ID}^{SA}(n_{ID}^{X}) \cdot 2^{14} + n_{ssf}^{PSSCH} \cdot 2^{9} + 510$, D2D(V2V)	N_{ID}^{SL}
Modulation	QPSK	QPSK	QPSK, 16QAM	QPSK
Layer Mapping/ Precoding	Single-Layer, Single Antenna (no MIMO or Tx Diversity)			
Transform Precoding	*As in uplink*			
Demodulation Reference Signals				
Group hopping	disabled	disabled		disabled
Reference Signal Identifier (n_{ID}^{RS})	-	n_{ID}^{X}	n_{ID}^{SA}	-
Slot Index (n_s)	-	-	n_{SS}^{PSSCH}, D2D $2 \cdot n_{SS}^{PSSCH}$ (slot 0), $2 \cdot n_{SS}^{PSSCH} + 1$ (slot 1), V2V	-
Sequence-shift pattern (f_{ss})	0	0 for D2D, 8 for V2V	$n_{ID}^{SA} mod 30$, D2D $\lfloor n_{ID}^{X}/16 \rfloor mod 30$, V2V	$\lfloor N_{ID}^{SL}/16 \rfloor mod 30$
Sequence hopping	disabled	disabled	disabled	disabled
Cyclic shift ($n_{CS,\lambda}$)	0	0 for D2D, {0,3,6,9} for V2V	$\lfloor n_{ID}^{SA}/2 \rfloor mod 8$, D2D $\lfloor n_{ID}^{X}/2 \rfloor mod 8$, V2V	$\lfloor N_{ID}^{SL}/2 \rfloor mod 8$
Orthogonal sequence $[w^{\lambda}(0)\, w^{\lambda}(1)]$	$[+1 + 1]$	$[+1 + 1]$, D2D $[+1 + 1 + 1 + 1]$, V2V	D2D: $[+1 + 1]\ if n_{ID}^{SA}\ mod 2 = 0$ $[+1 - 1]\ if n_{ID}^{SA}\ mod 2 = 1$ V2V: $[+1 + 1 + 1 + 1]\ if n_{ID}^{X}\ mod 2 = 0$ $[+1 - 1 + 1 - 1]\ if n_{ID}^{X}\ mod 2 = 1$	$[+1 + 1]\ if N_{ID}^{SL}\ mod 2 = 0$ $[+1 - 1]\ if N_{ID}^{SL}\ mod 2 = 1$
Synchronization Signals				
PSSS	$u = 26, if\ N_{ID}^{SL} \leq 167, else\ u = 37$ @ slot 0 symbols #1,#2 (normal cp), #0,#1 (ext. cp)			
SSSS	$N_{ID}^{(1)} = N_{ID}^{SL} mod 168, N_{ID}^{(2)} = \lfloor N_{ID}^{SL}/168 \rfloor$ @ slot 1 symbols #4,#5 (normal cp), #3,#4 (ext. cp)			

III. An Open Software Library for implementing The 3GPP D2D Radio Protocol

A. Scope – Features

lte-sidelink is a software library developed in MATLAB, that implements the most important functionalities of the 3GPP LTE sidelink interface[9]. It is freely and openly available through the following public online repository: https://github.com/feron-tech/lte-sidelink. The project is licensed under the GNU Affero General Public License v3.0. The library provides an (almost) complete implementation of the sidelink physical signals, physical channels and transport layer functionalities described in the 3GPP standard. In addition, it provides the necessary transceiver processing functionalities for generating and/or recovering a real sidelink signal for simulation/emulation purposes and/or over-the-air experiments using SDR boards. The code is highly-modular and documented in order to be easily understood and further extended. The library has many **usages**. For example, it may be used as:

- An LTE sidelink waveform generator.
- An end-to-end sidelink link-level simulator.
- A core component of a sidelink system-level simulator.
- A platform for testing new resource allocation/scheduling algorithms for D2D/V2V communications.
- A tool to experiment with live standard-compliant sidelink signals with the help of SDR boards.

The following **features** are currently (as of v1.2.0) supported:

- Sidelink air-interface compliant with:
    - "Standard" D2D based on Rel.12 and Rel.13
    - D2D tweaks for V2V communications based on Rel.14
- Broadcast transport & physical channel processing functionalities
    - Generation and recovery of MIB-SL messages
    - Encoding and recovery of the SL-BCH transport channel
    - Encoding and recovery of the PSBCH physical channel
    - Demodulation Reference Signals (DMRS) construction and subframe loading
- Sidelink discovery mode
    - Physical Signals and Channels: SL-DCH, PSDCH, PSDCH DMRS
    - Subframe/PRB discovery pool formation & UE-specific resource allocation
- Sidelink communication mode
    - Physical Signals and Channels for Control Signaling: SCI Format 0, PSCCH, PSCCH DMRS
    - Physical Signals and Channels for Payload: SL-SCH, PSSCH, PSSCH DMRS
    - Subframe/PRB communication pool formation & UE-specific resource allocation for control/data
- V2X sidelink communication mode
    - Physical Signals and Channels for L1 signaling: SCI Format 1 (V2V), PSCCH, PSCCH DMRS
    - Physical Signals and Channels for Payload: V2X PSSCH, PSSCH DMRS
    - Subframe/PRB pool formation & UE-specific resource allocation for V2X communication for control/data
- Synchronization preambles (PSSS, SSSS) construction & recovery
- Subframe creation, loading and time-domain signal transformation
- Complete receiver processing functionality for sidelink-compliant waveforms
    - time-synchronization
    - frequency-offset estimation and compensation
    - channel estimation and equalization
    - signal demodulation/decoding
- Example scripts for configuring and running end-to-end broadcast, discovery, and D2D/V2X communication transceiver simulation scenarios.

The code **repository** is structured as follows:

- The *home* directory includes the example scripts for testing sidelink functionality.
- The *core* directory includes the sidelink-specific functionalities implementations, i.e. the physical/transport channels, the DMRSs, synchronization signals, channel estimation procedures, organized in classes.
- The *generic* directory includes generic (non-sidelink specific) tx/rx functionalities implementations, organized in functions, i.e. signal, physical and transport channel blocks, necessary for core classes implementation.

[9] The majority of the content used in the current Subsection is also available in the project online repository, https://github.com/feron-tech/lte-sidelink/blob/master/README.md (in "readme" format) or in https://feron-tech.github.io/lte-sidelink/ (in "web-site-friendly" format). Refer to any of these links for software and documentation updates.

A list of library **dependencies/known issues** is also reported:

- All functionalities are developed in-house except for: i) CRC encoding/detection, ii) Convolutional Encoding/Decoding. For these, the corresponding MATLAB Communication Toolbox System Objects have been used. In-house versions of these two blocks will be also provided soon.
- Convolution channel coding has been applied for sidelink discovery and communication modes transport channel processing, instead of standard-compliant turbo coding.
- The last subframe symbol in SC-FDMA modulation is not zeroed as specified in the standard.
- V2X communications support is added but not fully tested.
- Testing of the code has been done in MATLAB R2016b.

Next, based on the provided library, we develop an end-to-end transceiver simulation example for the sidelink discovery mode, including the complete L1 processing and resources allocation procedures, following the respective specifications.

B. *An Example Usage: End-to-End Sidelink Discovery Mode Simulation*

The sidelink discovery mode is used for sending (in a broadcast way) short messages to neighbor UEs. The sidelink discovery transmission/reception procedure involves two key functionalities:

- Selection of time (subframes) and frequency (PRBs) resources for sending/monitoring discovery messages.
- L1 processing, i.e. signal generation (for tx) and transport block recovery (for rx) operations.

Next, we provide a high-level walkthrough for setting up, configuring, and running a complete transceiver simulation scenario for the sidelink discovery mode (refer to library file sidelink_discovery_tester.m).

1) Configuration

The available parameters may be organized in three subsets:

- **A basic** configuration set, providing the key operational system parametrization. It includes the cyclic prefix length (`cp_Len_r12`), sidelink bandwidth mode (`NSLRB`), sidelink physical layer id (`NSLID`), sidelink transmission mode (`slMode`), synchronization offset with respect to SFN/DFN #0 (`syncOffsetIndicator`), and synchronization period (`syncPeriod`), for both discovery and triggered broadcast/synchronization subframes.
- **Discovery Resources Pool** configuration, providing the sidelink resources allocation parametrization. The available parameters are briefly described in the following bullet points, while the interested reader could refer to the 3GPP standard 36.331, Section 6.3.8, for more details:
    - `discPeriod_r12`: the period (in # frames) for which the resource allocation configuration is valid (available configurations: 32,64,128,256,512,1024 frames)
    - `offsetIndicator_r12`: the subframe offset (with respect to SFN/DFN #0) determining the start of the discovery period.
    - `subframeBitmap_r12`: a length-40 bitmap determining the time (subframe) pattern used for sidelink transmissions (‘1’s determine the subframes available for sidelink transmissions)
    - `numRepetition_r12`: the repeating pattern of the subframe bitmap (available configurations: 1-5)
    - `prb_Start_r12`: the first PRB index of the bottom PRB pool available for discovery transmissions
    - `prb_End_r12`: the last PRB index of the top PRB pool available for discovery transmissions
    - `prb_Num_r12`: the number of PRBs assigned to top/bottom PRB pools for discovery transmissions
    - `numRetx_r12`: the number each discovery message is re-transmitted (available configurations: 0-3)
    - `networkControlledSyncTx`: determines if broadcast/sync subframes should be triggered (1 for triggering, 0 for no-triggering)
    - `syncTxPeriodic`: determines if the triggered broadcast/sync subframes are transmitted at once ("single-shot") or periodically (every 40 subframes)
    - `discType`: determines how sidelink resources are allocated to the different discovery transmissions, i.e. selected by each UE in an autonomous manner (Type-1) or configured by the eNB in a centralized manner (Type-2B).
- **UE-specific resources allocation** configuration, providing the specific subframes and PRBs allocated to each discovery message and the (potential) retransmissions. In particular:
    - For Type-1 resource allocation, a single parameter, `nPSDCH`, determines the exact resources subset. In particular, each `nPSDCH` value corresponds to a distinct combination of a single subframe and a PRB set used for carrying a single discovery message. Using different `nPSDCH` settings for distinct messages announcement allows to avoid intra-sidelink interference.
    - For Type-2B resource allocation, PRB and subframe resources are determined explicitly using a set of two parameters, `discPRB_Index` and `discSF_Index`. Lastly, for hopping-based resource allocation, the applied hopping patterns are determined based on set of three parameters, `a_r12, b_r12,` and `c_r12`.

2) *Running the Example*

An example Type-1 configuration is shown below.

```
cp_Len_r12              = 'Normal';
NSLRB                   = 25;
NSLID                   = 301;
slMode                  = 1;
syncOffsetIndicator     = 0;
syncPeriod              = 40;
discPeriod_r12          = 32;
offsetIndicator_r12     = 0;
subframeBitmap_r12      = repmat([0;1;1;1;0],8,1);
numRepetition_r12       = 5;
prb_Start_r12           = 2;
prb_End_r12             = 22;
prb_Num_r12             = 5;
numRetx_r12             = 2;
networkControlledSyncTx = 1;
syncTxPeriodic          = 1;
discType                = 'Type1';
n_PSDCHs                = [0; 6];
n_PSDCHs_monitored      = n_PSDCHs;
decodingType            = 'Soft';
chanEstMethod           = 'LS';
timeVarFactor           = 0;
```

Notice that in addition to the aforementioned parameters we have included: i) a parameter determining which resources the receiving UE will monitor for identifying potential discovery announcements (`n_PSDCHs_monitored`), ii) a set of three parameters (`decodingType, chanEstMethod, timeVarFactor`), used for tuning channel estimation and channel decoding operations at the receiver side.

