Universidade de Aveiro
2020

Departamento de Eletrónica,
Telecomunicações e Informática

Mohannad
Jooriah

Comunicações Veiculares Híbridas

Hybrid Vehicular Communications

Universidade de Aveiro
2020

Departamento de Eletrónica,
Telecomunicações e Informática

Mohannad Jooriah

Comunicações Veiculares Híbridas

Hybrid Vehicular Communications

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor Joaquim José de Castro Ferreira, Professor adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro, e do Doutor José Alberto Fonseca, Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Apoio financeiro da Plataforma Global de Apoio as Estudantes Sírios.

"Everyone you will ever meet, knows something you don't." –Bill Nye
My sincere gratitude to everyone, from whom I learned something.

o júri / the jury

presidente / president

Prof. Doutor António Manuel Melo de Sousa Pereira
Professor Catedrático, Universidade de Aveiro

vogais / examiners committee

Doutor Thierry Ernst
CEO, Yogoko

Prof. Doutor Joaquim José de Castro Ferreira
Professor Adjunto em Regime Laboral, Universidade de Aveiro

Palavras Chave ITS, Comunicaçoes Veiculares Híbridas, V2X, LTE, ITS-G5

Resumo

As Comunicações Veiculares são um campo de pesquisa promissor, com um grande potencial de desenvolvimento de novas aplicações capazes de melhorar a segurança nas estradas, a eficiência do tráfego, bem com o conforto e entretenimento dos passageiros. As tecnologias de comunicação veícular podem ser de curto alcance, como por exemplo ETSI ITS-G5 ou o canal PC5 do 5G, ou de longo alcance, recorrendo à rede celular (LTE ou 5G). No entanto, nenhuma das tecnologias por si só, consegue suportar a variedade expectável de aplicações para um número de veículos elevado nem tampouco todos os requisitos temporais e espaciais dos veículos conectados e autónomos. Assim, é proposto o uso colaborativo ou híbrido de comunicações de curto alcance, com latências menores, e de tecnologias de longo alcance, potencialmente com maiores latências, mas integrando dados agregados de maior abrangência geográfica.

Neste contexto, este trabalho apresenta um modelo de comunicações veiculares híbrido, capaz de fornecer conectividade por meio de duas Tecnologias de Acesso por Rádio (RAT), a saber, ETSI ITS-G5 e LTE, para aumentar a probabilidade de entrega de mensagens e, consequentemente, alcançar um sistema de comunicação veicular mais robusto, eficiente e seguro. A implementação de canais de comunicação de curto alcance é feita usando *Raw Packet Sockets*, enquanto que a ligação celular é estabelecida usando o protocolo *Advanced Messaging Queuing Protocol* (AMQP).

A contribuição principal desta dissertação foca-se no projeto, implementação e avaliação de uma sub camada hibrída de encaminhamento, capaz de isolar mensagens que se formam/descodificam a partir de processos de transmissão/receção. Esta camadada é, portanto, capaz de gerir o tráfego proveniente/destinado à camada de aplicação de sistemas inteligentes de transportes (ITS) adaptando e passando mensagens ITS entre as camadas mais altas da pilha protocolar e as tecnologias de acesso rádio disponíveis.

A sub camada hibrída de encaminhamento também potencia uma redução dos custos financeiros devidos ao uso de comunicações celulares e aumenta a eficiência do uso do espectro electromagnético disponível, ao introduzir um múdulo controlador da ligação celular, utilizando um *Beacon Detector*, que toma decisões informadas relacionadas com a necessidade de uma conexão a uma rede celular, de acordo com diferentes cenários.

Os resultados experimentais comprovam que as comunicações veículares híbridas cumprem os requisitos dos sistemas cooperativos de transporte inteligentes, ao tirarem partido das vantagens de ambas tecnologias de comunicação. Quando avaliadas de forma independente, constata-se que que a tecnologia ITS-G5 tem vantagens evidentes em termos de latência sobre a tecnologia LTE, enquanto que a tecnologia LTE tem melhor desempenho que a LTE, ai nível de débito e fiabilidade.

Keywords ITS, Hybrid Vehicular Communication, V2X, LTE, ITS-G5

Abstract

Vehicle Communications is a promising research field, with a great potential for the development of new applications capable of improving road safety, traffic efficiency, as well as passenger comfort and infotainment. Vehicle communication technologies can be short-range, such as ETSI ITS-G5 or the 5G PC5 sidelink channel, or long-range, using the cellular network (LTE or 5G). However, none of the technologies alone can support the expected variety of applications for a large number of vehicles, nor all the temporal and spatial requirements of connected and autonomous vehicles. Thus, it is proposed the collaborative or hybrid use of short-range communications, with lower latency, and of long-range technologies, potentially with higher latency, but integrating aggregated data of wider geographic scope.

In this context, this work presents a hybrid vehicle communications model, capable of providing connectivity through two Radio Access Technologies (RAT), namely, ETSI ITS-G5 and LTE, to increase the probability of message delivery and, consequently, achieving a more robust, efficient and secure vehicle communication system. The implementation of short-range communication channels is done using Raw Packet Sockets, while the cellular connection is established using the Advanced Messaging Queuing Protocol (AMQP) protocol.

The main contribution of this dissertation focuses on the design, implementation and evaluation of a Hybrid Routing Sublayer, capable of isolating messages that are formed/decoded from transmission/reception processes. This layer is, therefore, capable of managing traffic coming/destined to the application layer of intelligent transport systems (ITS), adapting and passing ITS messages between the highest layers of the protocol stack and the available radio access technologies.

The Hybrid Routing Sublayer also reduces the financial costs due to the use of cellular communications and increases the efficiency of the use of the available electromagnetic spectrum, by introducing a cellular link controller using a Beacon Detector, which takes informed decisions related to the need to connect to a cellular network, according to different scenarios.

The experimental results prove that hybrid vehicular communications meet the requirements of cooperative intelligent transport systems, by taking advantage of the benefits of both communication technologies. When evaluated independently, the ITS-G5 technology has obvious advantages in terms of latency over the LTE technology, while the LTE technology performs better than ITS-G5, in terms of throughput and reliability.

Contents

List of Figures

List of Tables

Glossário

3G Third Generation

3GPP Third Generation Partnership Project

4G Fourth Generation

ADAS Advanced Driver Assistance Systems

AL Access Layer

AMQ-Model Advanced Message Queuing Model

AMQP Advanced Message Queuing Protocol

AP Access Point

API Application Programming Interface

APN Access Point Name

ARIB Association of Radio Industries and Businesses

ARP Address Resolution Protocol

ASN.1 Abstract Syntax Notation One

ASTM American Society for Testing and Materials

ATIS The Alliance For Telecommunications Industry Solutions

ATMPVC Asynchronous Transfer Mode Permanent Virtual Connection

BPSK Binary Phase Shift Keying

BSA Basic Set of Applications

BSD Berkeley Software Distribution

BSS Basic Service Set

BTP Basic Transfer Protocol

CAM Co-operative Awareness Message

CAN Controller Area Network

CCH Control Channel

CCSA China Communications Standards Association

CDC Communications Device Class

CE Conformité Européene

CEN European Committee for Standardization

CEPT European Conference of Postal and Telecommunications Administrations

C-ITS Cooperative Intelligent Transport Systems

CLI Command-Line Interface

CRDA Central Regulatory Domain Agent

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

D2D Device to Device

DAIR Driver Aided Information and Routing System

DARPA Defence Advanced Research Projects Agency

DCC Decentralized Congestion Control

DCCP Datagram Congestion Control Protocol

DCF Distributed Coordination Function

DENM Decentralized Environmental Notification Message

DFS Dynamic Frequency Selection

DLL Data Link Layer

DRAM Dynamic Random-Access Memory

DSRC Dedicated Short Range communications

DSS Device Service Stream

DTC Duplicated Transmission via Cellular interface

ECU Electronic Control Unit

EDCA Enhanced Distributed Channel Access

EEPROM Electrically Erasable Programmable Read-Only Memory

EFC Electronic Fee Collection

EIRP Effective Isotropic Radiated Power

eMBMS evolved MBMS

EMEA Europe, the Middle East and Africa

eNB evolved-NodeB

EPC Evolved Packet Core

EPS Evolved Packet System

ERTICO European Road Transport Telematics Implementation Co-ordination Organisation

ESD Electrostatic Discharge

ETC Electronic Toll Collection

ETSI European Telecommunications Standards Institute

E-UTRA Evolved Universal Terrestrial Radio Access

E-UTRAN Evolved Universal Terrestrial Radio Access Network

FCC Federal Communications Commission

FDD Frequency Division Duplex

FEC Forward Error Correction

FIFO First-In, First-Out

FL Facilities Layer

FOSS Free and open-source software

FOTA Firmware Over-The-Air

GNU GNU's Not Unix

GPIO General-Purpose Input/Output

GPS Global Positioning System

GSM Global System for Mobile communications

HDR High Data Rate

HMI Human Machine Interface

HSPDA High Speed Download Packet Access

HSPA High Speed Packet Access

HSUPA High Speed Upload Packet Access

HW Hardware

HWN Heterogeneous Wireless Networks

I2I Infrastructure-to-Infrastructure

I2V Infrastructure-to-Vehicle

IANA Internet Assigned Numbers Authority

ICMP Internet Control Message Protocol

ICT Information and Communication Technologies

IEC International Electrotechnical Commission

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IMT International Mobile Telecommunications

IoT Internet of Things

IP Internet Protocol

IPC Inter-Process Communication

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

IPX Internetwork Packet Exchange

ISO International Organization for Standardization

ITS-S ITS Station

ITS Intelligent Transport Systems

ITSC ITS Communications

ITU International Telecommunication Union

ITU-R International Telecommunication Union – Radiocommunication Sector

LDM Local Dynamic Map

LGA Land Grid Array

LLC Logical Link Control

LTE Long-Term Evolution

LTE-Advanced Long-Term Evolution-Advanced

M2M Machine to Machine

MAC Medium Access Control

MANETs Mobile Ad hoc Networks

MBIM Mobile Broadband Interface Model

MBMS Multimedia Broadcast and Multicast Services

MEC Mobile Edge Computing

MIB Management Information Base

MIMO Multiple Input Multiple Output

mPCIe mini-PCI-express

MQ Message Queue

MQTT Message Queuing Telemetry Transport

MRC Maximal Ratio Combining

MSM Mobile Software Management

MTU Maximum Transmission Unit

NGV Next Generation V2X

C-V2X Cellular-V2X

NIC Network Interface Controller

NTP Network Time Protocol

OBU On-board Unit

OCB Outside the Context of a BSS

OFDM Orthogonal Frequency Division Multiplexing

OFDMA Orthogonal Frequency Division Multiple Access

OS Operating System

OSI model Open Systems Interconnection model

PC Personal Computer

PCM Pulse-Code Modulation

PDA Personal Digital Assistant

PDR Packet Delivery Ratio

PHC Physical Clock

PHY Physical layer

PMTU Path Maximum Transmission Unit

POSIX Portable Operating System Interface

PS Packet switching

PTP Precision Time Protocol

QAM Quadrature Amplitude Modulation

QIPCRTR Qualcomm IPC router interface protocol

QoE Quality of Experience

QoE	Quality of Experience
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RAN	Radio Access Network
RAT	Radio Access Technology
REACH	Registration, Evaluation, Authorisation and Restriction of Chemicals
RF	Radio Frequency
RI	Regulatory information
RIT	Radio Interface Technology
RLAN	Radio Local Area Networks
RoHS	Restriction of Hazardous Substances Directive
RSSI	Received Signal Strength Indication
RSU	Roadside Unit
RTT	Round Trip Time
SAE	System Architecture Evolution
SATA	Serial ATA
SC-FDMA	Single-Carrier FDMA
SCH	Service Channel
SCTP	Stream Control Transmission Protocol
SIB	Security Information Base
SON	Self Optimizing Networks
SRC	Short Range Communications
SRITs	Set of Radio Interface Technologies
SSD	Solid State Drive
SSH	Secure Shell
SW	Software
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TPC	Transmission Power Control
TR	Technical Report
TSDSI	Telecommunications Standards Development Society
TTA	Telecommunications Technology Association
TTC	Telecommunication Technology Committee
TTT	Transport and Traffic Telematics
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
UE	User Equipment
UMTS	Universal Mobile Telecommunication System
USB	Universal Serial Bus
USDOT	United States Department of Transportation
USSD	Unstructured Supplementary Service Data
UTRAN	Universal Terrestrial Radio Access Network
V2I	Vehicle-to-Infrastructure
V2N	Vehicle-to-Network
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VANETs	Vehicular Ad hoc Networks
vhost	Virtual Host
VPN	Virtual Private Network
VRU	Vulnerable Road User
WAVE	Wireless Access for Vehicular Environments
WCDMA	Wideband Code Division Multiple Access
WHO	World Health Organization
WLAN	Wireless Local Area Networks
WMAN	Wireless Metropolitan Area Networks
WPA2	Wi-Fi Protected Access 2
WPAN	Wireless Personal Area Networks
WRC	World Radiocommunication Conference
WWAN	Wireless Wide Area Networks
V2P	Vehicle-to-Pedestrian
RSRP	Reference Signal Received Power

CHAPTER 1

Introduction

In the 20th century, transportation was about moving cars.
In the 21st century, the transportation landscape is rapidly evolving.

1.1 Background

The freedom to live, work, study and do business in another country is one of the EU's fundamental freedoms [1]. The EU population treasures this more than any other freedom achieved through EU integration, as a recent Eurobarometer report [2] showed over eight in ten Europeans (82% of those surveyed) supporting freedom of movement anywhere in the EU. However, this movement is complicated without good connectivity. Transport activity across Europe is high, it has been rapidly increasing and set to continue growing with more than 1.2 million companies working in transport services in the EU, employing around 11 million people in 2017 [3]. This is good news for economy, passengers and trade, but puts more pressure on the transportation network, especially the roads networks knowing that car passenger travel remains the dominant transport mode accounting for well over 70% of total passenger transport [4], emerging the need to change the nature of transport infrastructure with the increasing employment of smart technologies. For instance, instead of an ever expanding grid for transport, less roads and rail infrastructure will become necessary with higher precision transport systems [5].

Huge strides have been made in connecting Europe and improving traffic safety and operation, allowing people, goods and services to travel within and across borders. However, as the number of journeys has increased – in Europe and elsewhere – the threat of vehicles accidents causing more and more fatalities raised up too. Although modern vehicles are being equipped with different technologies to help reducing the number of accidents, albeit in 2017, 25300 people lost their lives on EU roads [6], with road traffic injuries being the first cause of death among children aged between five and fourteen years old and among young adults aged between fifteen and twenty-nine, the eighth leading cause of death globally in 2018 and estimated to be the tenth leading mortality cause even by 2060, according to the World Health

Organization (WHO) road safety report for 2018 [7] and the projections of mortality and causes of death, 2016 to 2060 [8], and human factors are involved in approximately 90% of all the cases [9], most of which are related to slow reaction by car drivers and motorcyclists [10], which makes road accidents amongst the most serious socioeconomic problems and consequently an area where huge effort by academia, industry and governments are concentrated, to achieve technological, ecological and social sector evolution in order to tackle this global challenge and reach a smarter and more affordable mobility, where people are fully aware of the surrounding risks, towards zero accidents, seamless delays and minimized impact on the environment, while providing secure transmitting of information and keeping the users privacy respected.

Research and development towards safer, more reliable and more efficient transportation has never stopped, since it started in the mid-1960s with General Motor's Driver Aided Information and Routing System (DAIR) [11] until today, passing through many attempts and successes in Europe, Asia, and America, holding different names, methods and technologies. But it was until autumn 1994 when the United States Department of Transportation (USDOT) officially sanctioned the term Intelligent Transport Systems (ITS) [11], giving more recognition to the larger usage of technology in transit systems as well as private and public vehicles and highways. And later that year -in December-, the term ITS was adopted by the first international event concerned in improving the transportation systems which was held in Paris, initiated and co-organized by The European Road Transport Telematics Implementation Co-ordination Organisation (ERTICO) under the name "1st ITS World Congress" and the theme "**Towards an Intelligent Transport System**" [12].

1.2 Purpose

Connected vehicles technology research indicates that vehicle-to-vehicle communication safety systems may address up to 80% of collision-based accidents where the driver is not impaired, making it a substantial requirement for any modern Intelligent Transport Systems [11] and giving an important role to every effort being put into enhancing vehicles connectivity, which is the main motivation behind this work to design a multi-radio communication system able to the benefit from independent technologies and consolidate their features to improve the connection availability, spectrum efficiency and make the best use of the accessible resources.

1.3 Objective

The main objective of this dissertation is to build on the foundation of PASMO [13] project's *IT2S* vehicular communication platform developed at Instituto de Telecomunicações-Aveiro, by

- Developing a multi-technology system for wireless vehicular communications, able to benefit from the platform's features and capable of providing communication services by utilising a combination of cellular (network-based) communication (e.g. LTE) and direct (localized) communication (e.g. ITS-G5);

- Designing and implementing a hybrid routing sublayer capable of controlling the state of the wireless interfaces by managing the data traffic flows that passes through each available technology;
- Deploying a selection algorithm that optimizes the resource allocation (spectrum usage) based on the available networks and connections stability.