Transmission/reception operations are captured in corresponding function blocks, `discovery_tx()` and `discovery_rx()`, respectively. By default, `discovery_tx()` creates a standard-compliant discovery waveform for a period determined by the `discPeriod_r12` parameter. The waveform (stored in the `tx_output` variable) contains not only the discovery signal samples but also the triggered broadcast/synchronization signal samples for the specific period. An example call of the discovery transmitted waveform generation function block is as follows:

```
slBaseConfig = struct('NSLRB',NSLRB,'NSLID',NSLID,'cp_Len_r12',cp_Len_r12, 'slMode',slMode);
slSyncConfig = struct('syncOffsetIndicator', syncOffsetIndicator,'syncPeriod',syncPeriod);
slDiscConfig      =      struct('offsetIndicator_r12',      offsetIndicator_r12,      'discPeriod_r12',discPeriod_r12,
'subframeBitmap_r12', subframeBitmap_r12, 'numRepetition_r12', numRepetition_r12, ...
    'prb_Start_r12',prb_Start_r12,'prb_End_r12', prb_End_r12, 'prb_Num_r12', prb_Num_r12, 'numRetx_r12', numRetx_r12,
...
    'networkControlledSyncTx',networkControlledSyncTx, 'syncTxPeriodic',syncTxPeriodic, 'discType', discType);
slUEconfig = struct('n_PSDCHs',n_PSDCHs);
tx_output = discovery_tx( slBaseConfig, slSyncConfig, slDiscConfig, slUEconfig );
```

The generated time-domain waveform for a period of 50 subframes is illustrated in Figure 2. The subframe locations of the discovery message transmissions (and potentially re-transmissions) depend on the selected `nPSDCH` configuration. In addition, the triggered broadcast/sync subframes repeated every 40 subframes are also shown.

The frequency-domain resource allocation for the discovery message configured with `nPSDCH=0` is also shown in Figure 3. Three transmissions for the particular message have been configured (since `numRetx_r12=2`).

Next, the transmitted waveform passes through a typical channel, and the resulted waveform (stored in the `rx_input` variable) is fed to the discovery monitoring/receiving function block. This is called as follows:

```
discovery_rx(slBaseConfig, slSyncConfig, slDiscConfig,  ...
    struct('n_PSDCHs',n_PSDCHs_monitored), ...
    struct('decodingType',decodingType, 'chanEstMethod',chanEstMethod, 'timeVarFactor',timeVarFactor),...
    rx_input );
```

Notice that in addition to sidelink basic configuration (`slBaseConfig` and `slSyncConfig`) and discovery resources pool configuration (`slDiscConfig`), we provide as input the discovery messages monitoring search space and the channel estimation/decoding parameters. The discovery monitoring function returns the recovered (if any) discovery messages. For the specific configuration example, the following output is printed at the end of the execution (intermediate log/debug messages print-outs are also supported). It is clear that both discovery messages contained in the transmitted waveform have been recovered successfully at the receiver side.

```
Recovered Discovery Messages
          [At Subframe     1: Found nPSDCH =   0]
          [At Subframe     2: Found nPSDCH =   0]
          [At Subframe     3: Found nPSDCH =   0]
          [At Subframe    31: Found nPSDCH =   6]
          [At Subframe    32: Found nPSDCH =   6]
          [At Subframe    33: Found nPSDCH =   6]
```

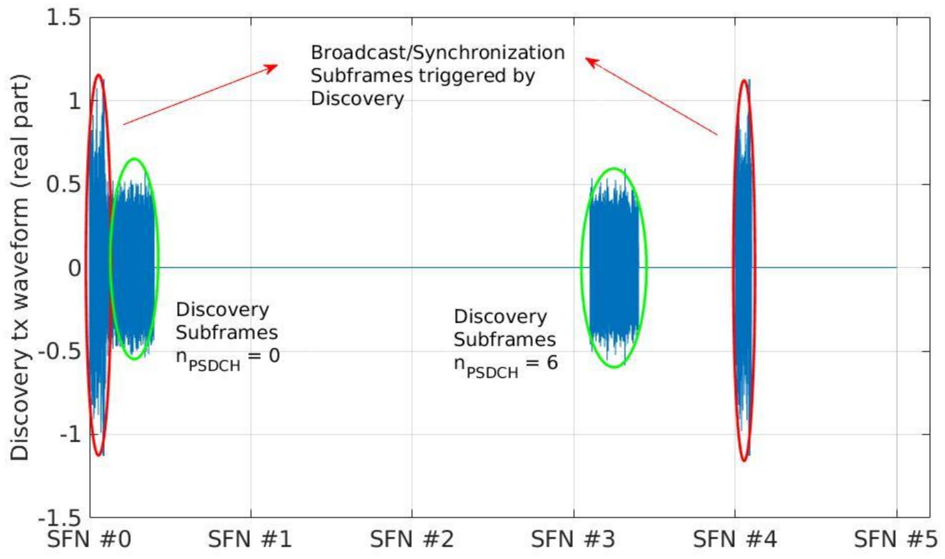


Figure 2: An example discovery mode time-domain waveform

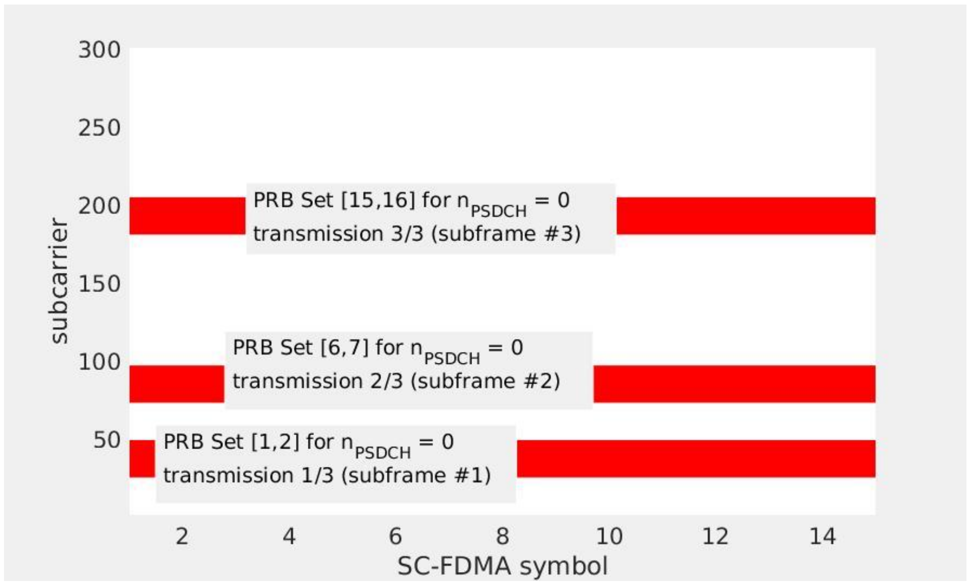


Figure 3: An example discovery mode frequency-domain allocation

C. Simulation-based evaluation results

A simulation-based evaluation study of the D2D radio protocol implementation is necessary for quantitatively characterizing the performance at the link-level and eventually identifying the signal-level range within which the implemented transport and physical channels are recoverable at the receiver side. This study is the starting point for evaluating the software modem in real-world conditions (see Section IV.E).

In this Subsection we consider the performance of the three following sidelink functionalities/modes:

1) Transmission of the **MIB-SL transport block**, which contains the sidelink system information, i.e. the sidelink bandwidth mode, and the frame/subframe timing reference (DFN); this is performed through the sidelink broadcast channels, SL-BCH and PSBCH.
2) Transmission of a **Discovery transport block** (with random payload), using the sidelink discovery mode; this is performed using the SL-DCH and PSDCH channels.
3) Transmission of a **Sidelink Communication Control message** (with payload corresponding to a specific sidelink configuration); this is performed using the SCI Format 0 transport channel and PSCCH.

The transceiver operations, i.e. the generation of the corresponding sidelink waveforms at the transmitter side and the recovery of transmitted information (control or data) at the receiver side, are determined in the *lte-sidelink* library. To evaluate the link-level performance of the developed operations we consider a large number of link realizations affected by a theoretical/statistical channel model. The model includes:

- An AWGN component, modeled by a target SNR level, called "Simulated SNR". Based on a specific SNR level we randomly generate a set of noise samples exhibiting the required noise power.
- A small-scale fading component, generated from a linear time-varying discrete-time channel impulse response for a generic OFDM system. The channel instance is generated based on a power delay profile and a Doppler frequency. In the simulation experiments we consider a relatively slow channel, with Doppler frequency equal to 10 Hz, whereas the number of taps is taken similar to the cyclic prefix length.

In the simulation model we don't consider any time-offset, clock-offset or frequency-offset impairments. Thus, the study assumes perfect synchronization and frequency-offset compensation. All the rest transceiver chain operations, at the signal, time-domain and frequency-domain levels, are taken into account. The non-accounted impairments will be instead considered in the over-the-air evaluation experiments presented at a later stage. A 5 MHz sidelink system has been tested. The target SNR level ranges from 14 dB, where perfect bit recovery behavior for all modes has been observed and goes down to -4 dB, which corresponds to highly challenging signal reception conditions. 2048 transport blocks are simulated per scenario, corresponding to approximately 1-2 million simulated bits

For each sidelink functionality, the following link-level KPIs are extracted:

- Bit Error Rate (BER): this is obtained by comparing the decoded bit payload with the known transmitted bit payload.
- Information Block Detection Ratio: this is obtained by checking the CRC checksum (or equivalently the payload) of the recovered MIB-SL message, discovery transport block, or SCI Format 0 message for each tested sidelink mode. This link-level KPI may be preferred over the raw BER KPI, since it reflects the end-to-end L1 decoding capability of the radio chain.
- Error vector magnitude (EVM): this is a metric for quantifying the link-level performance of the receiver at the modulation-level. It is a measure of "how far" the received I/Q constellation points are from their ideal locations. It is calculated exactly after the transmitted transport block (MIB-SL, discovery message, SCI Format0) has been recovered. The recovered bit sequence is re-encoded and modulated and then compared with the received modulated symbol-sequence given as input to the signal demodulator. The lower the EVM value is the higher is the detection quality. The observed EVM metric could be also easily mapped to an approximate "empirical" SNR value [23].

Results are presented in Figure 4.

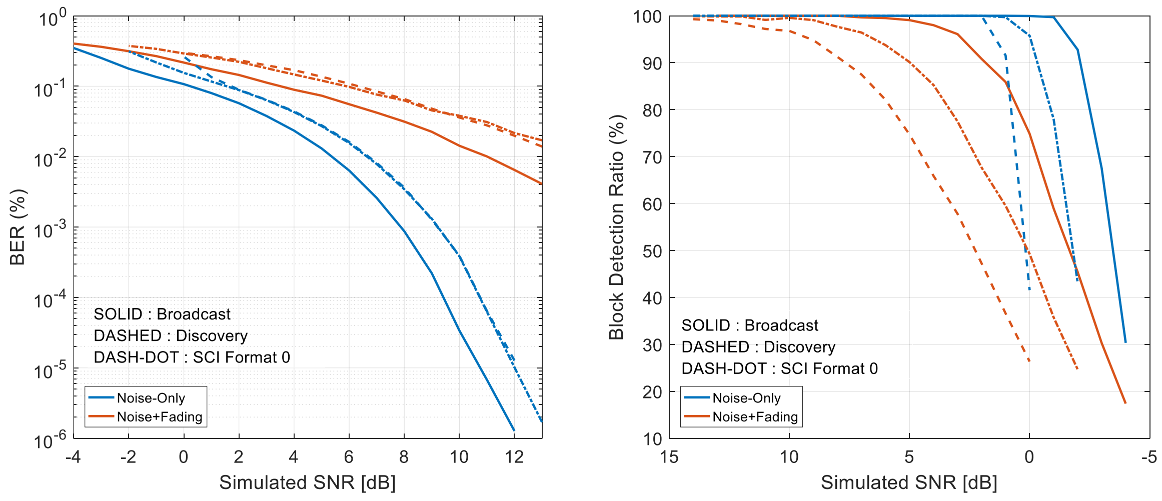


Figure 4: Simulation-based evaluation of D2D radio: Comparative results

IV. An LTE D2D Software Modem Prototype

A. Proposition

In the current Section we will present the ongoing activities regarding the development of a novel D2D software modem, based on COTS components and general purpose programming languages/tools. To the best of our knowledge, this is the first attempt in realizing a real-time standard-compliant 3GPP sidelink software modem. The modem comprises four major components:

- A transceiver implementation including the basic 3GPP D2D radio functionalities running in GPP hosts, such as laptops, desktops, or single-board platforms. The radio functionalities are based on the in-house *lte-sidelink* software library but have been implemented in C/C++ for embedded and real-time operation purposes. Currently, the real-time modem fully supports the sidelink broadcast channel mode, hence, hereafter we implicitly assume during the description only the related functionalities.
- An interface of the transceiver with Ettus USRP SDR boards (currently B210 boards are supported) for over-the-air signal transmission/reception. The interface is based on the Ettus UHD hardware driver[10], and in particular the provided C/C++ API[11]. The interface design is generic enough to support other SDR boards in the future.
- An interface with higher layers for supporting end-user applications. This is currently implemented using a socket application running in user-space.
- A configuration file determining the most crucial operation parameters, such as, the system bandwidth, the sidelink cell-id, the operating carrier frequency, the cyclic prefix mode, the synchronization period, the employed channel equalization method, the operation time, etc.

The specific activity follows the latest trends in softwarizing not only the core network functionalities of the future network but also the radio functionalities, including access nodes and user devices. Through the last few years the evolution of SDR hardware and standard computing equipment capabilities have rendered the (small-scale) software-based experimentation of 4G/4G+ base-station and UE technologies technically viable and economically affordable. Typical examples are OAI (OpenAirInteface)[12], OpenLTE[13], AmariLTE[14], srsUE/srsLTE[15], and LimeNET Network-in-a-Box[16].

B. Requirements, Design Principles, and Architecture

Modems are responsible for carrying out complex operations in very short periods of time. Moreover, they are designed for continuously operating within devices/platforms. Based on these features and limitations, the challenge of building a software modem is to create a fast and robust solution being able to fulfil all the requirements of a common modem, but at the same time provide a scalable implementation which leads to smooth integration. To epitomize, a software modem, as most of software products, should be characterized by code efficiency, scalability, maintainability, portability and robustness [17], but at the same time exploit any available resources and means to provide as fast code as possible.

In principle, the performance of a software modem is bound to the hardware of the hosted machine, and the most suitable programming language to exploit most of the hardware capabilities, is the traditional C in combination with modern C++ features [18]. These two programming languages are able to offer enhanced degrees of freedom to the development process. Furthermore, concerning the software modem portability, it is advisable to load most of the functionalities to the user space to avoid increased dependency on the hardware.

Figure 5 represents the outline of the software modem functionality, including the core transmitter/receiver (transceiver) processing, the interaction with the SDR board (currently USRP), and a “minimal” interface with the application space/layer. As the software modem will not be any more part of a specific hardware, a way of efficient data delivery between the application and the modem is required to be investigated. The most promising candidate seems to be the UDP C Sockets. The traditional Sockets are able to offer an open communication channel [19] and the adoption of UDP packets [20] adds minimum overhead to the transmission. In this manner, the application data may reach the software modem through the application directly. A possible extension would be through a Linux virtual interface [21] forwarding the traffic directly to the modem.

The core functionality of the transmitter takes place inside the UDP Socket Server. The main logic behind the transmitter is to establish a continuous process of forwarding data to the USRP board even if no data are available, as the USRP board is designed to continuously send data and not turning on and off respectively. To accomplish that, two main operations take place in parallel. The first one, shown as “3.1” in the top block of Figure 5, is responsible for retrieving the new arrived data, apply any signal/symbol/bit level processing and then update the data buffer. The second procedure, is responsible for reading data from the buffer/repository and forward them for transmission to the USRP board. If no data are available zeros are sent instead. The receiver is based on a similar logic with that of the transmitter as depicted in the bottom block of Figure 5. Figure 1The process

[10] https://files.ettus.com/manual/
[11] https://kb.ettus.com/Getting_Started_with_UHD_and_C%2B%2B
[12] http://www.openairinterface.org/
[13] http://openlte.sourceforge.net/
[14] http://amarisoft.com
[15] http://www.softwareradiosystems.com/
[16] https://www.crowdsupply.com/lime-micro/limenet

in the receiver is roughly the same but in reverse order. The USRP board continuously listens for new data and forwards them to the energy detector. Then, the energy detector is responsible for identifying the signal and update the buffer for signal processing. After the required processing is completed the data are sent to the application directly. Similar to the transmitter, a potential future work direction could be the implementation of a virtual interface which will drive the data directly to the application.

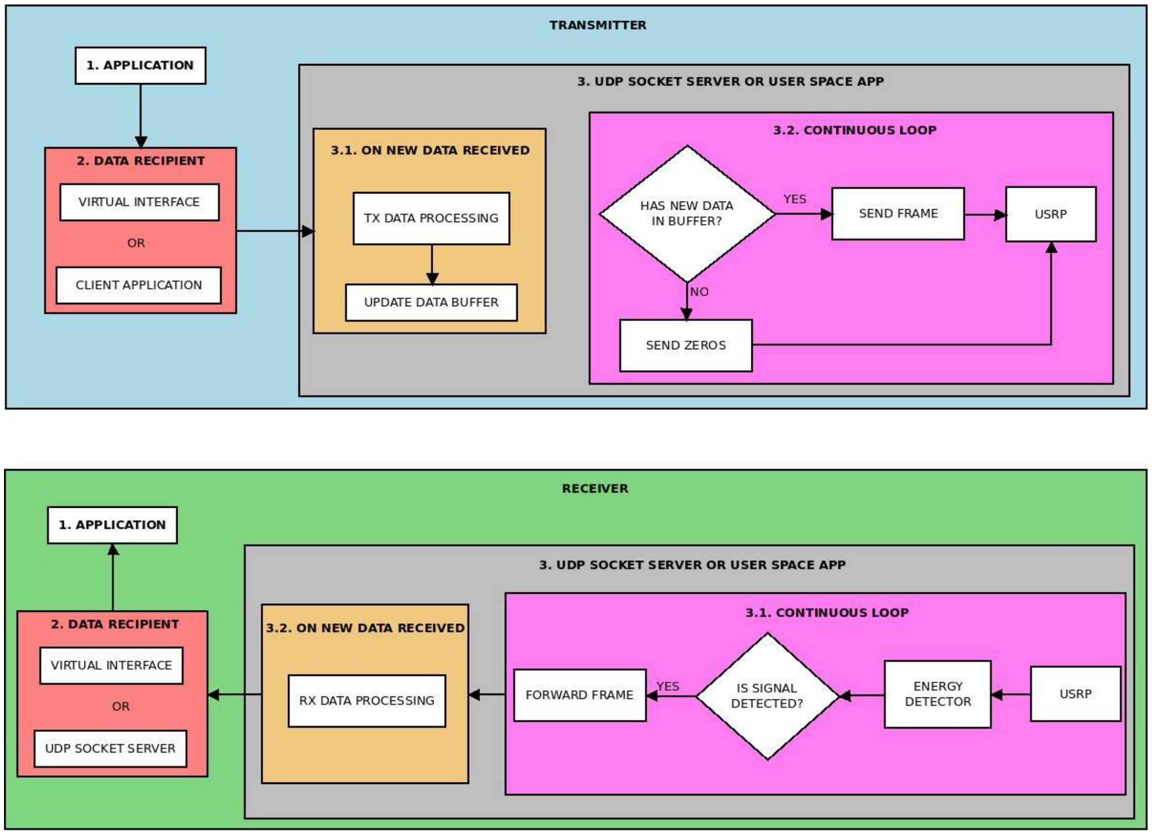


Figure 5: Software Modem Architecture Outline: Transceiver & Interfaces with SDR and Application

C. Building Blocks Detailing & System Optimization

The software modem may be divided into three logical entities (Figure 6) following the defined architecture:

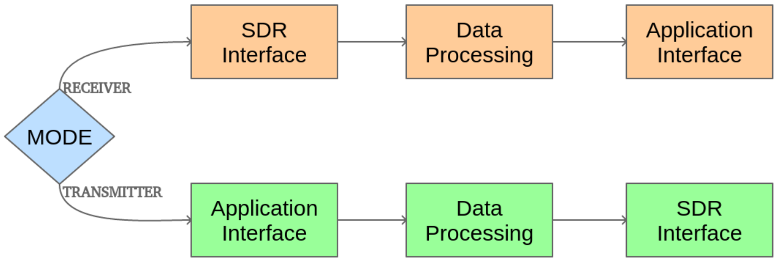


Figure 6: High level presentation of transceiver entities

The radio hardware or SDR interface is responsible for handling any kind of interaction between the modem and the SDR, including the reception and transmission of data. Next, data processing is responsible for the implementation of the D2D radio protocol specifications for both transmitting and receiving modes, and finally, the application interface will communicate with the data processing entity to receive and forward blocks. From a high-level perspective, the transmitter and receiver modes are similar but in reverse order.

1) Receiver

The most intensive tasks belong to receiver functionality, as the receiver is responsible to be (time) aligned with transmitter, apart from the conventional data processing of each received block. The following diagram illustrates all the receiver elements of the software modem.

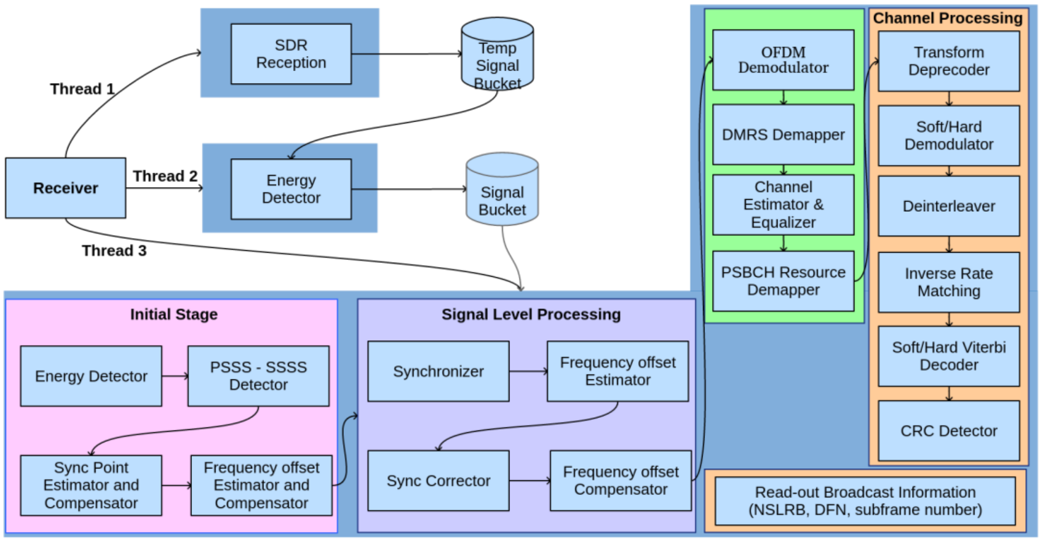


Figure 7: Software Modem Detailed Architecture: Receiver

The receiver mode is a quite demanding processing, and that is why it was decided to be divided into three different logical threads which are able to communicate, i.e. the "SDR Reception", the "Signal Energy Detector", and the "Data Processing". It was necessary to adopt a methodology where the threads shall be capable of communicating among each other. The most efficient way to do that, is instead of continuously looping and looking on global variables, to establish a publisher – subscriber protocol inside the receiver implementation between the different threads. It is not so popular in C/C++ as in many modern languages, however, it is already introduced by the Boost library and "condition_variable", which is also adopted in C++11. Therefore, the combination of "condition_variable" and mutex, alongside with scoped locks, let us implement an efficient publisher – subscriber protocol in C++. The existence of multiple threads may cause multiple interactions within our program. In this direction, to eliminate these interruptions, a new implementation was added to allow thread CPU affinity [22].