This system could provide persistent connectivity under different conditions, leading to several benefits such as enhancing vehicles connectivity, improving connection availability, avoiding a single point of failure, improving the spectrum usage efficiency, and Reducing financial costs.

1.4 Structure

This dissertation continues with 6 more chapters organized as follows:

- **Chapter 2 - Fundamental Concepts -** This chapter provides a detailed explanation of the background concepts for the realization of this Master's dissertation. The basic topics of Cooperative Intelligent Transport Systems, wireless vehicular communications and hybrid vehicular communication are also reviewed. In addition, to the related radio access technologies (i.e. ITS-G5 and LTE) and implementation related topics, namely Sockets and the Advanced Message Queuing Protocol (AMQP);
- **Chapter 3 - State of the Art -** A review of the available literature related to the hybrid vehicular communications and protocol selection schemes.
- **Chapter 4 - System Architecture -** Proposal of the specifications and characteristics of the hybrid communication model to be implemented.
- **Chapter 5 - Implementation -** Description of the executed implementation;
- **Chapter 6 - Tests and Results -** Presentation of the test results and assessment of the adequacy of the implemented communication channels;
- **Chapter 7 - Conclusions and Future Work -** Final remarks and identification of future work directions.

CHAPTER 2

Fundamental Concepts

2.1 C-ITS

2.1.1 Overview

In 2008, the European Commission published an action plan for the deployment of ITS in Europe [14]. This plan foresaw the definition of a mandate for the European standards organisations (mainly European Telecommunications Standards Institute (ETSI) and CEN) to develop harmonised standards for ITS implementation, in particular regarding cooperative systems. As defined by the ETSI "Intelligent Transport Systems ITS are systems to support transportation of goods and humans with Information and Communication Technologies (ICT), in order to efficiently and safely use the transport infrastructure and transport means (cars, trains, planes, ships)" [15]. ICT systems based on sensors and equipment integrated into vehicles and infrastructure to collect information, process and present it to the road user, are already deployed and can be spotted in the marketplace such as traffic, road, weather information systems, electronic tolling systems and cars Advanced Driver Assistance Systems (ADAS) [16]. ITS in this meaning is the result of collaboration between ICT and transport engineering to plan, design, operate, maintain and manage transport systems; stating the multimodal nature of the activities in this area [11], even though this presented work will focus on the technological aspect, and more precisely, the use of technologies in vehicles guidance in road networks to boost transport efficiency, sustainability, safety and security.

Stand-alone built in vehicles or infrastructure technologies like the aforementioned ones can help drivers to get a safer driving experience, reduce the number of collisions, improve transportation flow and make better decisions in critical situations in different driving scenarios, thus have positive effects on safety, environment and traffic management. However, benefits could be further magnified if individual vehicles were given the ability to communicate with the other elements of the transportation system (other vehicles, pedestrians, road infrastructure)

or in other words, allow transport system entities to **cooperate** and share their conditions and their local environment information with each other, and the term C-ITS raises up to refer to this new subset of Intelligent transportation systems. Chain collisions avoidance serves as a good example of the advantages of cooperating on the roads, since it can be completely eliminated if the vehicles in the area of the first crash were able to disseminate information about the accident in their vicinity. Over recent years, the emphasis in ITS and intelligent vehicles has turned into C-ITS, therefore complying with European Telecommunications Standards Institute vocabulary, in the scope of this document, the term ITS refers to the European C-ITS as issued by the EU commission in the standardisation mandate M/453 [17] with all its services and applications. in addition, ETSI ITS-G5 will be used as the reference access technology [15], [18].

From a technical standpoint, the overall communication in ITS environment (ITSC) can be subdivided into:

- **ITS domain** refers to ITS elements that are defined in ITS/ITSC dedicated standards;
- **Generic domain** refers to all the other ITS legacy elements that are not defined by ITS/ITSC standards.

ITSC denotes transportation-scenarios dedicated communication systems, that aim to comprise the ITS elements and provide ITS services to the road users by deploying purpose-built devices able to use specific vehicular environment oriented access technologies within vehicles and infrastructure cites to enable direct regular information exchange: 1) among vehicles or mobile devices (Vehicle-to-Vehicle (V2V)) in ad-hoc mode without relying on any third party infrastructure; 2) between vehicles and road side infrastructure (Vehicle-to-Infrastructure (V2I) and Infrastructure-to-Vehicle (I2V)); 3) and among road side infrastructure installations Infrastructure-to-Infrastructure (I2I).
It is common nowadays to use the term Vehicle-to-Everything (V2X) network to point to any kind of wireless communication in the context of ITS, such as V2V and/or (V2I,I2V) as well as to the communication between vehicles and Vulnerable Road User (VRU) (e.g. pedestrians, cyclists) or Internet gateways, and differentiate them from I2I, which is mostly done based on wired connections between roadside ITS stations [19]–[21].

Based on the principles of the Open Systems Interconnection model (OSI model) defined in ISO/IEC 7498-1 [22], communications in ITS can be demonstrated in a stacked layers form that is able to handle all the OSI model functionalities, in addition to extended features to handle ITSC applications' requirements and it is known as the *ITS Station (ITS-S)* reference architecture defined by ETSI [15] and ISO [23], which is outlined in figure 2.1 and in the more detailed informative figure 2.2. An ITS-S by definition is a functional instance specified by the Intelligent Transport System Station reference architecture that is able to interact with other instances in the system network which means that the term ITS-S indicates a set of possible functionalities rather than a physical unit, while an *ITS application* is a union of multiple interdependent ITS-S applications (e.g. client/server connection) to compose a system that defines and implements a service which can be executed in a one or more *use*

case by the end user of the system. Deployable ITS applications are grouped and classified under the term *Basic Set of Applications (BSA)* [15], [19], [24]. The ITS-S reference protocol stack consists of six main functional blocks classified to three horizontal protocol layers that can be directly mapped to the layers of the OSI model -as in figure 2.3-, two vertical protocol entities and atop of all the ITS applications block [15], [18], [19], [24], [25]:

- **Access Layer (AL)** comprises L1 and L2 of the OSI model and includes functions to run internal and external communication, manage prioritization of transmission requests, the Logical Link Control (LLC) and the technology of accessing various types of communication media and their associated protocols, which can be either ITS-specific such as ETSI ITS-G5 or non-ITS-specific (Ethernet, Bluetooth, GPRS, UMTS, LTE) where in the latter case, these communication systems are considered as logical links that transparently transport ITS data.
- **Networking and Transport Layer (NL)** comprises L3 and L4 of the OSI model, defines the addressing and routing of data through the network from source to destination and/or geographical based data dissemination using one or more networking protocol, of which IPv4 IPv6, GeoNetworking, IP over GeoNetworking. As well as providing end-to-end reliable data delivery using one or more transport protocol such as TCP, UDP, or an ITS-specific transfer protocol like the ETSI defined Basic Transfer Protocol (BTP). Internet Protocol version 6 (IPv6) is of particular importance in the operations of this layer by reason of its utilization by ITS network protocols (GeoNetworking) in the dynamic ITS short range access technologies and the handover to be done between them in addition to the Internet Protocol version 4 (IPv4)/IPv6 interoperability.
- **Facilities Layer (FL)** comprises L5, L6 and L7 of the OSI model, thus it is intended to provide assistance to the ITS applications and their given use cases that might require shared functions and data. The facilities layer accomplishes its services through providing three essential categories of facilities: 1) Application support facilities is the kernel of the generic functions that assist ITS BSA Applications' functionalities such as the generic support for Human Machine Interface (HMI), time management, ITS applications software maintenance support, Co-operative Awareness Message (CAM) management support and Decentralized Environmental Notification Message (DENM) management support; 2) Information support facilities covers the OSI model presentation layer and many functions of the OSI model application. It contains the facilities to handle generic data and databases management for ITS BSA such as data structures for different types and sources of information (from vehicle local sensors or received by different communication channels), data presentation, data encryption, ASN.1 encoding/decoding, position management and Local Dynamic Map (LDM) database.transparent ad-hoc routing support; 3) Communication support facilities covers the functionalities of the OSI model session layer. It includes facilities to attain the ITS BSA needful communication modes (unicast, broadcast, multicast, geocast), in cooperation with the Networking and Transport layer such as supporting different modes of applications addressing, communication session initialization, maintenance and closure, transparent

mobility management, as well as transparent ad-hoc routing support.

- **Applications** acts as a residence for ITS applications where these applications are securely maintained, authorized, classified, prioritized and associated to logical channels and specific use cases.
- **Management Entity (ME)** resides the general Management Information Base (MIB) and responsible for functionalities related to ITS station management, ITS applications management and ITS communications management such as ITS service advertisement, general congestion control, cross-layer communication and Regulatory information (RI) management.
- **Security Entity (SE)** resides the general Security Information Base (SIB) and responsible for the security and privacy services such as firewalls, identities authentication, security credentials, crypto key and certificates management.

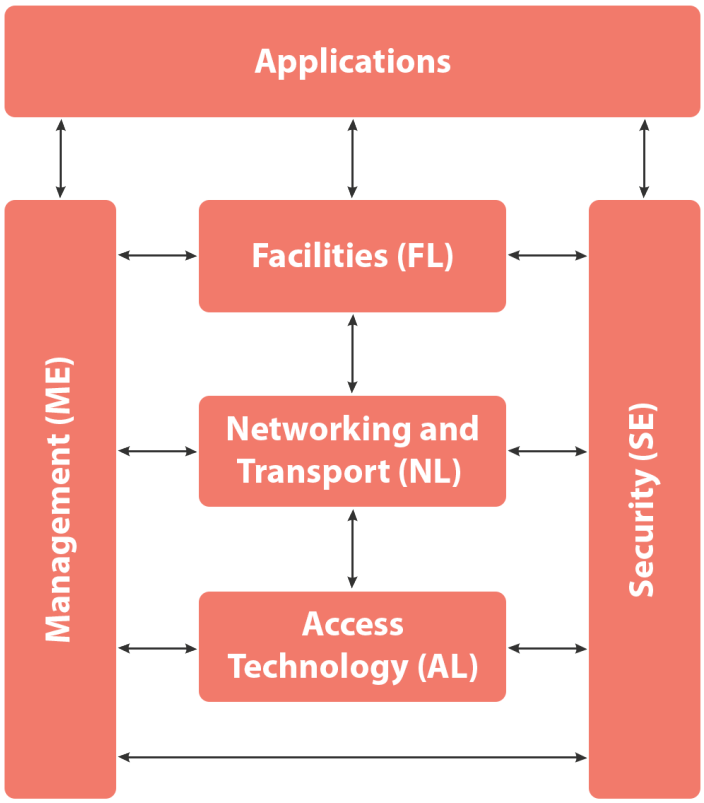


Figure 2.1: ITS station reference architecture/ITS-S host

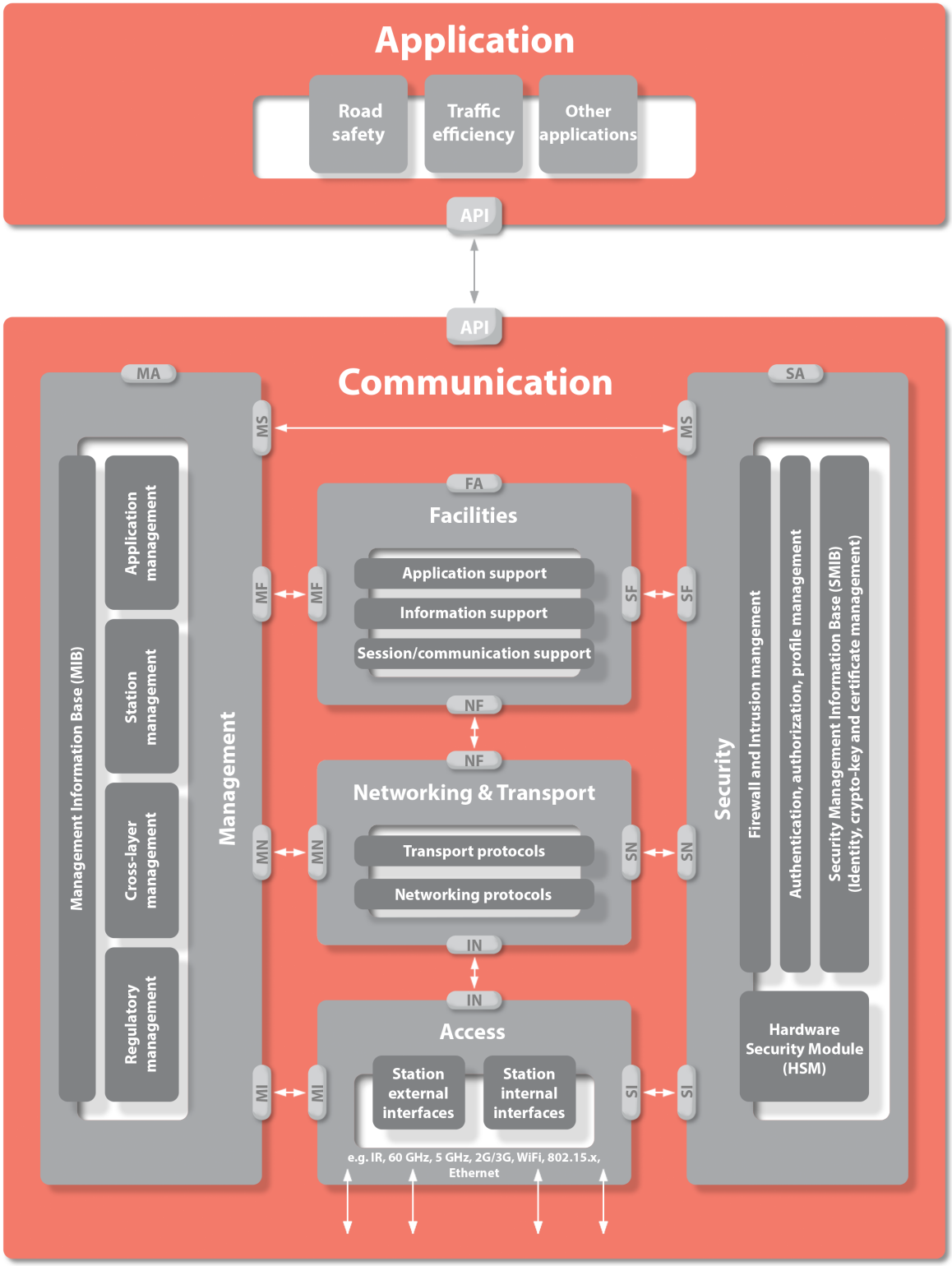


Figure 2.2: Examples of possible elements in the ITS station reference architecture

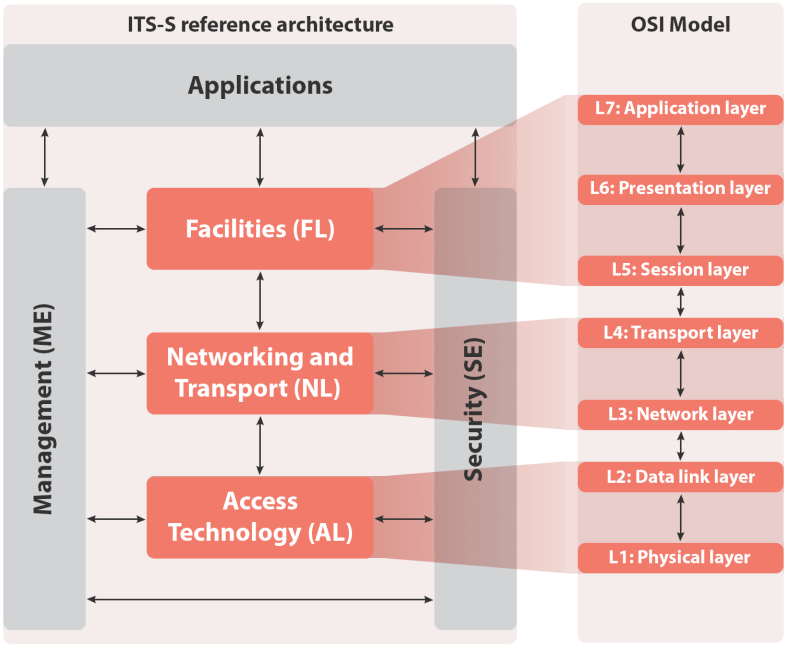


Figure 2.3: Mapping of OSI model and ITS-S reference architecture

2.1.2 ITSC functional elements

ITS reference architecture is the main building block of any ITSC component, thereby it is possible to specify the functionality of a particular component -and subsequently the functionality of the ITS-S- based on which layers and entities of the ITS reference architecture are deployed as summarized in table 2.1. ETSI defines four main ITSC *functional components* that might exist in any ITS-S [15]:

- **ITS Station host** Deploys the functionality of the entire ITS-S reference architecture with an essential purpose to be able to run ITS applications.
- **ITS Station gateway** interconnects two different OSI protocol stacks at layers 5 to 7 through the ITS facilities layer, therefore ITS-S gateways shall be able to convert protocols and their functionality excludes the applications block. Ideally, the network that interconnects the components of the ITS-S is termed as *ITS-S internal network* and it is connected to an OSI stack (non-ITS) through the ITS-S gateway.
- **ITS Station router** Interconnects two different ITS protocol stacks at layer 3 (ITS Networking and Transport layer), therefore ITS-S routers can have the ability to translate protocols and their functionality excludes the Facilities Layer (FL) as well as the applications block. Typically, ITS-S routers connect different functional components inside the ITS-S internal network, or the ITS-S internal network to an external ITS reference architecture stack.
- **ITS Station border router** provides the same functionality as the ITS-S router with a distinction that the border router is able to connect an ITSC stack to a non-ITS legacy networking protocol stack that does not support ITS management and security concepts,

therefore ITS-S border routers must be able to translate protocols[1].