2) Transmitter

The functionality of the transmitter is more straightforward than that of the receiver. More specifically, in the transmit mode, the software modem should generate the data block and forward it to the SDR for transmission. There are two issues in the transmitter design. Firstly, it is possible that if the processing chain is quite fast, then the USRP may not able to follow the transmission pace. A possible solution of using incremental insertions of processed signals inside a repository, would lead to infinite repository size which eventually may cause a memory leak. To prevent this, a similar publisher – subscriber logic is applied at the transmitter case as well, in order to keep the repository size at reasonable levels and provide a fast implementation with a low memory footprint. Secondly, according to the D2D protocol empty/zero subframes may occur during system operation as well, and consequently the USRP will not have anything to send, while it always assumes that data are available for transmission (at a constant pace). To overcome this issue, zero samples are sent instead.

The next diagram depicts the main operations of the transmitter.

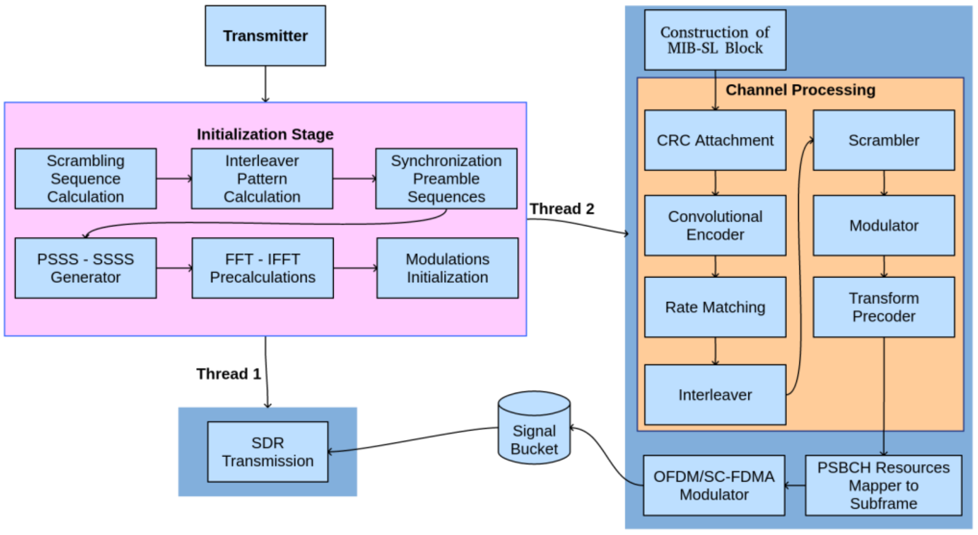


Figure 8: Software Modem Detailed Architecture: Transmitter

D. Runtime Performance Benchmarking

The most critical aspect of any software modem is its capability to cope with the tight real-time constraints imposed by the underlying radio protocol. In particular, assuming a fixed frame configuration, the modem should be able to perform complete packet processing, i.e. bit-level, symbol-level and signal-level functionalities, within specific time-limits determined by the frame duration. In case the USRP boards are used for over the air transmission/reception, as in the current project, then in the transmitter (receiver) side, each frame should be completely generated (recovered), within one system frame, in order to maintain real-time operation. For example, for LTE/D2D, if the transmitter (receiver) operates at a subframe level the packet generation (recovery) should last less than 1 msec, whereas for frame-level less than 10 msec. In the sidelink broadcast channel case a reference subframe is sent periodically every 40 msec, hence the transceiver should be at least able to follow this timing requirement.

In the following we present a collection of runtime performance benchmarking results for both the transmitter and the receiver parts of the developed software modem, specifically for the sidelink broadcast channel. We stress that these presented results refer only to the processing time spent for transmit and receive processing, and do not involve the interaction with the SDR board or higher layers, unless otherwise stated. The figures have been also broken down in important processing blocks in order to identify the potential real-time performance bottlenecks. Each presented value represents the processing time averaged over 500 test runs[17]. The results presented are obtained using five different types of typical local/remote/cloud-based GPP hosts/platforms, running Ubuntu desktop/server OS, in particular:

1) A high-end desktop platform ("FERON PC"), equipped with a high-performance quad-core Intel i7 CPU model, clocked at 3500 MHz, 16 GB RAM, and SSD storage.
2) A high-end headless desktop platform ("FLEX Node") part of the NITOS test-bed (Icarus node), equipped with an older generation quad-core Intel i7 CPU, clocked to 3400 MHz, less RAM (8 GB), and SSD storage.
3) A high-end remote virtual machine deployed in the Microsoft Azure Cloud ("Azure VM") and provided by FERON, equipped with an Intel Xeon CPU clocked at 2400 MHz, 4 GB RAM, and SSD storage.
4) A modern x86 single-board-computer (UPboard[18]), hosting a quad-core Intel Atom CPU, clocked up to 1920 MHz (typical speed: 1440 MHz), 2 GB RAM, and eMMC storage.
5) A lower performance x86 single-board-computer (Minnow-Turbot[19]), hosting a dual-core Intel Atom CPU, clocked up to 1460 MHz, 2 GB RAM, and SATA storage.

Detailed benchmarking results for the different platforms are presented in Table 8 for the transmitter and receiver software modem parts, along with the specifications of each platform. Further results are given in Figure 9 - Figure 14. The main findings for the receiver side are reported below:

- The high-end local desktop platform exhibits the optimal run-time performance for both initialization and continuous operation stages. The initialization stage lasts for almost 7 msec, meaning that an initial delay of 7 D2D reference subframes is expected. However, this is not critical, since the particular stage does not impact the continuous modem operation, but is executed once when the modem is "turned-on". The processing time for the continuous operation stage is less than 3 msec. This means that for the sidelink broadcast channel the protocol timing requirement of 40 msec is well above the frame average processing time.
- The remote FLEX NITOS node exhibits a slightly worse performance than the local desktop platform. In particular, 2% higher processing time for the initialization stage and 12% higher time for the continuous operation mode are reported. The Azure Cloud VM platform exhibits approximately 40% higher processing times than the local desktop platform.
- Two low-end single board computing platforms (UPboard and Minnowboard) based on Intel Atom CPU models have been also tested. Both platforms exhibit significantly lower performance, approximately an order of magnitude (factor 10) higher processing times. However, it is quite interesting that both boards are able to cope with the sidelink broadcast channel timing requirements.
- For all the platforms, the continuous operation stage processing is dominated by two signal-level operations, time synchronization and frequency offset compensation. Both operations consume approximately the 85-90% of the total processing time. For the desktop platforms the time is almost equally distributed to the two operations, whereas for the SBC-based hosts the time spent for the synchronization step is 2 times higher than that of frequency offset compensation. The frequency offset compensation high complexity is attributed to the usage of multiple exponential calculations. Potential improvements will be considered in future versions of the modem, for example the application of numerical methods and approximations for performing frequency offset compensation.
- Focusing on the bit-level and symbol-level processing stages, i.e., transport and physical channel processing, we notice that the lion's share of the processing time is attributed to three sub-processes: i) channel estimation & equalization (~45%) ii) viterbi decoding (~20-30%), and iii) OFDM demodulation (20-30%). The remaining sub-processes, i.e. transform deprecoding, QPSK demodulation, descrambling, de-interleaving, rate matching recovery, and CRC detection, consume only 5% of overall transport/physical channel processing time.

[17] It has been empirically observed that 500 repetitions are more than enough to reach statistically acceptable convergence levels.
[18] http://www.up-board.org/
[19] https://www.minnowboard.org/

Table 8: D2D Software Modem Transceiver Runtime Benchmarking Results

(Times in μsec unless otherwise stated)	***FERON Desktop PC***	***FLEX NITOS USRP Node***	***Azure Cloud VM***	***Up-board x86***	***Minnowboard Turbot x86***
RECEIVER					
Detection and Synchronization	6812	6978	9779	64059	74517
Synchronization	1007.91	1183.37	1411.89	11956.50	13738.9
Frequency Offset Estimation	3.51	4.20	4.83	43.54	49.12
Frequency Offset Compensation	1080.57	1165.68	1486.80	5440.01	6345.1
OFDM Demodulation	65.54	68.65	93.39	566.41	670.80
Channel Estimation & Equalization	148.17	163.08	213.30	836.70	976.47
Transform Deprecoding	5.62	6.01	7.87	52.90	67.69
QPSK Soft Demodulation	0.99	1.05	1.49	5.27	6.22
Descrambling	2.41	2.73	3.58	13.92	17.51
Deinterleaving	1.26	1.35	1.67	7.68	8.6
Rate Matching Recovery	4.682	4.86	6.99	44.27	50.01
Soft Viterbi Decoding	94.03	115.38	136.88	376.42	437.52
CRC detection	0.518	0.71	1.02	7.49	7.89
(Decoding Quality Estimation)	118.68	129.73	169.85	807.27	927.18
initialization stage (msec)	*6.81*	*6.98*	*9.78*	*64.06*	*74.52*
continuous operation stage (msec)	*2.42*	*2.72*	*3.37*	*19.35*	*22.38*
init x (compared to best)	*1.00*	*1.02*	*1.44*	*9.40*	*10.94*
co x (compared to best)	*1.00*	*1.12*	*1.40*	*8.01*	*9.26*
TRANSMITTER					
MIB-SL Encoding	0.40	0.57	0.64	3.41	7.23
CRC Attachment	0.38	0.53	0.61	2.93	7.92
Channel Encoding	2.66	2.92	4.16	16.73	31.37
Rate Matching	5.69	6.28	7.62	46.39	56.65
Interleaving	1.36	1.38	1.93	10.68	12.86
Scrambling	1.33	1.44	1.88	6.76	8.26
QPSK Modulation	1.63	1.66	2.22	9.39	11.97
Transform Precoding	5.39	5.36	7.39	46.43	64.18
Resources Mapping	7.43	7.70	10.46	34.06	50.68
OFDM Modulation	112.36	115.35	156.16	973.82	1240.79
transport channel processing (μsec)	*10.47*	*11.68*	*14.97*	*80.14*	*116.02*
physical channel processing (μsec)	*15.78*	*16.17*	*21.95*	*96.64*	*135.09*
signal level processing (μsec)	*112.36*	*115.35*	*156.16*	*973.82*	*1240.79*
total time (msec)	*0.14*	*0.14*	*0.19*	*1.15*	*1.49*
total time x (compared to best)	*1.00*	*1.03*	*1.39*	*8.30*	*10.76*
Platform Specifications					
Type	Desktop	Remote Node	Cloud VM	SBC	SBC
CPU model	Intel Core i7-4770K CPU	Intel Core i7-3770 CPU	Intel Xeon CPU E5-2673 v3	Intel Atom x5-Z8350	Intel Atom CPU E3826
CPU count/Cores/Threads Per Core	4/4/1	4/4/1	2/2/1	4/4/1	2/2/1
CPU frequency	3500 MHz	3400 MHz	2400 MHz	1440 MHz	1460 MHz
Cache	8 MB	8 MB	30 MB	1 MB	512k
RAM	16 GB	8 GB	4 GB	2 GB	2 GB
OS	Ubuntu Desktop 14.04	Ubuntu Server 14.04	Ubuntu Server 14.04	Ubuntu Server 14.04	Lubuntu

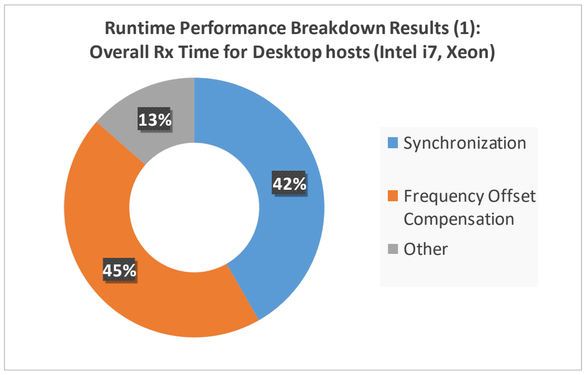


Figure 9: D2D Rx soft-modem runtime performance breakdown for overall processing time: Desktop hosts

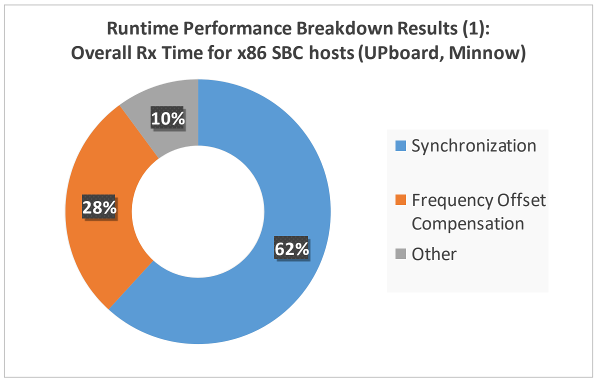


Figure 10: D2D Rx soft-modem runtime performance breakdown for overall processing time: SBC hosts

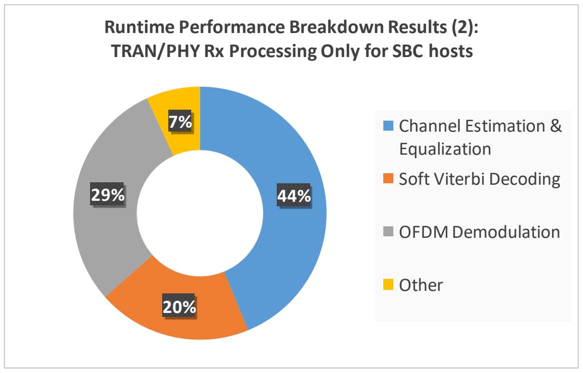


Figure 11: D2D Rx soft-modem runtime performance breakdown for TRAN/PHY processing: Desktop hosts

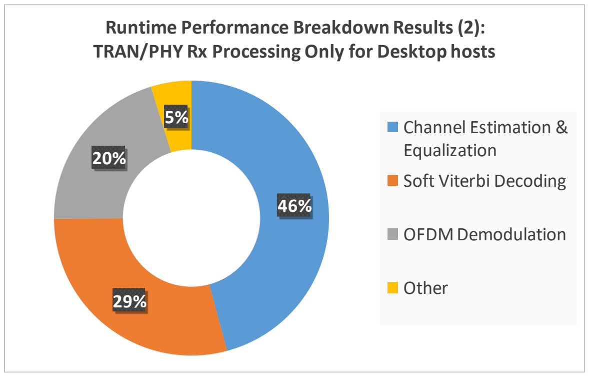


Figure 12: D2D Rx soft-modem runtime performance breakdown for TRAN/PHY processing: SBC hosts

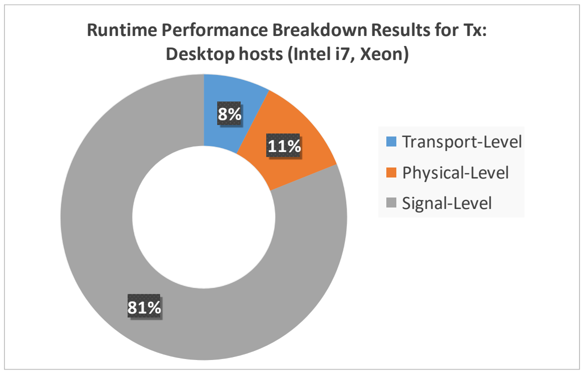


Figure 13: D2D Tx soft-modem runtime performance breakdown for TRAN/PHY processing: Desktop hosts

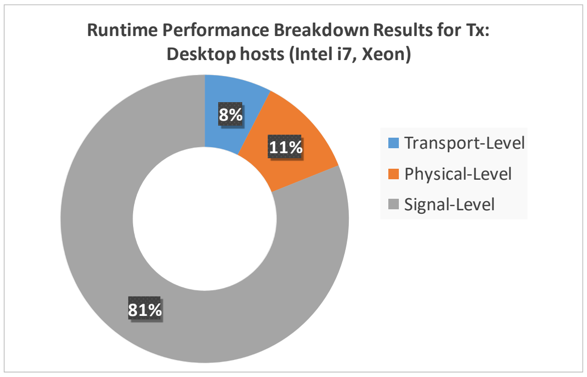


Figure 14: D2D Tx soft-modem runtime performance breakdown for TRAN/PHY processing: SBC hosts

E. Over-the-air Evaluation in FLEX NITOS Test-bed

FLEX (http://www.flex-project.eu) is an EU FP7 Collaborative Research Project, part of the FIRE ("Future Internet Research Experimentation") programme, which promotes experimentation-driven research by providing to external users an open and highly configurable experimental facility that uses LTE resources. FLEX's experimentation environment features both open source platforms and configurable commercial equipment that span macro-cell, pico-cell and small-cell setups. NITOS test-bed is part of the FLEX project and enables remote experimentation with SDR equipment as well. In this subsection we present over-the-air link-level evaluation results obtained using the NITOS test-bed as in Figure 15.

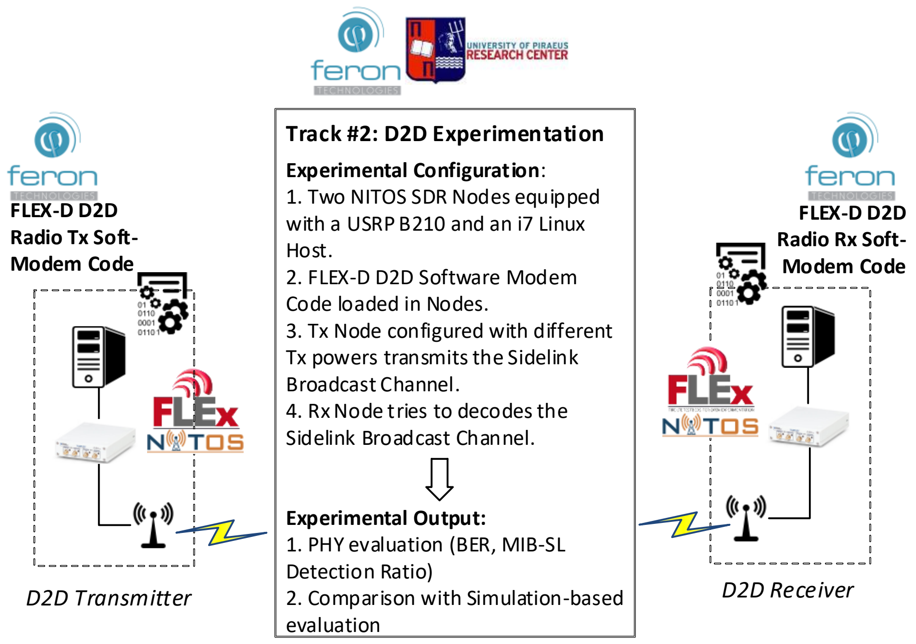

feron TECHNOLOGIES
UNIVERSITY OF PIRAEUS RESEARCH CENTER
FLEX-D D2D Radio Tx Soft-Modem Code
FLEX-D D2D Radio Rx Soft-Modem Code
FLEx NITOS
D2D Transmitter
D2D Receiver
Track #2: D2D Experimentation
Experimental Configuration:
1. Two NITOS SDR Nodes equipped with a USRP B210 and an i7 Linux Host.
2. FLEX-D D2D Software Modem Code loaded in Nodes.
3. Tx Node configured with different Tx powers transmits the Sidelink Broadcast Channel.
4. Rx Node tries to decodes the Sidelink Broadcast Channel.
Experimental Output:
1. PHY evaluation (BER, MIB-SL Detection Ratio)
2. Comparison with Simulation-based evaluation

Figure 15: D2D Software Modem Experimental Setup in FLEX NITOS Test-Bed

A baseline D2D radio link has been configured and tested in the NITOS platform, using two USRP-equipped nodes. Currently, the sidelink broadcast channel is fully supported by the real-time software modem implementation, thus we have performed tests only for the specific mode. The particular experimental evaluation has a significant added value compared to the simulation-based evaluation performed in Section III.C, as it enables:

- The assessment of signal detection and time recovery capabilities of the D2D radio receiver;
- The assessment of frequency-offset estimation/compensation mechanisms accuracy, since real hardware modules induce signal non-idealities in the tested transceiver chain (“hardware-in-the-loop”);
- The overall receiver performance evaluation in real –and not model-based– channel conditions, and the comparison with simulated-based studies.

In particular:

- A NITOS node has been used for hosting the transmitting D2D radio and another one for hosting the receiving D2D radio. Both nodes are equipped with USRP SDR boards, which support the standard LTE/D2D sampling rates.
- Tests have been performed for a carrier frequency of 2560 MHz; this is the UL frequency of Band 7 EARFCN 3350 (DL)/21350 (UL) FDD paired carrier.
- The receiving USRP gain has been fixed to 40 dB, whereas the transmitting gain (thus the D2D transmission power level) has been varied from 75 dB down to 39.5 dB, in order to cover a wide dynamic range (~35 dB) of signal reception levels, and investigate the PHY performance even at very extreme signal conditions.
- For each Tx gain configuration we continuously transmit a sidelink broadcast waveform and record at the receiving node 2,000 I/Q blocks as coming from the USRP board. Then, the soft-modem is applied for recovering the sidelink broadcast subframes information, i.e. MIB-SL, which carries the bandwidth mode and the sidelink frame/subframe timing. Based on decoding outcomes the following PHY metrics are extracted as a function of estimated (EVM-based) SNR:
    - PSBCH Bit Error Rate;
    - MIB-SL Success Detection Ratio;

The evaluation results are presented in Figure 16 along with results obtained from simulation for the sidelink broadcast channel in Section III.C. Based on the experimental outcomes the following are concluded:

- A wide system operation range has been indeed experimentally evaluated as demonstrated by the depicted estimated SNR range.
- The experimental (over-the-air) performance is slightly worse than that of the simulated performance under noise channel conditions but better than the simulated performance under time-varying fading channel. The former is expected since hardware non-idealities induce non-AWGN signal corruption effects, which are not perfectly compensated during the recovery process (synchronization, frequency offset compensation, and channel estimation/equalization). The latter is attributed to the stationary channel conditions prevailing in the test-bed.

- The experimentally evaluated MIB-SL detection ratio performance is quite high. For estimated SNR conditions down to 6 dB, perfect detection is achieved. At 0 dB SNR the detection ratio is still very high (~80-90%) and even at -3 dB SNR the detection ratio is 50%.