ITS-S functional components	Layers & Entities					
	Applications	FL	NL	AL	ME	SE
ITS-S host	✓	✓	✓	✓	✓	✓
ITS-S router			✓	✓	✓	✓
ITS-S gateway		✓	✓	✓	✓	✓
ITS-S border router			✓	✓	✓	✓

Table 2.1: ITS layers and entities in ITS-S functional components

In an Intelligent Transport Systems, the functional components mentioned above make up ITS-Ss and depending on the components included in a certain station, an ITS-S can belong to one of the four following types (as briefly shown in table 2.2) [15], [24]:

- **Personal ITS station** provides ITS applications for personal and nomadic devices;
- **Central ITS station** provides centralized BSA of ITS applications. Thus, it might need to be connected to a backend system. Central ITS station may conduct as a road and traffic operator in addition to playing the role of a service and content provider.
- **Vehicle ITS station** provides ITS applications for drivers and passengers. It can be connected to the proprietary in-vehicle network and get access to the vehicle data through the (ITS-S gateway) functional component.
- **Roadside ITS station** provides ITS applications from the roadside to personal and vehicle ITS stations. A roadside ITS station can be connected to the proprietary roadside network through the (ITS-S gateway) to access the data collected by the road network equipment (sensors, radars, cameras). In addition, it can be connected to the central ITS station and other ITS roadside stations through the (ITS-S border router) to exchange information, hence it is able to provide ITS applications to the users either in an independent way or by cooperatively coordinate with the central and roadside stations.

ITS Station type	ITS-S functional components			
	Host	Gateway	Router	Border router
Personal	✓			
Central	✓	✓		✓
Vehicle	✓	✓	✓	
Roadside	✓	✓	✓	✓

Table 2.2: ITS-S functional components in ITS stations

[1] In the context of ITSC, the term *ITS-S interceptor* is either equal to an ITS-S gateway or an ITS-S router or an ITS-S border router

An ITS Station of any type is meant to be deployed and work in the context of a larger ITS entity that includes in addition to the ITS station, several other control, instrumentation and communication systems or even the central system in the case of a central ITS-S. This extensive entity is referred to using the term *ITS sub-system* which in turn operates in the context of the overall Intelligent Transport Systems. According to ETSI [15] there are four types of ITS sub-systems able to interconnect among each other by the means of peer-to-peer communications (figure 2.5) [15], [16]:

- **Personal ITS sub-system** provides ITS services in the context of a portable device (like a PDA or a mobile phone), therefore it should contain a personal ITS-S.
- **Central ITS sub-system** provides ITS applications functionality in the context of an ITS center, therefore it contains a central ITS-S. Usually it is run by the operators of the ITS applications to monitor and support them.
- **Vehicle ITS sub-system** is an ITS sub-system in the context of an ITS equipment used in a vehicle. It contains a vehicle ITS-S responsible for communications and hosting ITS applications, plus all the devices to collect information about the vehicle's situation and local environment (e.g. sensors, Electronic Control Units (ECUs)), jointly termed as On-board Unit (OBU).
- **Roadside ITS sub-system** ITS sub-system in the context of roadside ITS equipment installed along the roads therefore it is known as Roadside Unit (RSU). It contains a roadside ITS-S responsible for communications and hosting ITS applications, plus all the devices, sensors, cameras to collect information about the road state (weather, congestion) and control traffic flow (traffic light, signs and signals).

2.1.3 Networks

ITS Station is the key component of ITS and allowing them to be effectively active in different varieties of communication scenarios is the main aim of ITSC networks. From a synoptic point of view, ITS network architecture can be assorted into *ITS-S internal network* and *external networks*. ITS internal network provides connectivity between ITS-S components (e.g. ITS host to ITS border router), while the external networks interconnect ITS-Ss (e.g. vehicle ITS-S to roadside ITS-S), or connect ITS-Ss with other network entities that might be outside the context of transportation system (e.g. an Internet server) [18]. Figure 2.4 abstractly depicts the ITS network architecture including the ITS station boundary and its connectivity to the proprietary network and the external networks.

External networks are categorized into ITS domain networks and generic domain networks (figure 2.6) as specified in (ETSI EN 302 665) [15] and can be further classified and defined as follows [15], [18]:

- **ITS ad-hoc network** is an ITS domain network that enables short range wireless ad-hoc communication among vehicle, roadside and personal ITS stations through dedicated wireless technologies (e.g. ITS-G5) that support high mobility, arbitrarily formed network topologies without the need for a coordinating communication infrastructure.

- **ITS access network** is an ITS domain dedicated network that provides 1) access to specific ITS services and applications owned or run by the road operators for instance; 2) roadside ITS stations interconnection and the connection to the central ITS station (e.g road traffic management center), therefore it is able to provides indirect communication among ITS vehicle stations through the connected ITS roadside stations;
- **Public access network** is a generic domain network that provides access to publicly accessible general purpose networks such as the LTE access network being used to connect vehicle ITS stations to the internet.
- **Private access network** is a generic domain network that provides secured access for a closed group of users (e.g. vehicle ITS-S) to a private network such as a company's intranet.
- **Core Network** provides legacy services such as WWW and email service to the ITS stations via the private and public access networks.

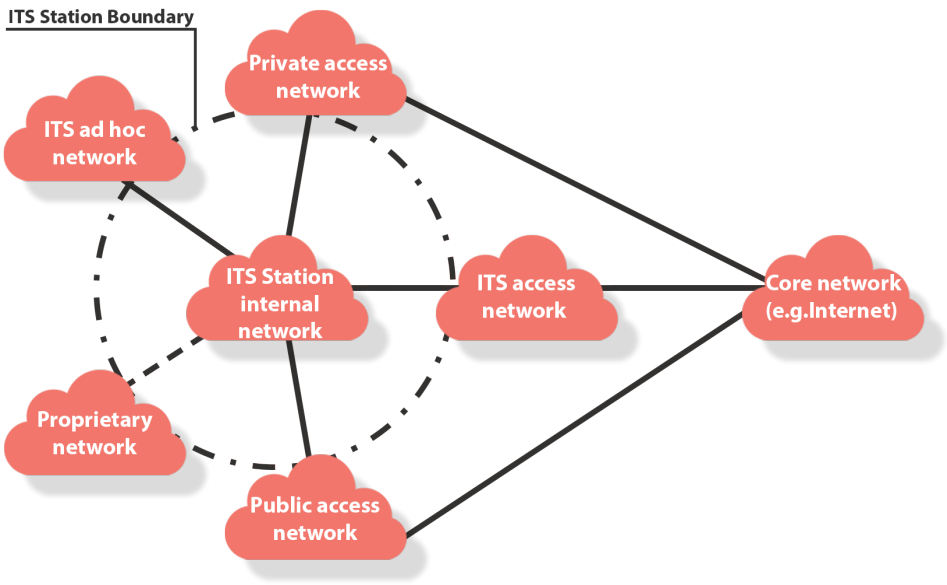


Figure 2.4: ITS station boundary in the context of the overall networking view

The different networks shall provide support for at least one of the use cases of road safety, traffic efficiency, infotainment and business applications. However, it is presumed that the communication within a single network does not meet all the requirements of all applications and use cases. Instead the external networks are interconnected as abstracted in figure 2.6 and combinations of networks are formed, in which multiple ITS access and networking technologies are applied. In addition to the networks listed above, an ITS station can also be attached to proprietary local networks of an ITS sub-system such as a Controller Area Network (CAN) in a vehicle ITS sub-system, and a Legacy roadside infrastructure in a roadside ITS sub-system [15], [16], [18].

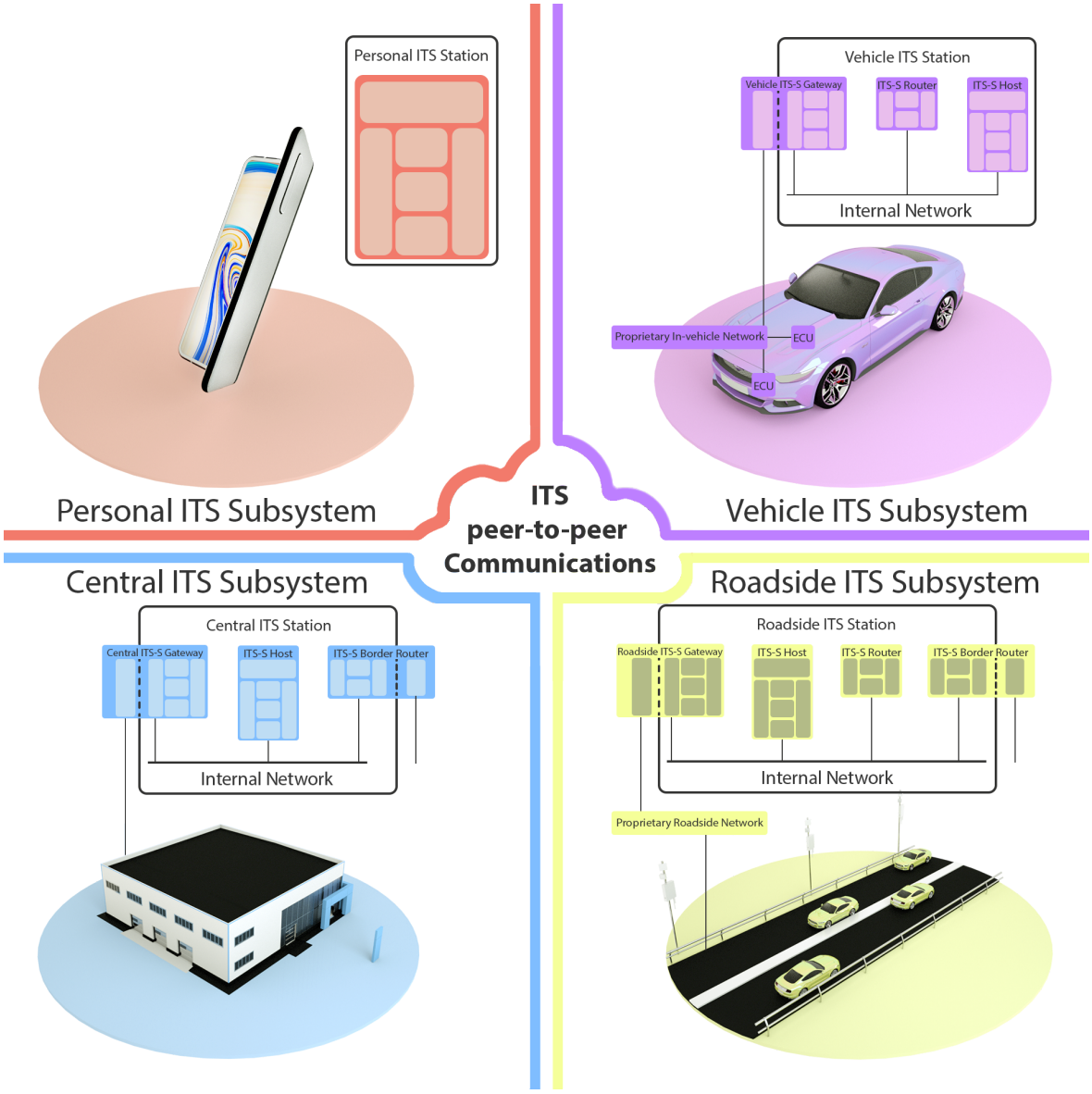


Figure 2.5: European ITS communication sub-systems

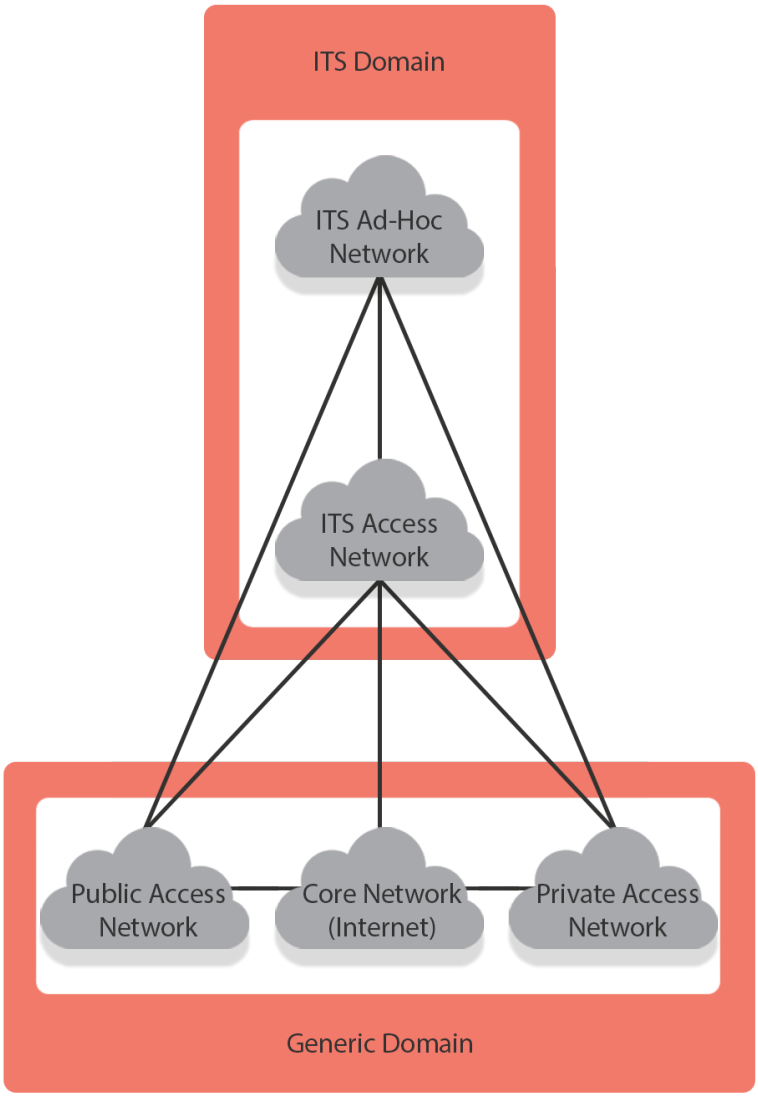


Figure 2.6: External networks involved in the ITS architecture and their interconnections

2.1.4 Applications

Key concepts in ITS are the ability of transport entities (vehicles, roadside infrastructure, pedestrians, etc.) to be connected either among each other in the form of exchanging knowledge of their local environment, collected from a range of sensor equipment, and provide these entities access to a set of different types of infotainment and entertainment services (via the internet for example) in order to improve transportation safety, make more intelligent use of the transport infrastructure, therefore enhance drivers' and passengers' welfare and comfort on the roads. The above-mentioned objectives and the existence of enabling technologies spur the development of large diversity of ITS applications with each application implementing one or more use cases. According to ETSI, the entire set of possible deployable applications and use cases that can be provided to several customers' profiles (Vehicle owner, vehicle driver, passenger or even the road traffic managers) in different transportation contexts (termed BSA) consists of three main ITS applications classes, namely, *road safety* to improve vehicles' occupants safety, *traffic efficiency* to improve the road traffic management, and other *infotainment and business* applications for Co-operative local services and Global internet services use cases.

- **Road/Traffic Safety Applications** are designed with the goal to bring down the number of road accidents, or at least minimize the damage of the inescapable ones by enhancing drivers' awareness of their direct and indirect environment factors (e.g. road accidents, pedestrians, road construction works, vehicles moving in the wrong direction and more) using V2V and/or V2I communication modes depending on the emergency level of the situation (V2V mode is mainly used for safety-critical applications while V2I is mainly used in safety-related applications) [19], [26]. However, traffic safety application in general, due to their critical use cases are the most resources demanding class of ITS applications. They require dedicated high quality hardware able to provide reliable and time-critical communication [16].
- **Traffic Efficiency Applications** have a primary objective of improving the traffic fluidity, but they can also have economical and environmental secondary benefits that are not directly related to traffic efficiency. Traffic efficiency application may be initiated directly from a roadside ITS station, or from a central ITS station to provides traffic information to road users through an authorized roadside ITS station to vehicle ITS stations or personal ITS stations which creates two possible communication scenarios:
  - Information dissemination from the roadside ITS station to personal and vehicle ITS stations can be done by broadcasting the messages within a specific area (Geobroadcast) or sent to a specific vehicle or personal ITS station (point to point);
  - Communication between ITS roadside station and the central ITS station is only based on point to point communication.