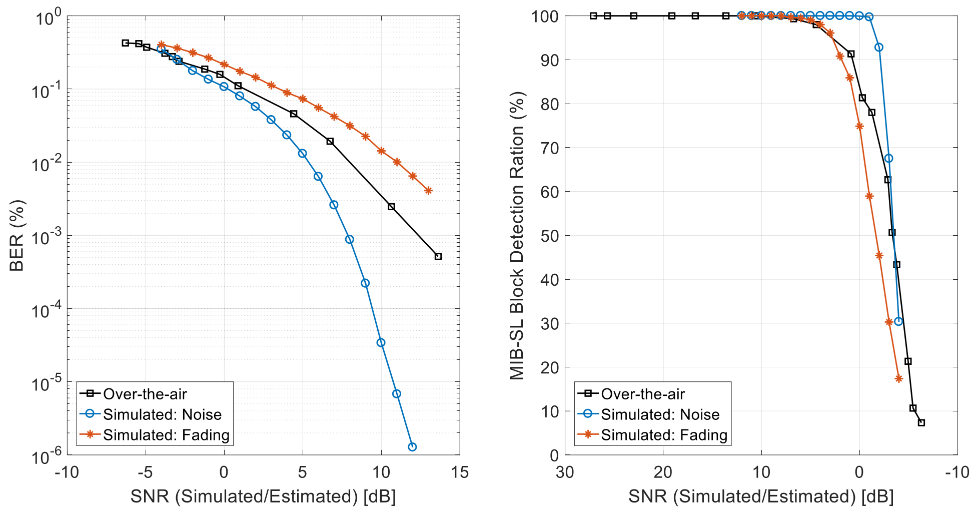


Figure 16: D2D Software modem experimental-based evaluation for the sidelink broadcast channel

V. Summary & Outlook

In this contribution we thoroughly analyzed the prospects of device-to-device communication introduction in cellular LTE technology networks. We first provided the necessary background and history details on D2D consideration within the 3GPP standards, pointing out the fact that D2D has been continuously evolving from a public safety facilitator to a key vertical domain (V2X, IoT) enabler. We continued with a detailed presentation of the core D2D radio protocol specifications, organized in a way that if one follows it step-by-step, he/she will come up with a complete view of the D2D end-to-end transceiver processing and the LTE-D2D coordinated resource allocation as stated in the standard. Next, we presented our open MATLAB software library ("lte-sidelink") which implements (almost all) the D2D radio functionalities. The library is available as open-source code in the most popular worldwide online repository (Github). Building on our in-house library we then discussed the current status and ongoing activities around the development of a real-time D2D software modem running in GPP hosts equipped with general purpose SDR boards. We presented the key design factors, the architecture, the development challenges, the core building blocks and indicative performance evaluation results for a first prototype realizing the sidelink broadcast operation mode.

We conclude by reporting a list of potential refinements, modifications, or extensions of the current contribution. First, we plan to extend the "lte-sidelink" library with new features as they become available through 3GPP release updates. Currently the focus of the library has been on core D2D features of Release 12 and 13, i.e. sidelink broadcast, discovery and communication channels/modes. New features as of Release 14 (June 2017) targeting vehicle-to-vehicle communications have been also incorporated but not fully supported yet. It is already planned that upcoming versions of the library will fully support V2V enhancements and also start considering D2D-related enhancements for the next 3GPP release (Rel.15) planned for June 2018. In addition, we are considering the extension of the D2D software modem prototype with new functionalities. At the current stage the modem fully supports the sidelink broadcast communication mode. Other sidelink modes, i.e. discovery & communication which are already supported by the corresponding software library will be also integrated to the modem. Emphasis will be put on the support of sidelink enhancements for V2X communications, a vertical sector entailing a plethora of research & innovation challenges and business opportunities.

Acknowledgment

The research leading to these results has received funding from the European Union's Seventh Framework Programme under grant agreement no 612050 (FLEX Project). The presented activities have been performed in the context of FLEX Open Call 2 Project "FLEX-D: Experimenting with Flexible D2D communications Over LTE".

References

[1] E. Dahlman, S. Parkvall, J. Skold, "4G LTE-Advanced Pro and The Road to 5G," 3nd edition, Academic Press, 2016.
[2] Rohde & Schwarz, "LTE-Advanced (Release 12) Technology Introduction," White Paper, Sep. 2014.
[3] Rohde & Schwarz, "Device to Device Communication in LTE," White Paper, Sep. 2015.
[4] Rohde & Schwarz, "LTE-Advanced (Release 12) Technology Introduction," White Paper, Sep. 2014.
[5] S.-Y. Lien, C.-C. Chien, F.-M. Tseng, Tien-Chen Ho, "3GPP device-to-device communications for beyond 4G cellular networks, "in IEEE Communications Magazine, vol.54, issue 3, pp. 29-35, Mar. 2016.

[6] 3GPP TR 22.803, “Feasibility study for Proximity Services (ProSe) (Release 12),” TR v.12.2.0, Jun. 2013.
[7] 3GPP TR 36.843, “Study on LTE Device to Device Proximity Services; Radio Aspects (Release 12)” TR v12.0.1, Mar., 2014
[8] 3GPP TR 36.877, “Study on LTE Device to Device Proximity Services; User Equipment (UE) radio transmission and reception (Release 12)” TR v12.0.0, Mar., 2014
[9] 3GPP RP-161839, “Further Enhancements to LTE Device to Device, UE to Network Relays for IoT and Wearables (FS_feD2D_IoT_relay_wearable), “3GPP TSG RAN Meeting #73, New Orleans, September 19 - 22, 2016
[10] 3GPP TS 36.211, “Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation (Release 14)” TS v14.0.0, Sep., 2016
[11] 3GPP TS 24.334, “Proximity-services (ProSe) User Equipment (UE) to ProSe function protocol aspects; Stage 3 (Release 13),” TS v.13.3.0, Mar. 2016
[12] 3GPP TS 23.003, “Numbering, addressing and identification (Release 13),” TS v.13.6.0, Jun. 2016
[13] 3GPP TS 36.212, “Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (Release 14)” TS v14.0.0, Sep., 2016
[14] 3GPP TS 36.213, “Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures (Release 14)” TS v14.0.0, Sep., 2016
[15] 3GPP TS 36.321, “Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification (Release 14)” TS v13.0.0, Sep., 2016
[16] 3GPP TS 23.303, “Proximity-based Services (ProSe); Stage 2 (Release 14),” TS v.14.1.0, Dec. 2016
[17] "CISQ Specifications for Automated Quality Characteristic Measures", OMG 2012
[18] "The GNU C Programming Tutorial", Edition 4.1
[19] Sandra Loosemore with Richard M. Stallman, Roland McGrath, Andrew Oram, and Ulrich Drepper, "The GNU C Library Reference Manual", version 2.24
[20] Alan Thomas, CCNA, CCSI, Global Knowledge Instructor, "The Packet Delivery Process: Remotely Connected Hosts", 2014
[21] Jin-Soo Kim, Kangho Kim, Sung-In Jung (2001), "Building a high-performance communication layer over virtual interface architecture on Linux clusters", Proceedings of the 15th international conference on Supercomputing, ACM: 335–347, 2009-02-09
[22] Robert Love, "Kernel Korner CPU Affinity", Linux Journal #111, July 2003
[23] R. A. Shafik, M. S. Rahman and A. R. Islam, "On the Extended Relationships Among EVM, BER and SNR as Performance Metrics," 2006 International Conference on Electrical and Computer Engineering, Dhaka, 2006, pp. 408-411.

Implementing NB-IoT in Software - Experiences Using the srsLTE Library

André Puschmann, Paul Sutton, Ismael Gomez
Software Radio Systems Ltd., Cork, Ireland
Email: {andre, paul, ismael}@softwareradiosystems.com

***Abstract*—NB-IoT is the 3GPP standard for machine-to-machine communications, recently finalized within LTE release 13. This article gives a brief overview about this new LTE-based radio access technology and presents a implementation developed using the srsLTE software radio suite. We also carry out a performance study in which we compare a theoretical analysis with experimental results obtained in our testbed. Furthermore, we provide some interesting details and share our experience in exploring one of the worldwide first commercial NB-IoT deployments.**

***Keywords*—NB-IoT, LTE, Software Defined Radio, srsLTE**

I. Introduction

Narrowband Internet of Things (NB-IoT) has been standardized by 3GPP in Release 13 as a new air interface with the aim to make LTE suitable for low-cost massive machine-type communication (m-MTC). NB-IoT is heavily based on LTE but makes a number simplifications and optimizations in order to reduce device costs, minimize power consumption and to make it work in unfavourable radio conditions.

This work presents an implementation of the NB-IoT standard, developed using the srsLTE software radio suite within the FP7 FLEX-IoT project. Since 2008, the Future Internet Research and Experimentation (FIRE) initiative has bridged the gap between visionary research and large-scale experimentation on new networking and service architectures and paradigms. The FLEX (FIRE LTE testbeds for open EXperimentation) project provides an open and operational LTE experimental facility based on a combination of configurable commercial equipment, configurable core network software, open-source components, and sophisticated emulation and mobility functionalities. Under the FLEX-IoT project, we have integrated our high performance software radio UE (srsUE) as an extension to the FLEX testbed facilities and extended the FLEX testbed facilities to include the NB-IoT features of 3GPP LTE release 13 for LPWAN. For an excellent overview about NB-IoT in general and it's differences to LTE in particular the interested reader is referred to [1] and [2].

In this report we provide a brief overview about NB-IoT and describe the available channels and signals in Section II. We then discuss the architecture of srsLTE and how the new NB-Iot components have been integrated into it. The main contributions of this article is a performance analysis carried out in Section IV. We study the achievable throughput in the downlink and compare results obtained from practical experiments with those from a theoretical analysis. In Section V we share our experience in exploring one of the first commercial NB-IoT deployments. Finally, we conclude the paper in Section VI.

II. NB-IoT Overview

This section provides a brief overview about NB-IoT, it's transmission schemes, downlink (DL) and uplink (UL) channels and signals and the different deployment schemes.

A. Transmission schemes, time structure and deployment options

NB-IoT uses the same downlink transmission scheme like normal LTE with OFDMA with a subcarrier spacing of 15 kHz. 12 subcarriers are combined to form a single carrier with a bandwidth of 180 kHz, i.e., NB-IoT only uses one physical resource block (PRB) instead of up to 100 PRBs. In the time domain, it also uses the same frame structure with frames of 10 ms length. Each frame consists of 10 subframes of 1 ms length and each subframe consists of 2 slots with a length of 7 OFDM symbols each. NB-IoT only defines the normal cyclic prefix. In addition to that, NB-IoT also defines the concept of so-called hyper frames. Similar to the system frame number (SFN), the hyper frame number (HFN) is also a ten-bit wide number that is incremented whenever the SFN wraps around, i.e., every 10240 ms.

In the uplink, NB-IoT provides two options: single and multi-tone transmissions. In the latter case, which is the default option, it uses the same SC-FDMA scheme with a 15 kHz subcarrier spacing and a total bandwidth of 180 kHz as LTE.

NB-IoT offers multiple deployment options. It can be deployed in standalone mode, i.e., without any pre-existing LTE network, or together with an already deployed LTE network. The latter has the advantage that existing sites can be used to roll out an IoT network without heavy investments in new hardware. In this case, the NB-IoT carrier can be placed inside one of the subcarriers of an existing LTE cell or in the guardbands of a cell. In Release 13, 3GPP has also standardized a new UE device category for NB-IoT, called Cat-NB1.

B. Downlink

Release 13 of the 3GPP standard defines six physical DL channels/signals for NB-IoT, all of which will be briefly described below [3]. Because NB-IoT only uses a single PRB, the channels are only multiplexed in time-domain. Figure 1 depicts how all DL signals are transmitted over the length of two frames.

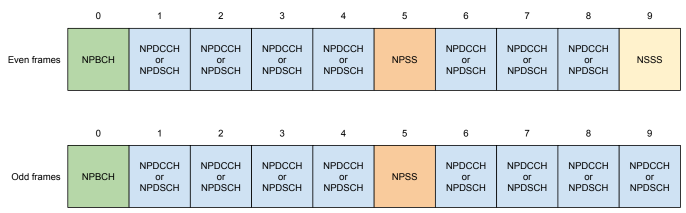


Fig. 1: NB-IoT downlink frame structure and time multiplexing (reproduced from [1]).