  Traffic efficiency applications running on a roadside station may make use of the road safety CAMs and DENMs they receive from vehicles and other roadside stations to determine their reaction to a certain event. In such case, two possibilities exist:

- CAMs and DENMs are directly forwarded to the central ITS station by point to point communication without any modification or processing.
- CAMs and DENMs are locally pre-processed in the roadside station and then the aggregated information are forwarded to the central ITS station by point to point communication.

It is important to mention that although traffic management applications do not present strict reliability and latency requirements, their performance is highly affected by delay and packet loss [16].

- **Infotainment and business Applications** provide access to comfort, convenience and entertainment services, generally similar to smart-phones applications but with greater Quality of Service (QoS) and Quality of Experience (QoE) requirements, due to the dynamic network topology caused by high nodes mobility in vehicular environments. Because of the diverse nature of these applications, their requirements can highly vary from one application to the other, but they generally share the long delay tolerance (up to a certain level) and high data throughput [16], [26]. This applications class can be further divided into two sub-classes:
  - Co-operative local services that offers location based services such as point of interest notification, automatic access control and parking management and media downloading;
  - Global internet services that offer communities services and ITS station life cycle management applications such as insurance and financial services, vehicle software/data provisioning and update, video on demand, messaging and calling services.

2.2 Wireless Vehicular Communications

Connecting means of transport has the potential to revolutionise the way we travel by expanding the area of individual vehicles' awareness beyond the field of their local sensors to include information collected and shared by other vehicles, as well as all the road users and the infrastructure. However, the long inter-vehicles distances, their fast movement in predetermined directions, and rapidly varying density makes it impossible to effectively use the traditional Wireless Local Area Networks (WLAN) techniques and even challenging for the classical Mobile Ad hoc Networks (MANETs) without affecting the reliable and the low-latency dissemination of information [27]. Therefore, coping with the new conditions since law-latency and reliability cannot be relinquished in some time-critical safety related situations, a special version of MANETs known as Vehicular Ad hoc Networks (VANETs) was designed by integrating MANETs with broadband mobile technology principles to overcome those challenges and support the vehicular environment required applications and use cases. Although VANETs share the main features of the ad hoc networks such as the short range, the narrow bandwidth, and the ability to organise and manage the network by the nodes themselves (the vehicles in our case), IEEE Vehicular Technology Magazine [28] lists the following characteristics that distinguish VANETs from other Mobile Ad hoc Networks:

- **Highly dynamic topology.** Due to high speed of movement between vehicles, the topology of VANETs is always changing. For example, assume that the wireless transmission range of each vehicle is 250 m, so that there is a link between two cars if the distance between them is less than 250 m. In the worst case, if two cars with the speed of 60 mph (25 m/sec) are driving in opposite directions, the link will last only for at most 10 sec.
- **Frequently disconnected network.** Due to the same reason, the connectivity of the VANETs could also be changed frequently. Especially when the vehicle density is low, it has higher probability that the network is disconnected. In some applications, such as ubiquitous Internet access, the problem needs to be solved. However, one possible solution is to pre-deploy several relay nodes or access points along the road to keep the connectivity.
- **Sufficient energy and storage.** A common characteristic of nodes in VANETs is that nodes have ample energy and computing power (including both storage and processing), since nodes are cars instead of small handheld devices.
- **Geographical type of communication.** Compared to other networks that use unicast or multicast where the communication end points are defined by ID or group ID, the VANETs often have a new type of communication which addresses geographical areas where packets need to be forwarded (e.g., in safety driving applications).
- **Mobility modelling and predication.** Due to highly mobile node movement and dynamic topology, mobility model and predication play an important role in network protocol design for VANETs. Moreover, vehicular nodes are usually constrained by pre-built highways, roads and streets, so given the speed and the street map, the future position of the vehicle can be predicated.
- **Various communications environments.** VANETs are usually operated in two typical communications environments. In highway traffic scenarios, the environment is relatively simple and straightforward (e.g., constrained one-dimensional movement), while in city conditions it becomes much more complex. The streets in a city are often separated by buildings, trees and other obstacles. Therefore, there is not always a direct line of communications in the direction of intended data communication.
- **Hard delay constraints.** In some VANETs applications, the network does not require high data rates but has hard delay constraints. For example, in an automatic highway system, when brake event happens, the message should be transferred and arrived in a certain time to avoid car crash. In this kind of applications, instead of average delay, the maximum delay will be crucial.
- **Interaction with on-board sensors.** It is assumed that the nodes are equipped with on-board sensors to provide information which can be used to form communication links and for routing purposes. For example, GPS receivers are increasingly becoming common in cars which help to provide location information for routing purposes. It is assumed that the nodes are equipped with on-board sensors to provide information which can be used to form communication links and for routing purposes.

Figure 2.7: ITS Basic Set of Applications

The physical radio frequency allocation for vehicular communications in the United States [29] and Europe [30], [31] is shown in figure 2.9, and it can be seen that both have allocated spectrum in 5.9 GHz range for Intelligent Transportation System uses.

The Federal Communications Commission (FCC) in the United States has divided the 75 MHz available spectrum in the 5.8505.925 GHz band into seven equal channels of 10 MHz and one 5 MHz guard band at the low end of each channel, with the seven main channels being split into one *Control Channel (CCH)* (Institute of Electrical and Electronics Engineers (IEEE) channel number 178) for communication management and to broadcast road safety-related messages; and six *Service Channels (SCHs)* for traffic efficiency and infotainment applications [16].

While in Europe, the European Conference of Postal and Telecommunications Administrations (CEPT) has divided the allocated spectrum into four sub-bands containing one *Control Channel* and seven fixed *SCHs* including a 5 MHz guard band at the low end of each channel [16], [32], [33]:

- **ITS-G5A frequency band** in the frequency range [5.875 to 5.905] GHz; contains channels SCH1, SCH2 and the control channel CCH (IEEE channel number 180); and is set aside for ITS road traffic safety applications.
- **ITS-G5B frequency band** in the frequency range [5.855 to 5.875] ;contains channels SCH3 and SCH4; and is set aside for ITS non-safety road traffic applications.

- **ITS-G5C frequency band** in the frequency range [5.470 to 5.725] ;contains the SCH7 channel; and is used to provide public Radio Local Area Networks (RLAN) access.
- **ITS-G5D frequency band** in the frequency range [5.905 to 5.925] GHz; contains channels SCH5 and SCH6; and is set aside for future usage of ITS road traffic applications.

2.3 Hybrid Vehicular Communication

Catered for the huge demand on high quality connectivity anytime and anywhere, wireless technology has seen rapid progress in the recent years and under the WLAN, Wireless Wide Area Networks (WWAN), Wireless Metropolitan Area Networks (WMAN) and Wireless Personal Area Networks (WPAN) families, numerous standards are deployed and in most cases, with interfered coverage areas, simultaneously giving the ability to the users to establish hybrid connections to different networks in the same location at the same time, forming Heterogeneous Wireless Networks (HWN). In addition, when a terminal has the capability of accessing the Internet for example by connecting to two or more different networks through different types of Radio Access Technologies (RATs), it has the privilege to select the best available access networks for its specific Quality of Service (QoS) needs [34]. Wireless communication in vehicular environments is not an exception, there are numerous of wireless communications technologies that meet ITS requirements, including ITS-G5, CEN DSRC, Bluetooth, mmWave and mobile cellular systems (3G/4G, LTE, 5G). Consequently, equipping vehicles with multiple RATs elevates their ability to be continually connected, but it also comes with other challenges that need to be addressed, like the interoperability, the backward compatibility, and other multiple access layer network architecture related issues. The term "hybrid communication" in vehicular environments is mainly used to identify combining **localized** direct communication (e.g. ITS-G5, IEEE WAVE) between vehicles, infrastructure and any other transportation system entity called V2X; and **wide area** communications networks (e.g. cellular networks, Internet) Vehicle-to-Network (V2N)[2] where signals are communicated through cellular telecommunication networks. Direct V2X communication is suitable for collision avoidance and automated driving purposes, while network based V2N is good for broadcasting information to broad areas and large amount of data. It is obvious that no single wireless communication technology could fulfill all the specifics of communication technology and objective of ITS. As a matter of fact, it is possible for these two communication systems to complement each other sensibly in certain scenarios in order to fill corresponding technological gaps, and an appropriate mix of these two different technologies is what hybrid vehicular communication all about.

Hybrid communication got real momentum in Europe when it was introduced by the European Commission in their C-ITS deployment platform Phase-1 report [35], that mentioned the functional and technical communication requirements variations of different Connected, cooperative and automated ITS services and applications, affirming that a hybrid communication approach including multiple technologies and radios is the only way to support

[2]V2N including telematics is regarded as part of V2X

continued deployment of ITS services and applications today and in the future. The report also concluded to the importance of access-layer agnostic, that is the ability of transmitting C-ITS messages independently of the from the underlying communication technology.

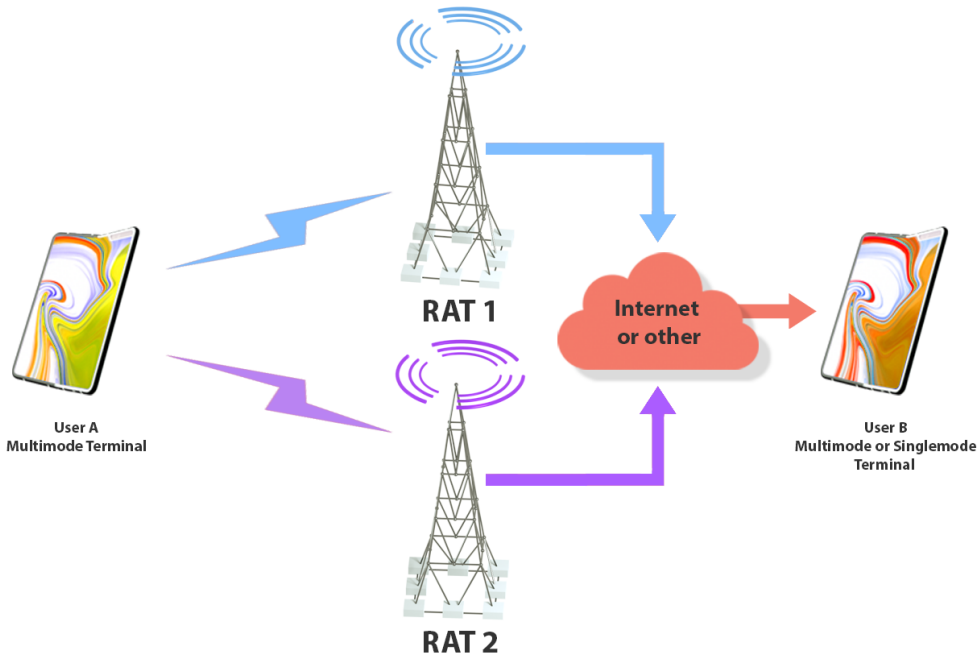


Figure 2.8: Hybrid-link connection

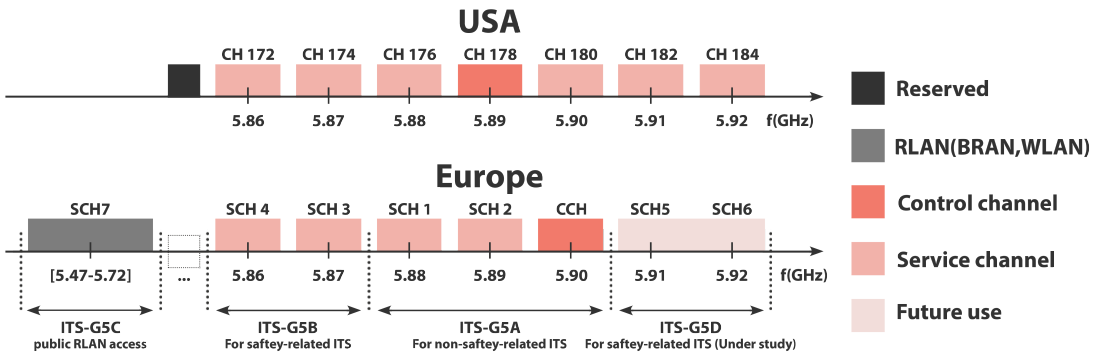


Figure 2.9: Spectrum allocation for vehicular communication in EU and the US

2.4.1 Short Range Communications - ITS-G5

Since the birth of communications in the 5 GHz bands for Intelligent Transport Systems (ITS) in the US [36], the term Dedicated Short Range communications (DSRC) was devoted to the definition of communication systems that are designed to transfer information between vehicles and roadside systems over a short range, wireless link. Technical standardization of Dedicated Short Range communications (DSRC) was firstly done by the American Society for Testing and Materials (ASTM) through [37] and its following [38] which were meant to be an extension of IEEE 802.11 technology into the high-speed vehicle environment, describing the MAC and PHY specifications for wireless connectivity using DSRC services. Upper layers of the communication stack in the US are specified by the IEEE Std 1609 set of standards [39]–[43], under the name Wireless Access for Vehicular Environments (WAVE). Over the time, different radio standards adopted the same concept, for instance, ETSI ITS-G5 access layer technology [44] in Europe that refers to radio spectrum and protocols associated with localized vehicular communications functioning at the 5.9 GHz band within an ITS that has the communications architecture defined by [15]; and the International Organization for Standardization (ISO) ITS-M5 [45], which introduces the specifications of a communication interface compatible with the ITS station and communication architecture specified in ISO 21217 [46]. Another example in Europe would be the major Electronic Toll Collection (ETC) and Electronic Fee Collection (EFC) standards (at 5.8 GHz), namely *CEN-DSRC* technology (also named TTT-DSRC) standardized by European Committee for Standardization (CEN) [47]–[49], and later included in ETSI European harmonised multi part standard [50], [51]; and *HDR-DSRC* standardized by UNINFO in Italy and by ETSI ES 200 674-1 [52].

Accordingly, and since there is no consensual global agreement on the definition of DSRC [53] which was initially used to describe the access layer technology in of the IEEE WAVE communication stack, in addition to the confusion that occur with the European tags for toll road systems (CEN-DSRC), in the scope of this document, terms like “localized communications”, “short-range localized communications” or mostly “Short Range Communications (SRC)” are used to refer to wireless Vehicle-to-Everything (V2X) technologies standards specifically designed for transportation applications, that provide direct short-range data exchange between Intelligent Transport Systems elements (e.g. vehicles, infrastructure, pedestrians) without the need for association nor for authentication at the PHY/MAC layers. Localized vehicular communications operate in the 5.9GHz reserved spectrum band and have an approximate maximum range of 1000 m [16].

All available vehicular communication Short Range Communications (SRC) protocols build their communication stacks on top of the unique IEEE 802.11p-2010 [54] that was in turn based on the extensive testing and analyses of wireless communications in a mobile environment documented in ASTM E2213-03 [38] Historically, IEEE Std 802.11 family of standards as originally published in 1997 [55], and through several revisions re-affirmations until IEEE 802.11-2007 was designed to specify one common MAC layer and several PHYs to

provide wireless connectivity among fixed, portable, and moving access points and stations within a local area, by allowing stations to form and join communication clusters, known in the IEEE 802.11 parlance as Basic Service Set (BSS) through -relatively- time consuming synchronization, authentication and association processes. The Basic Service Set is the basic building block of an IEEE 802.11 LAN, and stations are allowed to dynamically form, join and even move from one BSS to another by repeating the same BSS synchronization procedure every time, which introduces significant delays while connecting or re-connecting to the network, which in the best conditions takes 600 ms [56] but usually takes several seconds [57], and even longer delays can be observed in dense regions or heavy traffic conditions [58]. Hence, all the included standards -at that time- were suitable for users that are stationary or moving with low speed like while walking, but not very convenient for high mobility environments the where physical layer properties are rapidly changing and very short-duration communications exchanges are required, especially in situations the stations are rapidly moving and transactions must be completed in time frames much shorter than the minimum possible amount of time required to perform standard authentication and association to join a BSS [54]. In 2010, IEEE presented the IEEE 802.11p-2010 [54] -as an amendment to IEEE 802.11-2007 [59]- which specifies the extensions to IEEE 802.11 for WLANs (WLANs) to support new emerging applications utilizing wireless communication between vehicles, in an effort to decrease road traffic accidents and improve road traffic efficiency. It describes the functions and services required by stations to operate in a rapidly varying environment, and to exchange messages without joining a BSS. It also defines the signaling techniques and interface functions used by stations communicating Outside the Context of a BSS (OCB) [54].

The PHY layer of the IEEE 802.11p uses Orthogonal Frequency Division Multiplexing (OFDM) with BPSK, QPSK, 16QAM and 64QAM, similarly to the IEEE 802.11a [60], but with some modifications to the Orthogonal Frequency Division Multiplexing (OFDM) specifications to achieve robust communication under high velocities, that is, enabling the half-clocked mode through doubling OFDM timing parameters, hence [16], [54], [60]1:

- reducing the bandwidth to 10 MHz (from 20 MHz in IEEE 802.11a) which reduces the effects of Doppler shifts, and at the same time halves the data rates to be in the range [3, 27] Mbits instead of [6, 54] Mbits at bandwidth 20 MHz.
- allowing for larger guard bands which reduces the inter-symbol interference caused by multi-path propagation.

In order to provide sufficient communication ranges -up to 1000 m [16]- for vehicular scenarios, significantly high Effective Isotropic Radiated Power (EIRP) levels are allowed by IEEE 802.11p, compared to typical WLAN defined in the IEEE 802.11 standards. For instance, the European maximum EIRP for ITS communications is 33 dBm (2 W)[3], and in the US, 44.8 dBm (30 W) is allowable for emergency vehicles and 33 dBm for safety relevant messages[4] [44], [54], [59].

[3]Maximum EIRP in Europe for IEEE 802.11a is 500 mW [59]
[4]Maximum EIRP in the US for IEEE 802.11a is 800 mW [59]

The MAC medium access algorithm deployed by 802.11p to schedule transmissions is called Enhanced Distributed Channel Access (EDCA) as defined in the IEEE 802.11-2016 [60]. EDCA was introduced in 2005 to the IEEE 802.11 with the IEEE 802.11e-2005 amendment [61] to add QoS features to the Distributed Coordination Function (DCF) algorithm, which basically implements the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) access method, i.e. the EDCA can be described to be a CSMA/CA algorithm with the possibility to prioritize data traffic and change the algorithm execution through modifying a set of parameters, and these parameters in IEEE 802.11p are set to values specifically chosen for vehicular communication scenarios [44], [54], [60].

Some other tweaks and extensions were standardized in IEEE 802.11p to allow low overhead operations and guarantee fast and reliable exchange of safety messages. However, the major change to the overall 802.11 standard was giving stations the possibility not only to communicate within a BSS, but also to exchange data frames without any prior network establishment, i.e. no authentication and association procedures are allowed at the MAC layer in 802.11p, and such data frames are defined as being transmitted OCB. When OCB mode is enabled, a data frame can be sent to either an individual or a group destination MAC address, allowing immediate communication, avoiding the latency associated with IEEE 802.11 authentication, association, or data confidentiality services that need to be done when establishing or joining a BSS, and leaving them to be handled by upper layers. This capability is particularly well-suited for use in rapidly varying communication environments such as those involving vehicles where the interval over which the communication exchanges take place may be of very short-duration, since tuning to the right dedicated frequency band is all it takes to be able to transmit or receive frames in OCB mode [16], [44], [54], [60].

The comparison of 802.11a with 802.11p done in [62] shows that the latter achieves better performance in vehicular environments measured as packet delivery ratio, throughput and end-to-end delay. Authors of [63] presented a comparison of using regular WiFi (802.11a) and DSRC/WAVE (802.11p) technologies for non-safety transmission using TCP protocol with single-path and multi-path mode. Results show that handover time in WiFi networks is longer than in DSRC.

In March 2012, the amendment IEEE 802.11p-2010 [54] was incorporated into the IEEE 802.11-2012 [64] (the base standard at that time), and was classified as superseded but the name 802.11p is still used to refer when the OCB mode is activated enabling the communication outside the context of a BSS. IEEE 802.11-2012 was subsequently updated in 2016 into IEEE 802.11-2016 [60] that included in addition to the previously published standards, all amendments between 2012-2016. It is also important to mention that IEEE is working on a new vehicular communications standard for new V2X application since March 2018. The new standard that is expected to be published in December 2021 under the name IEEE 802.11bd Next Generation V2X (NGV), is aiming for higher throughput, better reliability, and extended range while keeping backward compatibility with 802.11p [65].

The dominant short-range localized communication access technologies for vehicular communication systems, are the IEEE Wireless Access for Vehicular Environments (WAVE)

Characteristics	IEEE 802.11p	IEEE 802.11a
EU band	5.850-5925 GHz	5.15–5.725 GHz
Channel spacing	10 MHz	20 MHz
Data rate [Mbps]	3, 4, 5, 6, 9, 12, 18, 24, 27	6, 9, 12, 18, 24, 36, 48, 54
Subcarriers	52	52
Modulation	BPSK, QPSK, 16-QAM, 64-QAM	BPSK, QPSK, 16-QAM, 64-QAM
Signaling	OFDM	OFDM
FEC rate	12, 23, 34	12, 23, 34
Subcarrier spacing	0.15625 MHz	0.3125 MHz
Guard time	1.6 µs	0.8 µs
Symbol interval	8 µs	4 µs
Preamble interval	32 µs	16 µs

Table 2.3: IEEE 802.11a and IEEE 802.11p

implemented in the US, and the European ETSI ITS-G5, and both of them use the IEEE 802.11-2016 [60] Wi-Fi standard to implement their physical (PHY) and Medium Access Control (MAC) layers [44], [66]. The present work follows the European protocol standards for supporting vehicular communications, according to the communication stack specified by the ITS-station reference architecture (figure 2.1) which uses ETSI ITS-G5 as an access technology. ITS-G5 is the technology specified for the Access Layer (AL) of the ITS station reference architecture (figure 2.1), which uses already existing standards for communications. The Access Layer consists of the two lowest layers of the protocol stack, namely the physical layer and Data Link Layer (DLL) as shown in figure 2.10. The physical layer is fully compliant with the IEEE 802.11-2016 [60]. The DLL is divided into two sublayers; the Medium Access Control (MAC) which follows IEEE 802.11-2016 to provide transmission scheduling mechanisms to minimize interference; and the Logical Link Control (LLC) (that is based on the IEEE/ISO/IEC 8802-2-1998 [67]) to add the ability of distinguishing between different network layer protocols. ITS-G5 also implements a cross layer vertical entity for Decentralized Congestion Control (DCC) (ETSI TS 102 687 [68]) to control the network load and to ensure that a single ITS station does not consume all resources on the channel as can be seen in figure 2.11.

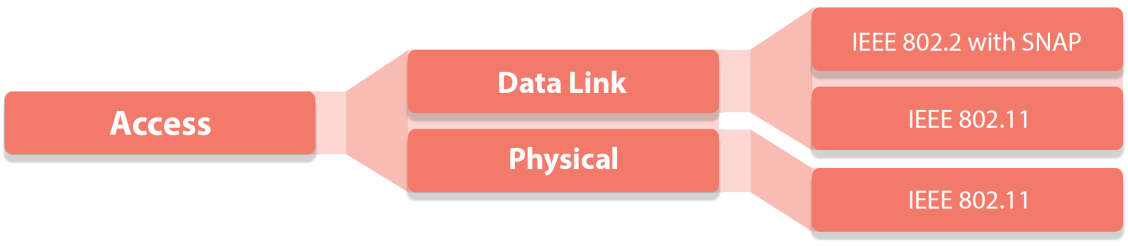


Figure 2.10: Protocols comprising the access layer

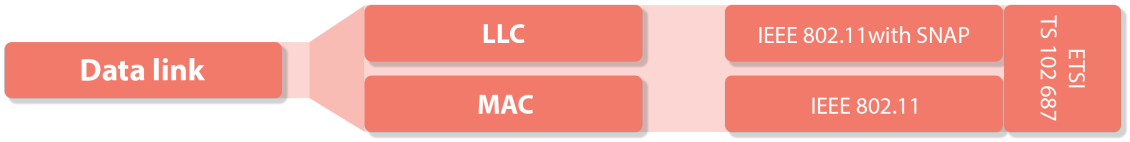


Figure 2.11: Protocols and DCC management entity comprising the data link layer

2.4.2 LTE

LTE is the Fourth Generation (4G) of mobile communication systems, initially designed by a union of telecommunications regional standard development organizations, mobile devices manufacturers and network operators already running UMTS, all known as Third Generation Partnership Project (3GPP)[5] with an initial purpose for LTE to be an enhancement of the existing Third Generation (3G) Universal Mobile Telecommunication System (UMTS)[6], and to *provide the starting point for a smooth transition to 4G radio access*, before it became a completely different technology yet still compatible with the older releases UMTS and Global System for Mobile communications (GSM) [69]–[71].

The earliest recognised initiative to overpass some of the inherent limitations of UMTS after the High Speed Packet Access (HSPA) was taken in November 2004 at the Radio Access Network (RAN) workshop, with more than forty contributions by different scientific, commercial and industrial bodies sharing their visions and plenty of ideas and suggestions to set up the evolution framework of 3GPP Radio Access Technology and the future required features that should be added to the 3G UMTS to reach a simpler, faster, more flexible and more reliable multi-service telecommunications system, with a higher capacity and lower cost per bit, capable of handling the dramatic escalation of mobile data usage driven by 3G success, and at the same time providing a high quality voice calls service, towards approaching the fixed access networks user experience added to all the feature offered by mobile devices [70]–[73].

One month later, in December 2004, with respect to the recommendations of the RAN workshop, 3GPP commenced a study item aiming to ensure UMTS competitiveness and keep it up to the market needs for the next ten years and beyond [72], [74] with an overall goal of creating a network of a flat architecture and an IP-based radio-access technology that only relies on Packet switching (PS) for all types of network services. The results of the first phase of the study were published 6 months later, as Technical Reports (TR) under Release-7 of

[5]3GPP Organizational Partners: Association of Radio Industries and Businesses (ARIB), The Alliance For Telecommunications Industry Solutions (ATIS), China Communications Standards Association (CCSA), ETSI, Telecommunications Standards Development Society (TSDSI), Telecommunications Technology Association (TTA), and Telecommunication Technology Committee (TTC)

[6]UMTS was also developed by 3GPP

the 3GPP specification series (3GPP TR 25.913)[7] [75], [76] and (3GPP TR 25.912)[8] [74] with the former addressing the requirements for the Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN), and the latter describing the air interface design process [69], [74], [75].

3GPP Technical Report (TR 25.913) [74] suggested moving ahead with the study while focusing on the PS domain services in different network levels, and for these studies to be related but not limited to the radio-interface physical layer, radio-interface layer 2 and 3, Universal Terrestrial Radio Access Network (UTRAN) architecture and Radio Frequency (RF) issues. The report also considered among 3GPP goals; supporting high speed mobility; wide coverage areas; low handover interruption times, and it also referred to the importance of the new adopted advanced multi-antenna technologies; the flexible carrier bandwidth and the tested User Equipment category capabilities in the system performance. Hence it provided the test conditions under which the guidance for the requirements to be met by the E-UTRA and the UTRAN system, mostly in the form of relative values to a "reference baseline" chosen to be the latest UMTS version available at that time (3GPP Release-6 HSPDA/HSUPA) as shown in table 2.4 whereas table 2.5 compares the absolute values of both LTE Release-8 main key performance requirements and UMTS Release-6 capabilities[9].

	LTE Release-8 requirements relative to UMTS Release-6	
	Downlink	Uplink
Avg. cell spectral efficiency [Times*Rel-6]	3-4	2-3
Cell edge spectral efficiency [Times*Rel-6]	2-3	2-3
Avg.user throughput [Times*Rel-6]	3-4	2-3

Table 2.4: LTE Release-8 relative key performance requirements

	Downlink		**Uplink**	
	UMTS	LTE	UMTS	LTE
Peak data rate [Mbps]	14.4	>100	11	>50
Peak spectrum efficiency [bps/Hz]	3	>5	2	>2.5
Avg. cell spectral efficiency [bps/Hz/cell]	0.53	>1.6-2.1	0.33	>0.66-1.0
Cell edge spectral efficiency [bps/Hz/user]	0.02	>0.04-0.06	0.01	>0.02-0.03
Avg. user throughput [bps/Hz/user]	0.05	>0.17-0.27	0.032	>0.07-0.11

Table 2.5: LTE Release-8 key performance requirements

LTE Downlink and Uplink performance evaluation conducted later by several international

[7] Was approved at RAN#28-June 2005 [69]
[8] Was approved at RAN#33-September 2006 [69]
[9] at the time of publishing (TR 25.913)

stakeholders of the telecommunications sector[10] confirmed the ability of E-UTRA of reaching the minimum peak data rates, user throughput and spectrum efficiency as defined by (TR 25.913 [74]) and the tests results were published in May 2007 through reports (R1-072444 [77]) and (R1-072261 [78]).

The project expanded after Release-7 was frozen in December 2007, but with less number of options comparing to what was specified by earlier work and was known officially as the Evolved Packet System (EPS), under which, two 3GPP sub-projects (work items) were running as illustrated in figure 2.12 [72], [76]:

- System Architecture Evolution (SAE): concerned in designing the new core network architecture and resulted in creating the Evolved Packet Core (EPC).
- Long Term Evolution LTE: aimed to evolving the RAN, the air interface as well as the User Equipment (UE) and it resulted in creating E-UTRAN.

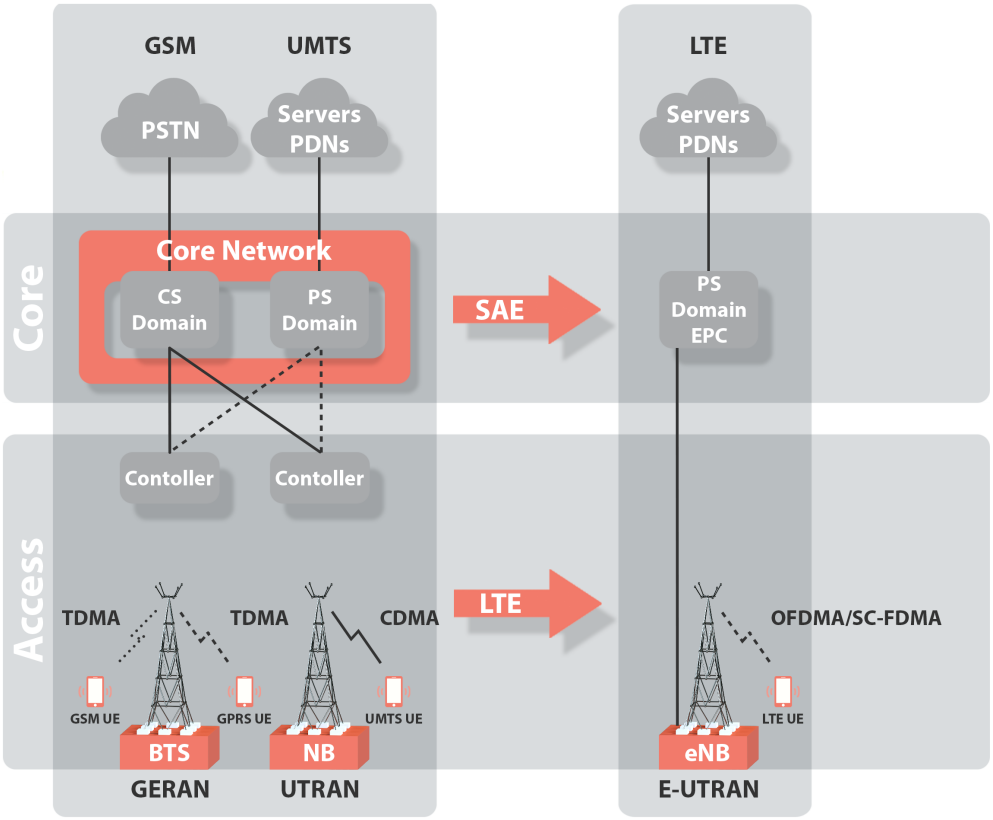


Figure 2.12: Evolution of the system architecture from GSM and UMTS to LTE

Although the term LTE was the name of part of the ongoing EPS, it became colloquially used -afterward even by 3GPP- to refer to the entire project while using E-UTRA[11] in radio specifications [69], [72].

The LTE access network is simply a network of base stations (evolved-NodeBs (eNBs)) forming a flat architecture 2.13. There is no centralized intelligent controller, eNBs are normally

[10] Ericsson, Huawei, InterDigital, Motorola, NEC, Nortel, Siemens, Qualcomm, Samsung and Texas Instruments

[11] The terms LTE and E-UTRA are synonymous [69]

inter-connected via the X2-interface and towards the core network by the S1-interface. The reason for distributing the intelligence amongst the base-stations in LTE is to speed up the connection set-up and reduce the time required for a handover which are considered as crucial parameters for the end-user, especially in online gaming and video streaming.

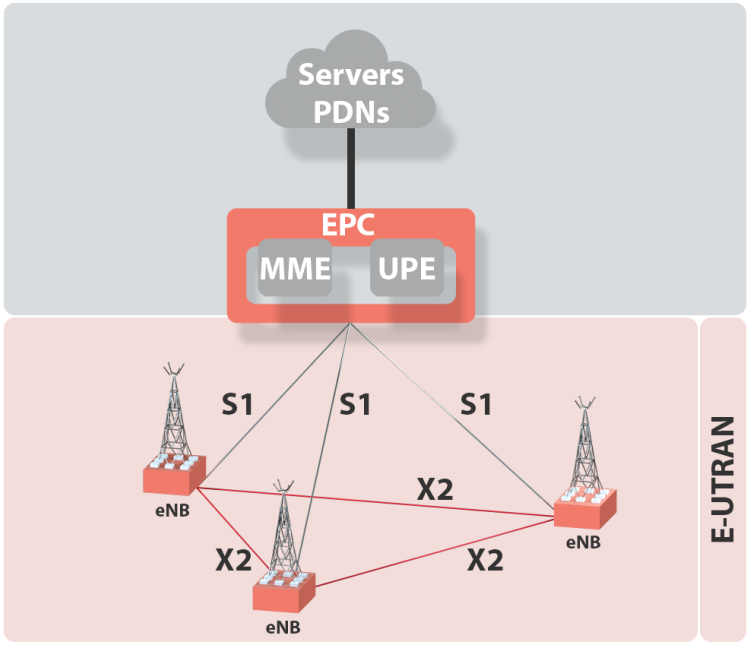


Figure 2.13: LTE Access Network

By December 2008 and when Release-8 was ready to be frozen, LTE was ready to function and to be deployed including almost all the essential features of the system as listed briefly in table 2.6, while postponing -due to the competition with IEEE WiMAX [72]- the least important peripheral ones to be covered by Release-9 which included some enhancements on LTE Release-8, and in turn was frozen in December 2009, at the same time that the first LTE commercial networks were being put in service in Norway and Sweden [72], [79].