1) Narrowband Primary Synchronization Signal (NPSS): The NPSS, as well as the NSSS described below, are primarily used for initial cell search and for acquiring time and frequency synchronization. The NPSS is transmitted in every frame in subframe number 5. It only uses 11 of the 14 available OFDM symbols of the subframe to protect a potential LTE transmission in the same subframe. The signal itself consists of a single length-11 Zadoff-Chu (ZC) sequence that is either directly mapped to the 11 lowest subcarriers (the 12th subcarrier is empty in the NPSS) or is inverted before the mapping process. After successful detection of the NPSS, a NB-IoT UE is able to determine the frame boundaries of a DL transmission.

2) Narrowband Secondary Synchronization Signal (NSSS): The NSSS is transmitted every 20 ms in every even-numbered frame in subframe number 9. The signal consists of a length-132 sequence that is generated from a scrambling sequence and a length-132 ZC sequence. The NSSS only carries the cell ID.

3) Narrowband Reference Signal (NRS): The NRS is required by the UE to successfully demodulate any of the data or control channels. For this purpose, the eNB broadcasts a known pilot signal that can be used to estimate the DL channel. With this knows reference, the UE tries to correct the effects of the channel before demodulating the signal. The NRS is transmitted in every subframe carrying either the NBPCH, NPDCCH or the NPDSCH. It is distributed over 4 OFDM symbols (the last and second last symbol of each slot) and uses a total of 8 resource elements (RE) per antenna port.

4) Narrowband Physical Broadcast Channel (NPBCH): The NPBCH is the first physical channel that a UE attempts to receive and demodulate after synchronizing with a cell. The NPBCH conveys important information that is encoded in the Narrowband master information block (MIB-NB), such as the SFN (the four most significant bits), the HFN (the two least significant bits), system information block (SIB) scheduling information and the NB-IoT operation mode. In order to allow successful decoding of the MIB-NB also in extraneous radio conditions, it is transmitted in the first subframe of every ever frame over a total period of 640ms, i.e., over 64 frames. For this purpose, the MIB-NB signal is split into 8 blocks, each of which is repeated 8 times. The NPBCH is QPSK modulated and uses a tail-biting convolution encoding (TBCC).

5) Narrowband Physical Downlink Control Channel (NPDCCH): The NPDCCH is the only control channel defined for NB-IoT, i.e., there is now uplink counterpart like in normal LTE. It carries the DL and UL scheduling grants, acknowledgment information for UL transmissions, paging indication, system information update and random access response (RAR) scheduling information. For NB-IoT only three DL control information (DCI) formats are specified and only three possible positions exist in the NPDCCH. This simplifies the decoding overhead for the UE significantly compared to LTE. Like the NPBCH, the NPDCCH is also QPSK modulated and protected using TBCC.

6) Narrowband Physical Downlink Shared Channel (NPDSCH): The NPDSCH is the main data channel and carries SIBs, upper layer data, and random access response (RAR) messages.

C. Uplink

The 3GPP set of specifications defines two uplink channels, the Narrowband Physical Random Access Channel (NPRACH) and the Narrowband Physical Uplink Shared Channel (NPUSCH). It is interesting to note that there is no equivalent to the Physical Uplink Control Channel (PUCCH) of LTE in NB-IoT. In addition to that, NB-IoT defines the Narrowband demodulation reference signal (DMRS).

1) Narrowband Physical Random Access Channel (NPRACH): As in normal LTE, the NPRACH is used by an UE to signal the eNB that it wants to establish a connection with it. The attachment procedure is a contention-based access method in which the UE transmits a random access preamble. One such preamble consists of four symbol groups each of which is transmitted on a different subcarrier. Each symbol group in turn consists of the cyclic prefix (CP) plus 5 symbols, with a total length of either 1.4 ms or 1.6 ms, depending on the format defined by the eNB. In order to increase the coverage the eNB may request a UE to repeat the preamble transmission up to 128 times. The subcarriers used to transmit the NPRACH follow a certain hopping pattern that depends on a randomly selected initialization value.

2) Narrowband Physical Uplink Shared Channel (NPUSCH): The only means to communicate data from the UE to the eNB, except for the random access, is over the NPUSCH. There exist two formats, NPUSCH format 1 is used to carry user data, whereas format 2 is used to carry uplink control information. Upon reception of the preamble, the eNB responds with the random access response (RAR) which initiates the transmission of the first scheduled transmission by the UE (Msg3). After the contention resolution sent from the eNB, the NB-IoT random access procedure is complete.

3) Narrowband demodulation reference signal (DMRS): In order for the eNB to be able to estimate the UL channel, the UE transmits a demodulation reference signal (DMRS) that is multiplexed with the actual NPUSCH symbols. Depending on the NPUSCH format, i.e. either format 1 or format 2, one or three symbols per SC-FDMA slot are used to transmit the DMRS. For example, for the case of format 1 NPUSCH with a subcarrier spacing of 15 kHz, the 4th symbol of every slots contains the DMRS symbols in all allocated subcarriers.

4) Uplink Baseband Signal Generation: In order to generate the SC-FDMA baseband signal for the NB-IoT UL it is important to note an interesting difference to normal LTE. In fact, each modulation symbol (both data and reference symbols) for all UL transmissions undergo a phase rotation that depends on the position of the particular symbol. This reduces

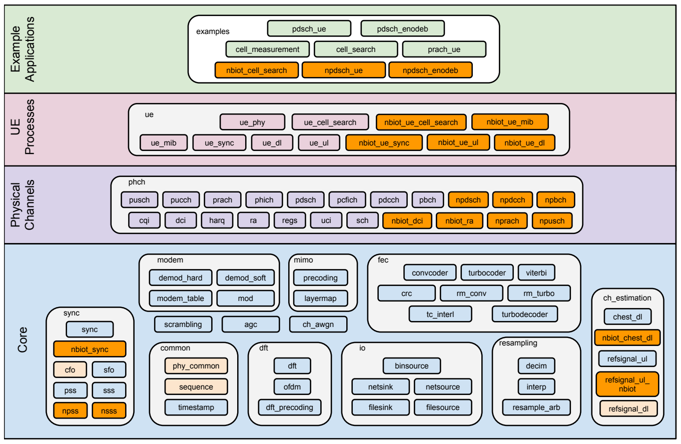


Fig. 2: srsLTE's modular architecture with new NB-IoT components.

the peak-to-average power ratio (PAPR) because it effectively yields a $\pi/2$-BPSK or $\pi/4$-QPSK modulation scheme with phase continuity between symbols, i.e., no zero-crossing in the constellation diagram between symbols.

III. SRSLTE

srsLTE is a high-performance open-source software radio LTE physical layer library [4]. It is specifically designed to support new air interface modifications and enhancements using a modular architecture with clean inter-component interfaces.

A. *Library Structure*

Figure 2 shows the modular architecture of the srsLTE PHY-layer library. Functional modules are organized into a hierarchy from elementary DSP components at the bottom to physical channels, processes and example applications at the top.

B. *NB-IoT Extension*

As part of the FLEX-IoT project, srsLTE has been extended to support the new NB-IoT radio access technology. In doing so, we leveraged from the great modularity and extensibility of the library and consequently followed the same approach for the NB-IoT physical layer components, including all DL/UL channels and signals defined in the standard. All orange blocks in Figure 2 have been added to srsLTE during the course of the NB-IoT extension.

srsLTE provides two example applications that act as PHY-only NB-IoT eNodeB and UE, similar to the already existing counterparts for LTE. The `npdsch_enodeb` example acts as an eNodeB by accepting data traffic on a TCP socket and transmitting it using the NB-IoT downlink physical channels. The number of subframes and the Transport Block Size (TBS) used for the downlink transmissions can be dynamically changed by the user through the command line.

The `npdsch_ue` example acts as the counterpart and provides a PHY-only UE that receives and decodes the DL transmission of the eNB.

IV. PERFORMANCE EVALUATION

This section evaluates the achievable downlink throughput in NB-IoT. We carry out a theoretical analysis first before comparing the results with experimental data obtained from our NB-IoT implementation.

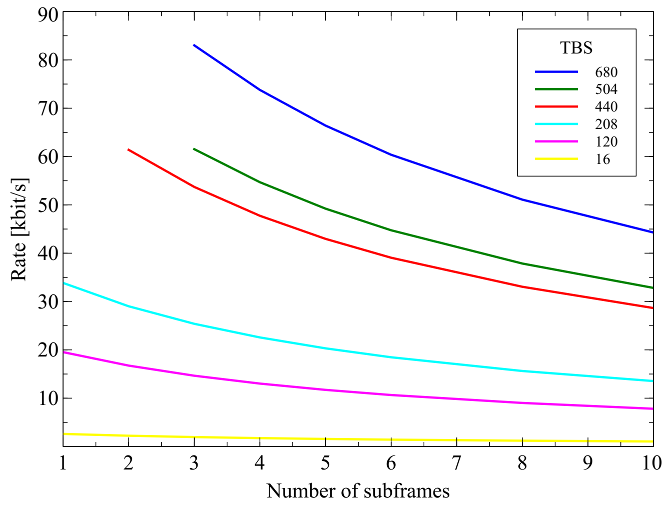
A. *Theoretical Analysis*

As for normal LTE, data to be transmitted at the physical layer in NB-IoT needs to have a size of a so-called transport block (TB). The transport block size (TBS) for the DL ranges between 16 and 680 bits. Depending the configuration, a single TB may be carried by one or more subframes. Transporting the largest TBS of 680 bits requires at least 3 subframes. Therefore, the often stated peak data rate in the DL is 226 kbit/s. This value, however, is only of a theoretical nature as it doesn't include the overhead associated with any NPDSCH transmission (except for SIB transmissions). This overhead includes at least one subframe for the NPDCCH to carry the DL grant as well as a minimum of 4 empty subframes that allow the UE to receive and decode the grant and to prepare for the reception of the actual NPDSCH. In total, at least 8 ms are needed to transmit the full TB carrying 680 bits. Please note that in NB-IoT there can only be a single hybrid automatic repeat request (HARQ) process running at the same time. Therefore, interleaved NPDCCH/NPDSCH transmissions are not possible. We also assume only a single transmission of each NPDSCH, i.e., now repetitions. This results in a maximum achievable DL data rate in unacknowledged mode (UM) of 85 kbit/s. Depending on the actual scheduling constraints of the eNB and the number of subframes and repetitions for the NPDSCH, this number may be much lower in reality. Figure 3(a) shows the maximum achievable DL rates for a single UE in UM for a number of TBS sizes as a function of the number of NPDSCH subframes. Note that because not all TBS sizes are possible for every number of subframes, we selected the closest possible value for each configuration in order to make the results comparable to each other.

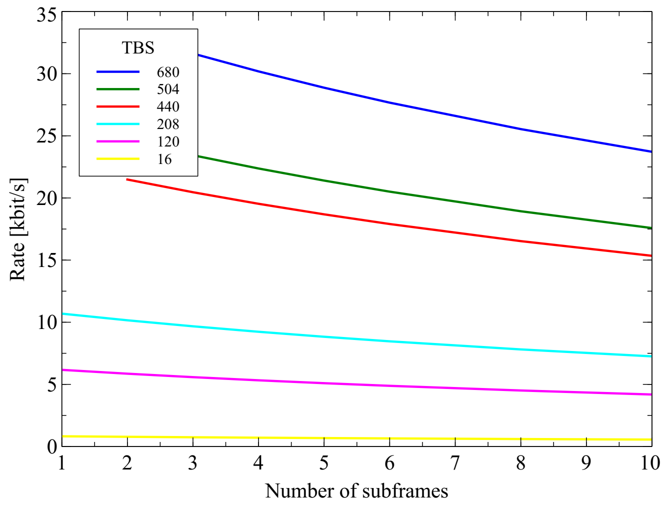
In acknowledged mode (AM), the actual achievable rate reduces even further than in UM because there is an extra gap between the NPDSCH (data) and the NPUSCH (acknowledgement) transmission of at least 12 subframes. Therefore, even in the best case, at least 21ms are required to transfer 680 bits worth of data, resulting in a maximum rate of 32,38 kbit/s. This again assumes a single NPDCCH subframe for the grant, a 4ms gap, 3 subframes for the NPDSCH, a 12ms gap and one subframe carrying the NPUSCH with the acknowledgement. Figure 3(b) plots the maximum achievable DL rate for a single UE in AM for a number of TBS as a function of the number of NPDSCH subframes.