Despite the entire performance evolution, this new system has passed over many obstacles faced the previous UMTS by applying major changes in all network levels which reflected into the system scalability, simplicity and universality. UMTS limited scalability can be easily observed in the multi-path fading it shows while trying to get a higher transmission speed by increasing the carrier bandwidth of the UMTS Wideband Code Division Multiple Access (WCDMA) air interface (standardised to 5MHz). WCDMA was replaced in LTE by OFDM which instead of using one full carrier bandwidth to spread the complete signal, it uses several 180kHz narrowband carriers and divides the data stream between them, leading to a higher bandwidth flexibility since assigning more or less narrowband carriers is all it takes to alter the bandwidth while maintaining all the narrowband channels characteristics, thus significantly reducing the multi-path fading effect. The simplicity surfaces in the transition from the traditional circuit switching used in UMTS for main voice and messaging services,

LTE Release-8 main features		
Core	Internet Protocol (IP) version	IPv4, IPv6
	Transport mechanism	Packet switching
	IP connectivity	During registration
RAN	RAN components	eNB
	Cell capacity	>400 users
Air Interface	Carrier bandwidth	1.4-20 MHz
	Multiple Access scheme	OFDMA/SC-FDMA
	Frame Duration	10 ms
	Transmission interval	1 ms
	MIMO antennas	Yes
	Operation mode	FDD and TDD
Latency	Cell-plane	<100 ms
	User-plane	<5 ms
Handover interruption	Real-time services	<300 ms
	Non real-time services	<500 ms
Mobility support	<15 km/h	Optimized
	15-120 km/h	High performance
	120-500 km/h	Maintained

Table 2.6: LTE Release-8 main characteristics

to a fully IP based packet switching approach for all types of network services including the radio base station backhaul connections and network inter-nodes interfaces, while giving the choice to the operators to decide which protocols to use below the IP layer, the thing that outstandingly simplifies the network design and implementation while keeping an entirely transparent and interchangeable physical layer, giving LTE devices the ability to support seamlessly fast handover (300ms for time critical services and 500ms for non time critical services) between GSM, UMTS and LTE sessions, awarding LTE its universality advantage over other systems [72], [74], [76], [80].

Simultaneously as the development of LTE, while 3GPP was completing Release-8 LTE work item, the International Telecommunication Union (ITU) was building upon the success of International Mobile Telecommunications (IMT)-2000 (3G) and working on preparing the requirements of the new global 4G mobile wireless broadband technology that will join the IMT family under the name IMT-Advanced through a 9 steps process within its strategic IMT future vision (2002) [80], [81], that starts with the issuance of a Circular Letter to invite proposals for radio interface technologies, and ends with implementation of the selected radio technology.

The first IMT-Advanced workshop was held in May 2007, followed by publicly sharing the circular letter (5/LCCE/2) [82] in March 2008, inviting all the concerned organisations to submit complete descriptions of their proposed Radio Interface Technology (RIT) or a Set of

Radio Interface Technologies (SRITs) that fulfills IMT-Advanced technical requirements and guidelines listed in International Telecommunication Union – Radiocommunication Sector (ITU-R) reports (M.2133, M.2134, and M.2135) until November 2009 [76], [82]–[85].

Driven by the fact that LTE at that time, as defined in Release-8 until March 2008, was not able to reach the minimum required performance for all the key features determined by IMT-Advanced, in addition to ensure a more efficient use of the additional IMT spectrum band that was devoted for mobile services in the World Radiocommunication Conference (WRC) 2007, and the will of meeting the future needs of operators and end-users, 3GPP rapidly organised the first workshop to discuss IMT-Advanced in April 2008. Right after the workshop, an ahead of time Release-9 study item was approved to prepare 3GPP RIT proposal for IMT-Advanced, and it resulted into two Technical Reports, the first of which (3GPP TR 36.913) is on the "**Requirements for further advancements for Evolved Universal Terrestrial Radio Access (E-UTRA) (LTE-Advanced)** [86]" published under Release-8[12], and the second (3GPP TR 36.913) to be a detailed Release-9[13]" **Feasibility study for Further Advancements for E-UTRA (LTE-Advanced)** [87]". Thereafter, 3GPP did an initial candidate submission of Long-Term Evolution-Advanced (LTE-Advanced) in September 2008 including a high level description of LTE-Advanced main features [88], and it was until RAN#45 October 2009 when the complete submission [89] took place as an SRITs[14], including self-evaluation results, the technology specifications, link budget templates as well as compliance templates for supported services, spectrum, and technical performance [71], [83], [90].

At that point, and in parallel to the candidate submissions assessment happening in ITU-R, individual Release-10[15] Work Items were created in 3GPP to carry out the work on the detailed standards of LTE-Advanced to introduce the enhancements endorsed in the aforementioned Release-9 Study Item as table 2.7 shows the new Rel-10 features that were integrated into LTE Rel-8 and Rel-9 [76].

Enhancement	**Work period**	**Document**
Carrier Aggregation	Dec.09 - Jun.11	RP-100661
Upload multiple antenna transmission	Dec.09 - Mar.11	RP-100959
Enhanced Downlink Multiple Antenna Transmission	Dec.09 - Mar.11	RP-100196
Relay Nodes	Dec.09 - Jun.11	RP-110911
Further enhancements to MBMS	Jun.10 - Mar.11	RP-101244
Self Optimizing Networks (SON)enhancements	Mar.10 - Jun.11	RP-101004
Minimization of drive tests	Dec.09 - Jun.11	RP-100360

Table 2.7: LTE-Advanced Release-10 main enhancements [69]

On the 21st of October 2010, ITU announced the completion of the assessment phase of the

[12]Was approved at RAN#40-June 2008
[13]Was approved at RAN#45-September 2009
[14]Includes an FDD RIT component and a TDD RIT component.
[15]Was frozen in March 2011

six proposals received in October 2009, declaring that **LTE-Advanced** and **WirelessMAN-Advanced**[16] have met all IMT-advanced requirements and officially designating them as true Fourth Generation technologies, and passing them to the last stage of IMT-Advanced process to finalise all the comprehensive technical characteristics to implement these two systems [72], [91].

LTE-Advanced, by a big margin, got more support from operators and hardware manufacturers than the IEEE WirelessMAN-Advanced, which assured that 3GPP LTE Release-10 and beyond (LTE-Advanced) is the world's prevailing 4G global wireless mobile broadband communications technology [72] with 655 LTE commercial deployments, 308 of them are LTE-Advanced installations, offering services worldwide through more than 4.7 billion active LTE connections of all releases occupying more than 48% of the market in the end of 2018, according to statistics done by 5g-Americas [92] and Ovum [93]. Moreover, LTE is still expected to be the dominant mobile communication technology for the few upcoming years and reaching a peak with 6 billion active connections in 2022, even with the 5G put in service as figure 2.14 shows [92].

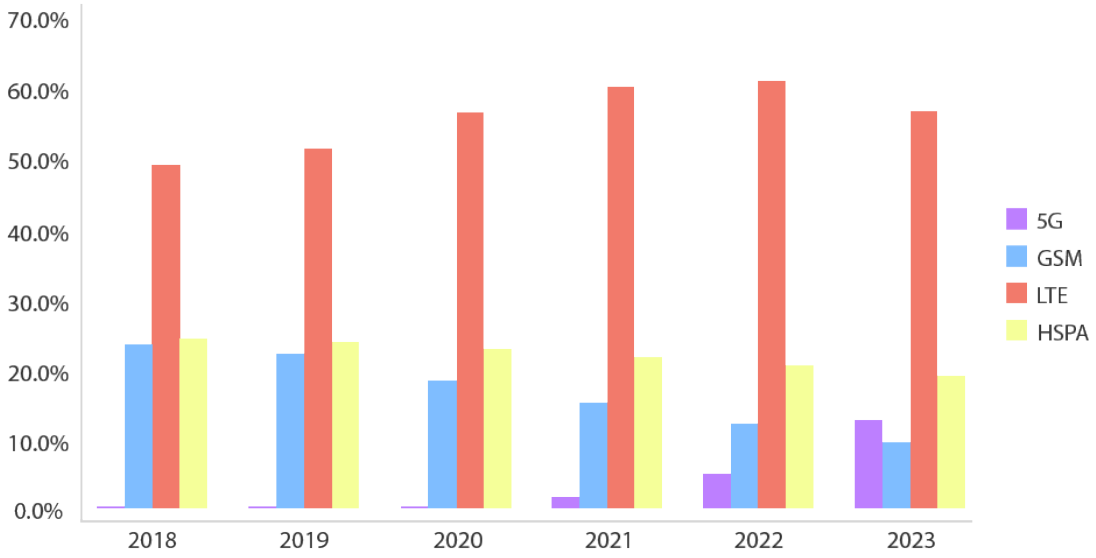


Figure 2.14: Global market share by technology forecast

Starting from 3GPP Release 12 [94], standardization of LTE-based D2D communication has been considered, as part of ProSe services, a new D2D interface (designated as PC5, also known as sidelink at the physical layer) was introduced for short range communications while allowing the long range communication between devices over the cellular network through the Uu interface. 3GPP started addressing the communication needs associated to ITS services as part of LTE Rel-14 V2V Work Item [95], by enhancing its services for vehicular use cases, specifically addressing high speed (up to 250Kph) and high density (thousands of nodes), including unicast, multicast and sidelink transport options [96]. As a result, the term LTE-V2X, or more generally Cellular-V2X (C-V2X) was introduced to incorporate the

[16] Also known as (mobile WiMAX 2.0) and (IEEE 802.16m)

PC5-based vehicular short range communication and the Uu-based long range communication, as shown in figure 2.15 The motivation to include direct communication in cellular standards for vehicular use-cases is two-fold: some vehicular use-cases require high message rates (in the order of 10 messages per second or more) and low delays, which are hard to achieve by indirect cellular communication and would most-likely require a high level of spacial reuse, i.e. high cost of deployment. Additionally, from the mobile operator point of view, direct communication can reduce the traffic (and resulting costs) in the core network, a fact that is also exploited by offloading in other domains [97], [98]. It is important to mention that C-V2X is yet facing different challenges such as its readiness for V2V safety applications, in addition that it operates in the same 5.9 GHz frequency band together with the vehicular communication specific technologies (e.g. ITS-G5) which introduces further challenges related to coexistence and spectrum sharing.

LTE-Advanced since then was getting enhancements and improvements with every 3GPP release; and all enhancements of LTE Rel-13, Rel-14 and beyond (if not related to 5G) are running under the trademark "LTE Advanced Pro".

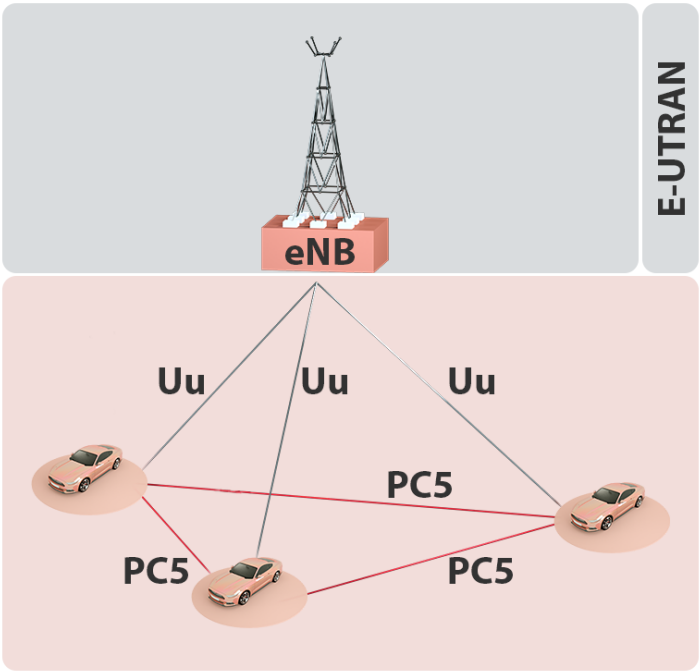


Figure 2.15: LTE-based vehicular communication

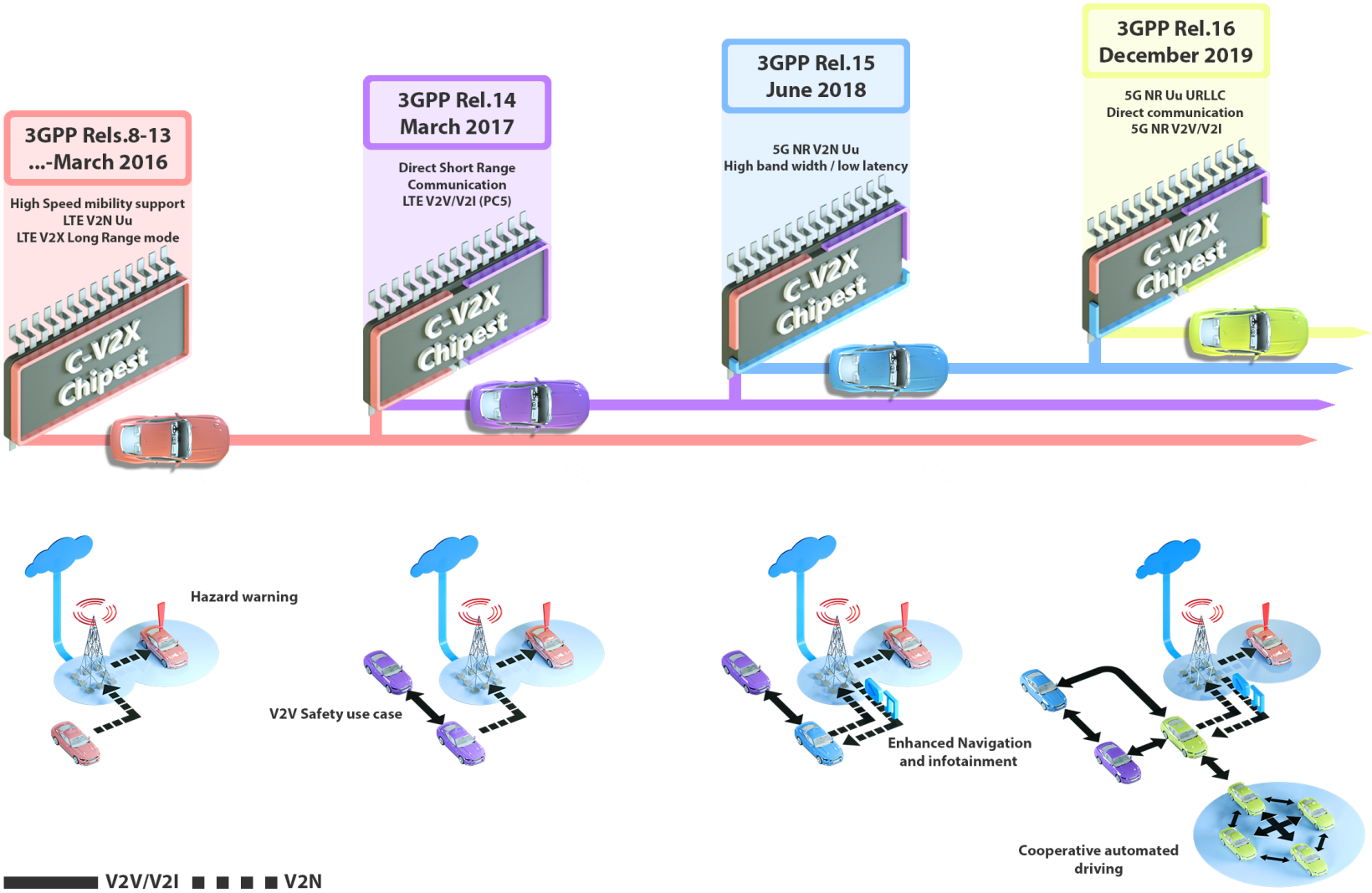


Figure 2.16: Standardisation of LTE-V2X and 5G-V2X at 3GPP

2.5 Sockets

Approximately 40 years ago (1979), the Computer Systems Research Group (CSRG) at the University of California, Berkeley (UCB) was tasked and funded by the Defence Advanced Research Projects Agency (DARPA) to develop their network (ARPAnet), and port the TCP/IP software -which was still undergoing development- to the UNIX operating system. The UCB already had their own version of UNIX that was known as Berkeley Software Distribution (BSD), thus a considerable part of the work was to create the interface between TCP/IP-based network and the networked applications that uses it. The team decided that the new interface should mainly rely on the existing UNIX system calls and algorithms while adding new functions only when it is absolutely necessary. In January 1983, the migration of the ARPANet to TCP/IP was officially completed when the new protocols were permanently activated, and 4.2BSD was the first widely available UNIX release including the TCP/IP network implementation in addition to a variety of networking tools and an Application Programming Interface (API) of the resulting interface that was called the *socket interface* or the *Berkeley socket interface*. Several performance improvements were added to the BSD software in the later years and even though all the networking code, both the kernel support (such as the TCP/IP and Unix domain protocol stacks and the socket interface), along with the applications (such as the Telnet and FTP clients and servers), were developed independently from the AT&T-derived Unix code. However, using any BSD release at that time required a source code license for Unix from AT&T. Therefore, starting in 1989 the Computer Systems Research Group (CSRG) at Berkely provided the first of BSD Networking Software releases (Net/1) followed by (Net/2) in 1991, until 4.4BSD-Lite (Net/3) in 1994 and finally 4.4BSD-Lite2 in 1995, which were all publicly available and contained all the networking code and various other pieces of the BSD system that were not constrained by the Unix source code license requirement and all remaining proprietary AT&T source code had either been replaced or, in the case of six source code files that could not be trivially rewritten, removed. This project had a great success that the final two releases from Berkeley (4.4DSD-Lite and 4.4BSD-Lite2) were then used as the base for other systems (BSD/OS, FreeBSD, NetBSD, and OpenBSD) in addition to many Unix systems that started with a version of the BSD networking code, including the sockets API (Berkeley-derived implementations) or even for other operating systems kernels that built their own networking implementation and socket API from scratch based on the BSD socket approach, such as the Linux's kernel networking implementation (1991) that was modeled on 4.3BSD and uses the BSD socket interface as the mean to communicate with the user level, in that it supports BSD sockets (with some extensions) and the full range of TCP/IP networking, making the The Berkeley socket interface the de facto standard interface for the most majority of networking applications [99]–[103].

Sockets were designed to be a method of Inter-Process Communication (IPC) that links asynchronous processes/applications with a single bidirectional channel, and they can be useful for both stand-alone and network applications since the processes that rely on socket-based communication can fully reside on the same system or different systems on different networks,

therefore sockets allow information exchange between processes on the same machine or across a network to distribute work to the most efficient machine, and they easily allow access to centralized data [101], [103], [104].

A *socket* is a bidirectional communications connection point (endpoint) that can be named and addressed in a network within a *communication domain.* Supporting different communication domains (Networking Domain), besides the TCP/IP was another wise step by the Berkeley team due to the aforementioned fact that when the BSD socket interface was being conceived, the TCP/IP protocol was still being developed and at the same time, a number of other competing protocols were being used or researched by different organizations, each of which is characterized by a set of properties that determine its communication domain:[99], [101], [103], [104].

- the supported *protocol family* including the underlying protocols, packets structure and communications facilities;
- the addressing scheme (*address family*) that indicates which type of addressing is being used, and consequently the format of a socket "address" as a method of identifying sockets, since every communication protocol specifies its own format for its networking address;
- the communication range whether between processes within the same local host or between different networked hosts.

In early BSD sockets implementations, it was visioned that communication domains can be made up of one or more protocols of a protocol family backed by one or more address schemes for one or more of the family's protocols intending to allow a single protocol family to use multiple address families. But in actuality, no protocol family supporting multiple address families has ever been defined and as a result, all the modern socket implementations define the protocol family prefixed macros to be synonymous with the corresponding address family ones. For this purpose Linux uses two sets of `C` macro constants, the first uses the prefix "`PF_`" that stands for the Protocol Family, while the second uses the prefix "`AF_`" that stands for the Address Family. both `PF_` and `AF_` macros are defined in the the `<sys/socket.h>` header file (Code 1), which sets the `PF_` (protocol family) value for a given protocol to be equal to the corresponding `AF_` (address family) value for that protocol, which simply means that the socket interface accepts either the `PF_` or the `AF_` macro to specify the domain to be used [99], [101]–[103], [105].

Generally speaking, communication domains can be seen -according to their use cases- as

- *local domain* provides communication between processes within the same local system using socket-type files that are addressed (named) with file system pathnames (e.g. `/tmp/sock`);
- *networking domains* use sets of protocols to provide local-remote communication using complicated addressing schemes, that are based on domain-specific *socket address structures* (`sockaddr_*`) to pass and receive addresses without requiring the socket API to recognize the addressing format.

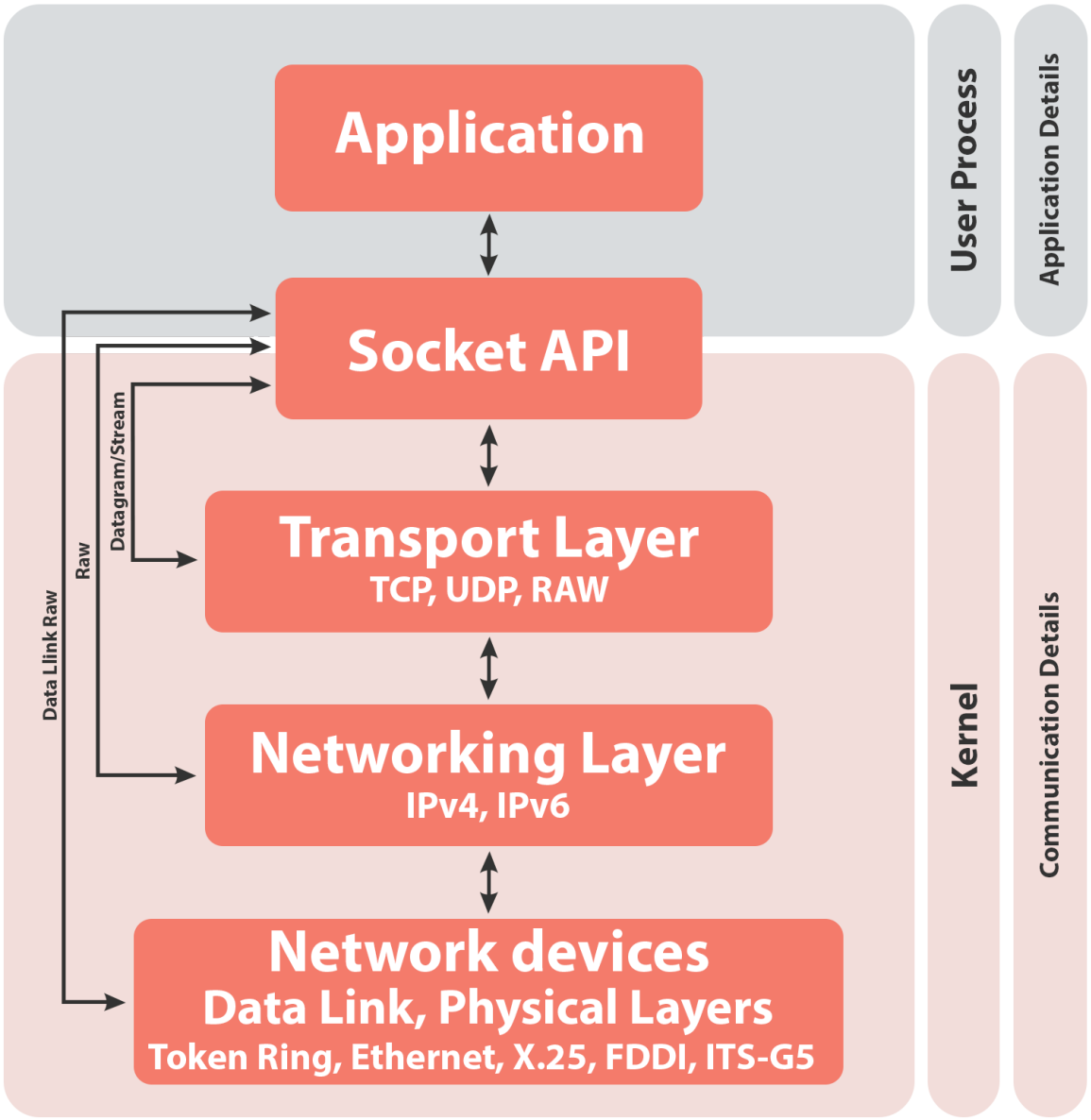


Figure 2.17: Socket API and Linux networking layers

However, in practice, the situation is not that simple and numerous communication domains are listed under each of the two main categories mentioned above, since new communication domains are defined and created by multiple technological and industrial entities to fulfill their specific needs and applications requirements, such as the Asynchronous Transfer Mode Permanent Virtual Connection (ATMPVC) or the Qualcomm IPC router interface protocol (QIPCRTR) which are defined in Linux as `AF_ATMPVC` and `AF_QIPCRTR` respectively, while other domains are designed to support general purpose local or network communication by reaching different layers of the network communication stack.

- `AF_LOCAL(AF_UNIX)` is used to provide communication between processes on the same host efficiently. Traditionally, UNIX domain sockets can be either unnamed (unaddressed) when created as pairs, or bound to a filesystem pathname using the `sockaddr_un` address structure.
- `AF_INET` provides communication between processes running on the same host or on hosts connected by a network (Internet), each of them has a functioning implementation of the Internet Protocol version 4 (IPv4), able to use the Internet family of protocols and addressed using the `sockaddr_in` address structure according to the IPv4 addressing format (an IPv4 address and a port number).
- `AF_INET6` provides communication between processes running on the same host or on hosts connected by a network (Internet), each of them has a functioning implementation of the Internet Protocol version 6 (IPv6), and addressed using the `sockaddr_in6` address structure according to the IPv6 addressing format.
- `AF_PACKET` is used to send and receive raw packets at the device level (datalink layer) allowing the user to implement protocol modules in user space on top of the physical layer, by directly communicating with the device driver by parsing and passing the addresses using the `sockaddr_ll` address structure.

Datatype	**Description**	**Header**
`char`	Basic single character	builtin
`int`	Integer variable	builtin
`unsigned char`	Positive only `char`	builtin
`unsigned short`	Positive only short `int`	builtin
`uint32_t`	Unsigned 32-bit `int`	`<sys/types.h>`
`uint16_t`	Unsigned 16-bit `int`	`<sys/types.h>`
`in_addr_t`	IPv4 address, defined as `uint32_t`	`<netinet/in.h>`
`sa_family_t`	Address family data type	`<sys/socket.h>`
`in_port_t`	TCP, UDP port, defined as `uint16_t`	`<sys/socket.h>`
`struct in_addr`	IPv4 address, defined as `uint32`	`<netinet/in.h>`
`struct in6_addr`	IPv6 address, defined as array of `unsigned char`	`<netinet/in.h>`

Table 2.8: Datatypes required by socket address structures

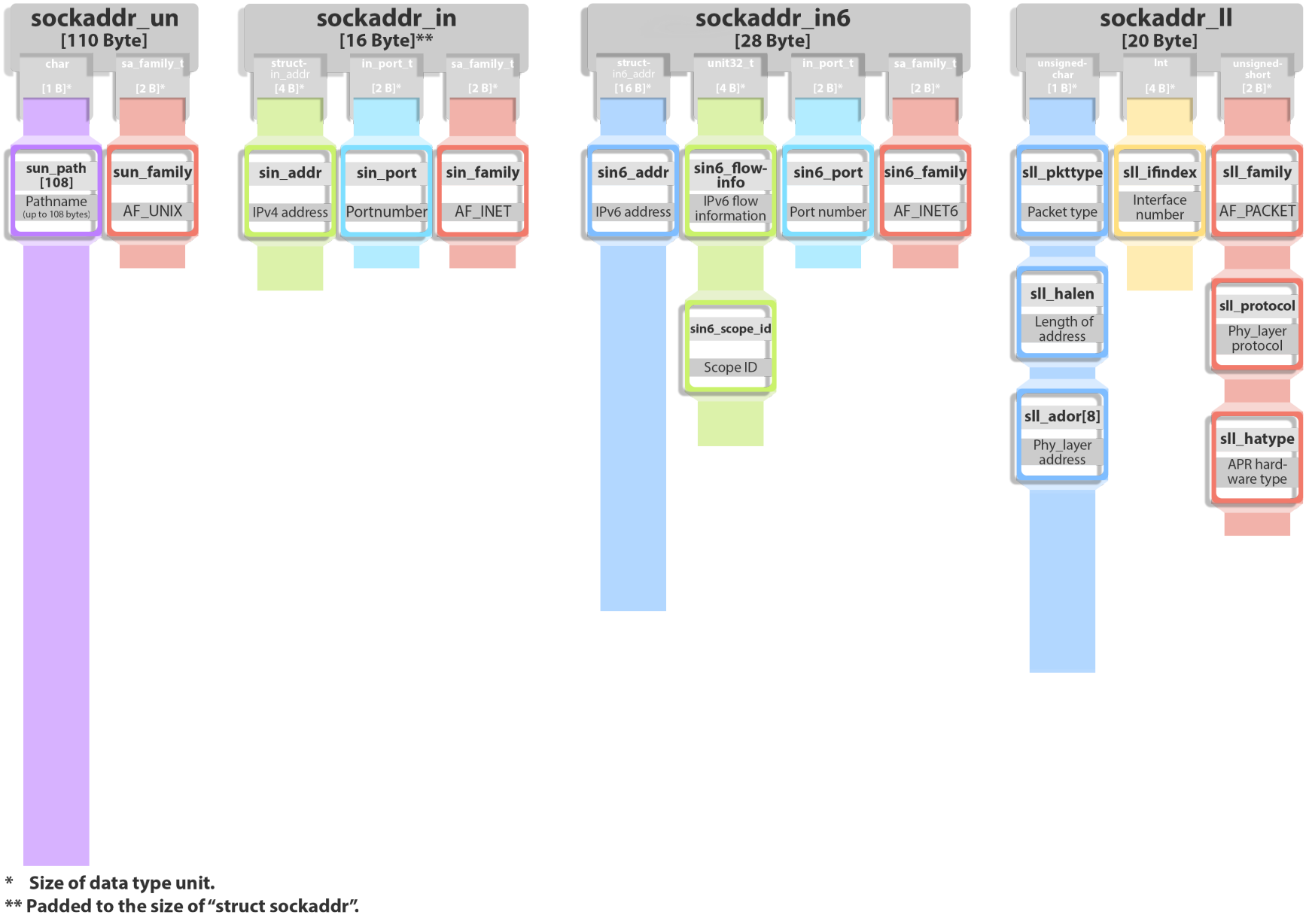


Figure 2.18: Socket Address Structures

```
[...]

#include <stddef.h>
#include <sys/types.h>

[...]

/* Protocol families.  */
#define PF_LOCAL  1 /* Local to host (pipes and file-domain).  */
#define PF_UNIX   PF_LOCAL /* POSIX name for PF_LOCAL.  */
#define PF_INET   2 /* IP protocol family.  */
#define PF_AX25   3 /* Amateur Radio AX.25.  */
#define PF_ATMPVC 8 /* ATM PVCs.  */
#define PF_INET6  10  /* IP version 6.  */
#define PF_PACKET 17  /* Packet family.  */
#define PF_QIPCRTR  42  /* Qualcomm IPC Router.  */

[...]

/* Address families.  */
#define AF_LOCAL  PF_LOCAL
#define AF_UNIX   PF_UNIX
#define AF_INET   PF_INET
#define AF_AX25   PF_AX25
#define AF_INET6  PF_INET6
#define AF_ATMPVC PF_ATMPVC
#define AF_PACKET PF_PACKET
#define AF_QIPCRTR  PF_QIPCRTR

[...]
```

Code 1: <sys/socket.h>

Depending on the communication domain in use, different choices and combinations for the communication type and supported protocols are available. Sockets are able to provide several types of communication according to the connection nature and reliability characteristics. In spite of that, it is possible to generally divide them into two major classes, namely *connection-oriented* and *connection-less* communication, with the main aspect of distinction to be whether or not a session is established between the socket peers before information exchange starts.

- **Connection-less communication** implies that no active, real-time connection is established, over which a dialogue or data transfer can take place. However, these sockets allow data to be exchanged in the form of predetermined-size messages called *datagrams*, that are routed according to designated addresses and names (included in each datagram) that identify where to find and how to reach each socket. Connection-less sockets preserve boundaries between datagrams; and exceeding the predefined datagram size causes the routers in the path between the socket peers to drop the unsuitable packets. The absence of session establishment or any verification mechanisms result in higher throughput, in addition to the ability of targeting multiple destinations at the same time (multicast and broadcast) without recreating new socket pairs, however, it is related to low data integrity and unreliable data transmission (except when used in the local communication

domain `AF_LOCAL`), since datagrams can be routed to their destination through different paths while no record of their existence is kept, and no attempt to recover from the transmission error is made, which may lead to out of order, duplicated or even lost messages at the receiving side. Connection-less sockets can be seen functionally -for the purpose of simplification- as a post-office box, if a message is put in the box, it cannot be absolutely guaranteed that the receiver will get the letter and it might be needed to wait for the response to approve the reception. The most important types of connection-less sockets are, the *datagram socket* (`SOCK_DGRAM` in Linux) which employs the User Datagram Protocol (UDP) in the Internet domain (`AF_INET` and `AF_INET6`) and the *raw socket* (`SOCK_RAW` in Linux) that gives the programmer a doorway to the underlying communication protocols that support socket abstractions with the ability of writing special or custom protocols, by allowing direct access to the IP layer packets and headers using the IP and Internet Control Message Protocol (ICMP) protocols when operating in the Internet domain, and even to the datalink layer packets and headers using the IEEE 802.3 set of protocols (Ethernet) when used in the `AF_PACKET` domain, the functionality that replaced the obsolete `SOCK_PACKET` socket type which has been used in old systems to receive raw packets directly from the device driver [99]–[101], [104], [106].