B. *Experimental Results*

After the theoretical analysis, this section will now assess the performance of our prototype implementation and compare it against them. In practice, the above stated results (e.g., 85 kbit/s DL throughput) can only be achieved if the channel conditions between eNB and UE are excellent and the eNB is actually able to schedule transmissions for a specific UE using the minimum number of subframes to, e.g., transmit 680 bit of data in 8 subframes only. Using a single NB-IoT downlink



(a) Unacknowledged mode (UM).



(b) Acknowledged mode (AM).

Fig. 3: Maximum achievable DL rate for a single UE

carrier, however, this is difficult to achieve because the so-called anchor carrier requires specific subframes to be filled with periodic information, such as synchronization signals (i.e. NPSS and NSSS) as well as system information messages (i.e., MIB and SIBs). Those signals will always have higher priority as scheduling grants and user data carried over NPDCCH or NPDSCH, respectively. Our basic NB-IoT eNB example therefore uses a simple static resource allocation scheme in which a NPDCCH containing a DL grant is transmitted in subframe 1 of each frame and the corresponding NPDSCH from subframe 6 onwards in the same frame. Using this configuration allows us to transmit 680 Bits of data in subframe 6, 7 and 8 of each frame, i.e., every 10ms. Hence, using this resource allocation scheme, the maximum achievable DL rate is 68 kbit/s. Figure 4 shows the results of an experiment to assess the maximum achievable DL rate of our prototype implementation. The experiment was carried out under favourable radio conditions with an SNR of approximately 20 dB. We ran the experiment with three different NPDSCH configurations, i.e., with one, two, and three subframes. For each configuration, we also varied the TBS size from minimal to maximal setting (i_{tbs}) value. In total, we ran 36 experiments. Each run lasted about one minute, until the measured rate stabilized. From the plot it can be seen that in a high SNR regime with a NPDSCH bit error rate of 0%, the experimental results exactly match the theoretical calculation for each TBS. We are currently executing a set of simulations and experiments to evaluate the DL rate under lower SNR regimes.

V. Exploring Commercial Deployments

Several network operators around the world have chosen to adopt NB-IoT for MTC communication. Vodafone, for example, began to roll out their network in Spain in late January 2017, starting in Madrid and Valencia, but already announcing the expansion to other Spanish cities, such as Barcelona, Bilbao, Mlaga and Seville [5]. Only days later, on February 1st, we discovered a live NB-IoT carrier in downtown Barcelona in Vodafone's 800 MHz frequency band, which has been used from then on to test and validate standard conformance of our implementation.

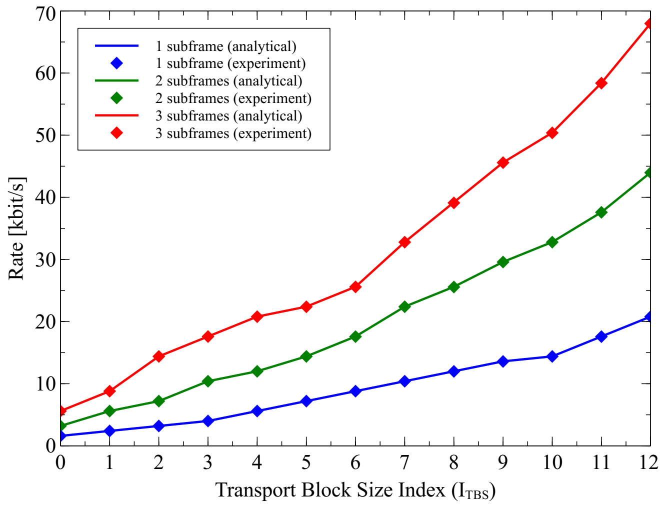


Fig. 4: Comparison of theoretical and experimental DL rate (20 dB SNR).

Figure 5 shows a power spectral density and waterfall plot of the aforementioned Vodafone LTE band centered at 806 MHz over the full cell bandwidth of 10 MHz. The NB-IoT carrier is clearly visible on the left-hand side of the spectrum. It is transmitted with slightly higher power than the rest of the LTE signal. Because the NB-IoT carrier is embedded in and uses resources of an existing LTE cell, it is said to operate in *in-band* mode.

During Mobile World Congress (MWC) 2017, SRS showcased a fully functioning DL receiver and decoder for NB-IoT. This demonstration not only decoded the live transmission of the Vodafone cell but also displayed various information about the signal, such as the Carrier Frequency Offset (CFO), Sample Frequency Offset (SFO) and Block Error Rate (BER) of the NPDSCH.

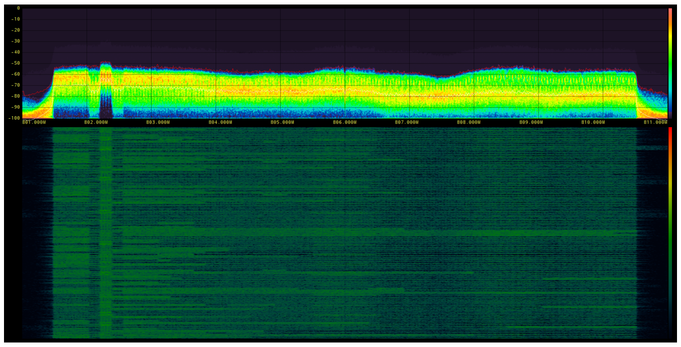
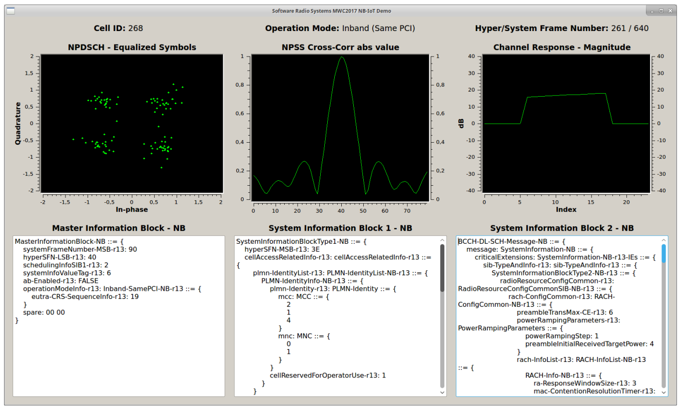


Fig. 5: Waterfall plot of the 10 MHz Vodafone cell at 806 MHz in Barcelona, Spain with the NB-IoT carrier clearly visible in the left part of the signal.


Software Radio Systems MWC2017 NB-IoT Demo
Cell ID: 268
Operation Mode: Inband (Same PCI)
Hyper/System Frame Number: 261 / 640
NPDSCH - Equalized Symbols
NPSS Cross-Corr abs value
Channel Response - Magnitude
Quadrature
In-phase
dB
Index
Master Information Block - NB
System Information Block 1 - NB
System Information Block 2 - NB

```
MasterInformationBlock-NB ::= {
    systemFrameNumber-MSB-r13: 90
    hyperSFN-LSB-r13: 40
    schedulingInfoSIB1-r13: 2
    systemInfoValueTag-r13: 6
    ab-Enabled-r13: FALSE
    operationModeInfo-r13: Inband-SamePCI-NB-r13 ::= {
        eutra-CRS-SequenceInfo-r13: 19
    }
    spare: 00 00
}
```

```
SystemInformationBlockType1-NB ::= {
    hyperSFN-MSB-r13: 3E
    cellAccessRelatedInfo-r13: cellAccessRelatedInfo-r13 ::=
{
        plmn-IdentityList-r13: PLMN-IdentityList-NB-r13 ::= {
            PLMN-IdentityInfo-NB-r13 ::= {
                plmn-Identity-r13: PLMN-Identity ::= {
                    mcc: MCC ::= {
                        2
                        1
                        4
                    }
                    mnc: MNC ::= {
                        0
                        1
                    }
                }
                cellReservedForOperatorUse-r13: 1
            }
        }
```

```
BCCH-DL-SCH-Message-NB ::= {
    message: SystemInformation-NB ::= {
        criticalExtensions: SystemInformation-NB-r13-IEs ::= {
            sib-TypeAndInfo-r13: sib-TypeAndInfo-r13 ::= {
                SystemInformationBlockType2-NB-r13 ::= {
                    radioResourceConfigCommon-r13:
RadioResourceConfigCommonSIB-NB-r13 ::= {
                        rach-ConfigCommon-r13: RACH-
ConfigCommon-NB-r13 ::= {
                            preambleTransMax-CE-r13: 6
                            powerRampingParameters-r13:
PowerRampingParameters ::= {
                                powerRampingStep: 1
                                preambleInitialReceivedTargetPower: 4
                            }
                            rach-InfoList-r13: RACH-InfoList-NB-r13
::= {
                                RACH-Info-NB-r13 ::= {
                                    ra-ResponseWindowSize-r13: 3
                                    mac-ContentionResolutionTimer-r13:
```

Fig. 6: Screenshot of the NB-IoT UE shown at MWC2017.

Figure 6 shows the graphical user interface of the UE displaying the constellation diagram of NPDSCH (upper left image), the correlation peak of the NPSS (upper middle image) and the channel response (upper right image). The UE also shows the content of MIB, SIB1 and SIB2 (bottom three text fields in Figure 6).

VI. Conclusion

With the finalization of Release 13 of the LTE Advanced Pro specification in June 2016, 3GPP has provided a new radio access technology based on LTE to address the requirements and challenges of the Internet of Things.

In this paper, we have summarized our efforts in developing a software implementation containing all relevant DL and UL channels and signals defined in the standard as well as example applications that facilitate testing and experimenting with this new technology.

Furthermore, the paper also described experiments carried out to assess the performance of the implementation and compares the obtained results with our analytical model. Our analysis showed that the theoretical peak data rate of 226 kbit/s in the DL cant be achieved in practice due to the control overhead associated with any transmission. In fact, the maximum achievable DL data rate for a single UE is 85 kbit/s in unacknowledged mode and 32.38 kbit/s in acknowledged mode, which can only be achieved with the highest modulation and coding scheme, using the lowest number of subframes and no repetitions. The maximum UL rate of 68 kbit/s achieved in our experiments can be viewed as an realistic upper limit in single carrier deployments. The 20% reduction compared to the theoretical maximum can be well explained by the overhead caused by the periodic DL transmissions, i.e., synchronization signals and system information messages.

Acknowledgments

The research and development leading to these results has received funding from the European Research Council under the European Community Seventh Framework Program (FP7/2014-2017) grant agreement 612050 (FLEX project).

References

[1] Y. P. E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi, “A Primer on 3GPP Narrowband Internet of Things,” *IEEE Communications Magazine*, vol. 55, no. 3, March 2017.

[2] Rohde & Schwarz, “Narrowband Internet of Things - Whitepaper,” 2016, available at http://www.rohde-schwarz.com/appnote/1MA266.

[3] ETSI, “TS 136 211: LTE: Evolved Universal Terrestrial Radio Access (E-UTRA): Physical channels and modulation,” 2016.

[4] Software Radio Systems, “srsLTE: Open Source LTE,” available at https://github.com/srsLTE.

[5] Vodafone, “Commercial launch of NB-IoT in Vodafone Spain,” 2017, available at http://www.vodafone.com/content/index/what/technology-blog/nbiot-commercial-launch-spain.html.