- **Connection-oriented communication** requires creating a connection before a dialog can start between the applications, providing a reliable, bidirectional, sequenced, and unduplicated flow of data while operate in connected pairs (connection-oriented socket can be connected to only one peer), and are widely used for client/server interaction especially in the Internet domain through *stream sockets* (`SOCK_STREAM` in Linux) that provides Byte-stream communication channel with no dividing lines or boundaries between the messages, and the bytes written on one end of the socket are received on the other end as one continuous stream of bytes with no record length, block size, or concept of a packet at the receiving end, and usually employs the Transmission Control Protocol (TCP). Some references use the example of the telephone call to simplify and explain the concept and functionality of connection-oriented sockets by resembling sockets to telephones, sockets' network addresses to telephone numbers and the ability of establishing a line of communication between a local socket (client) and a remote socket that offers a service (server) by indicating the address of the remote one to the process of dialing the telephone number of the person to be called, and when the receiver of the call (the server) answers the telephone, the connection is established and it remains active as long as both parties require it. Other connection-oriented socket types are, the `SOCK_SEQPACKET` which provides a reliable two-way connection-based data transmission path for datagrams of fixed maximum length using the TCP/IP while preserving message boundaries and data sequence; `SOCK_RDM` which provides a reliable datagram layer similar to the `SOCK_SEQPACKET` but it does not guarantee ordering; and the Datagram Congestion Control Protocol (DCCP) `SOCK_DCCP` that explicitly establishes a DCCP connection (consists of two separate unidirectional connections,

called half-connections) between hosts allowing them to negotiate and select the proper congestion control mechanism scheme, and setting up a Path Maximum Transmission Unit (PMTU) to avoid packets fragmentation, intending to mitigate the influence of UDP lack of any congestion control mechanism on the network performance. DCCP permits each of the half-connections can use a different congestion control scheme, allowing greater control over the network knowing that each half-connection is a one-way, unreliable datagram pipe [99]–[101], [104].

Not all combinations of socket *family* and *type* are valid. Table 2.9 shows some examples of the available combinations, along with the used protocol(s) for each pair. The cells filled with "N/A" are valid but do not have handy acronyms, while the blank cells means that no protocols are supported for corresponding combination.

Socket Address Family	**Socket Type**		
	`SOCK_DGRAM`	`SOCK_STREAM`	`SOCK_RAW`
`AF_LOCAL(AF_UNIX)`	N/A	N/A	
`AF_INET`	UDP	TCP,SCTP	IP,ICMP
`AF_INET6`	UDP	TCP,SCTP	IP6,ICMP6
`AF_PACKET`	Ethernet		Ethernet

Table 2.9: Socket family and type combinations

Despite the structural and operational differences between connection-oriented and connection-less sockets, they still share some common characteristics especially from a programming and management standpoint. The socket interface uses the concept of file descriptor and expands it to that of *socket descriptor*, which is a numerical reference returned by the function that created the socket, shared by the process and operating system to designate a specific active socket in a process and to be used by applications to utilize the socket, thus a socket can be seen -from an operating system perspective- as an abstraction for network communication, the same way as a file is an abstraction for file system communication. Socket descriptors alongside with file descriptors are allocated and maintained by the system kernel -in the same descriptor table for each process- sharing the same number space (*file unit* numbers), with no distinction between the two while allocating or retrieving the descriptors, therefore an application cannot have a file descriptor and a socket descriptor with the same value. When an application creates a socket, a pointer to the internal data structure that holds the socket information is put in the descriptor table and the index of that pointer is returned to the calling function as a socket descriptor, which is the only value the application needs to work with while using the socket because the Operating System (OS) handles all the underlying operations by retrieving the pointer from the descriptor table and accessing the information that the application requires. This information (socket data structure) is similar to that of a file, however, it is network oriented since it includes information such as the socket local address, the remote socket address (the socket at the other end of a connection) and the ports being used by the socket. Moreover, using the (file/file descriptor) analogy

to deal with sockets complies with the universality concept of UNIX I/O model and allows programmers to reference files and sockets interchangeably providing a great deal of flexibility since they can use sockets as if they were open files, with functions like `read(2)` [107] and `write(2)` [108] being able to operate upon both open files and sockets. This analogy also helps to visualize network communication that starts by opening a connection (socket) to the network, then data can be read from or written to the socket and when communication is no longer needed, the connection to the network can be closed and the resources reserved for the socket can be released. Thus, generally in a universal UNIX I/O environment, any application that needs to perform network input/output, can use the same four basic system calls -`open(), read(), write(),` and `close()`- in addition to the catchall `ioctl()` system call [99], [101], [103], [104]:

- **open()** to prepare for input or output operations.
- **close()** to stop previous operations and return resources.
- **read()** to get data and place in application memory.
- **write()** to put data from application memory and send control.
- **ioctl()** to set options or use features and operations outside the UNIX Universal I/O model, such as buffer sizes and connection behavior.

In Linux, Socket I/O can be performed using the aforementioned conventional system calls, or using a range of socket-specific system calls (table 2.10) that are especially designed to support network programming requirements for a variety of complex scenarios; and to abstract away all the details of different networking implementations and communication domains (TCP/IP, IPX, same host applications, inter-process communication, system loggers, print queues, etc). For instance, the `socket()` system call (Code 2) creates a socket (endpoint for communication) within a communication domain (protocol family) specified by the argument `domain`; communication semantics (mainly stream, datagram, or raw) defined by the argument `type`; and the `protocol` argument which specifies a particular protocol to be used with the socket. Normally, only a single protocol exists to support a particular socket type within a given protocol family, in which case protocol can be specified as 0. A successful `socket()` call returns a file descriptor for the newly created socket (of type `int`), while on error, (-1) is returned, and `errno` is set appropriately. For example, to create a raw IPv4 web socket to connect an application directly to the IP layer without using either the TCP or UDP transport (Code 3), it is required to specify `AF_INET` for the address family, `SOCK_RAW` as a socket type, while the `protocol` can be set based on the application's needs to any valid IANA IP protocol defined in RFC 1700 [109], [110] assigned numbers or their corresponding macros in Linux, such as the `IPPROTO_UDP` and `IPPROTO_TCP` to receive the type of packets specified (UDP, TCP) right from the Transport layer with their layer 4 (TCP/UDP) headers; or to set the layer 4 header in case of outgoing packets. Another possible option is `IPPROTO_RAW` to bypass the Transport layer to the Network layer and access the IP protocol directly.

```c
#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

Code 2: socket() system call

```c
#include <sys/socket.h>
#include <netinet/in.h>

int sockfd = socket(AF_INET, SOCK_RAW, int protocol);
```

Code 3: Create an IPv4 raw web socket using socket() system call

System Call	Description
socket()	create an endpoint for communication (socket)
bind()	bind a name (address) to a socket
connect()	initiate a connection on a socket
listen()	listen for connections on a socket for incoming connections
accept()	accept a connection on a listening socket
getsockname()	get socket name (address)
getpeername()	get name (address) of connected peer socket
socketpair()	create a pair of connected sockets
send()	send a message on a socket when it is in a “connected” state
recv()	receive a message from a socket when it is in a “connected” state
sendto()	send a message on a socket to a specific address (no connection needed)
recvfrom()	receive a message from a specific socket (no connection needed)
shutdown()	shut down all or part of a full-duplex connection on the socket
setsockopt()	set options on sockets
getsockopt()	get options on sockets
sendmsg()	send a message (passed by pointer) on a connected socket
recvmsg()	receive a message from a socket using a “msghdr” structure
accept4()	nonstandard Linux extension, accept a connection on a socket with “flags”
recvmmsg()	an extension of recvmsg() to receive multiple messages on a socket
sendmmsg()	an extension of sendmsg() to send multiple messages on a socket

Table 2.10: Linux socket-specific system calls

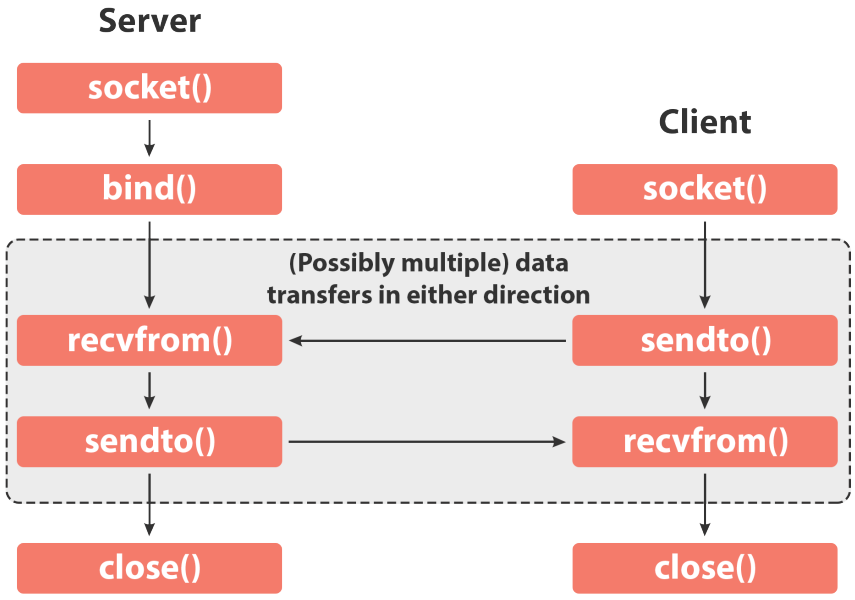


Figure 2.19: Connection-less Socket Communication

	Raw	ICMP	UDP	TCP
Overhead[bytes]	20–60	20–60+[4]	20–60+[8]	20–60+[20–60]
Message Size[bytes]	65,535	65,535	65,535	N/A
Reliability	Low	Low	Low	High
Message Type	Datagram	Datagram	Datagram	Stream
Throughput	High	High	Medium	Low
Data Integrity	Low	Low	Medium	High
Fragmentation	Yes	Yes	Yes	Unlikely

Table 2.11: Packet Type features [105]

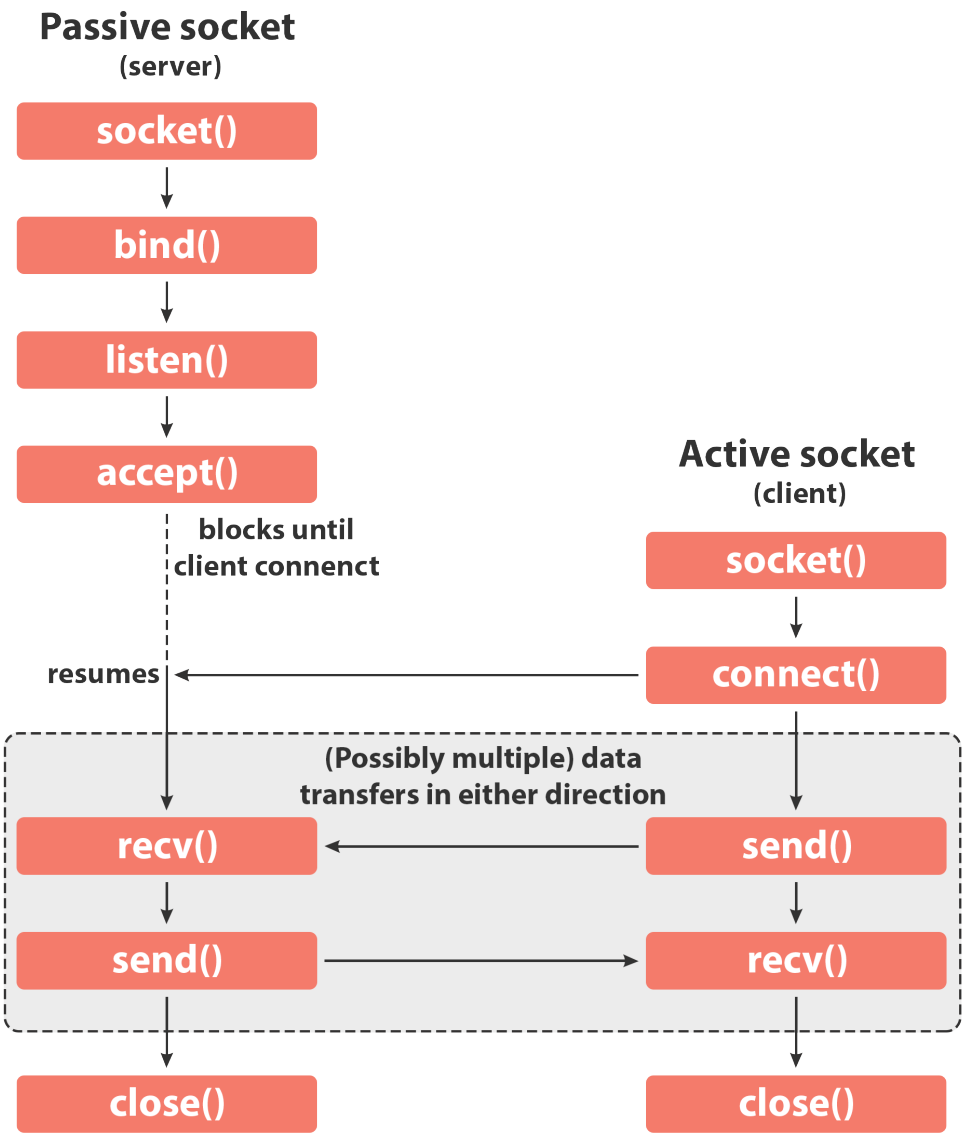


Figure 2.20: Connection-Oriented Socket Communication

2.6.1 Overview and Model Architecture

The AMQP is an open standard application layer protocol that uses TCP for reliable passing of business messages between applications or organizations. It securely connects systems, feeds business processes with the information they need and reliably transmits onward the instructions that achieve their goals. AMQP is able to provide interoperable communication between applications in different organizations that use different platforms and technologies in reliable time-independent manners, since it reliably operates at a distance, or over poor networks even when systems are not available simultaneously [111].

There are currently two completely independent versions of Advanced Message Queuing Protocols, namely *AMQP 0-9-1* and *AMQP 1.0*. However, despite the name, they are completely different messaging protocols that essentially share nothing at the wire level. Each of AMQP 0-9-1 and AMQP 1.0 has its own model, scope and topology, for instance, symmetry is one of the key differences, viz, AMQP 0-9-1 protocol is asymmetric, in that each connection is defined to have a conforming "client" end, and a messaging middleware server (also called "brokers") end. As such, it can be described to be very broker-oriented, while on the other hand AMQP 1.0 is fully symmetric and places no such constraints on the roles of connection endpoints that it permits broker-less point-to-point communication [112], [113].

Figure 2.21: Overall AMQ-Model

This work utilizes the *RabbitMQ* open source message broker to implement the *AMQP* communication, and since *RabbitMQ* was originally developed to support *AMQP 0-9-1* which is the "core" protocol supported by the broker, therefore, in the scope of this document, the term AMQP only refers to AMQP 0-9-1 and this section is dedicated to introduce its model and general architecture [114].

AMQP is not symmetrical, and any full AMQP communication scenario consists of clients that produce and consume messages; and servers that accept clients connections and implement the AMQP message queuing and routing functions, hence, at a high level, three active participants and one item must exist in any AMQP communication [112]:

- The **message** is the application data passed from client (publisher/consumer) to server (broker) and from server to client, that can correspond to an application-level message,

one frame of a data stream, or even a file transfer, etc.

- The **publisher** (also known as producer) is a client application that publishes messages to the server (messaging broker).
- The **messaging broker** is the server that distributes the message to different clients (consumers) according to defined rules.
- The **consumer** client application that requests messages from a messaging broker.

The AMQP server must not remove or modify any existing message information. However, it can add information to the messages in the form of content headers in order to specify some messages characteristics, such as:

- **Message persistence:** reliable message delivery is always prone to some threats like a serious network failure, server crash or overflow. Therefore, AMQP gives users the ability to mark there messages as *persistent* to be securely held on the server's disk and guaranteed to be delivered.
- **Message priority level:** higher priority messages have mainly two advantages, first of which that they are sent ahead of lower priority messages waiting in the same message queue, and the second is to be relatively protected against message discarding that servers might apply in some cases to maintain a specific QoS level.

Since AMQP is a network protocol, the publishers, consumers and the broker can all reside on different machines of possibly different architectures, and even implemented in different programming languages, and since interoperability demands these entities to be able to communicate in any case, the AMQ-Model (figure 2.21) specifies a modular set of components and standard rules which must be made available by an AMQP compliant server implementation in order to achieve the semantics defined in AMQP specification. There are three main types of component (collectively referred to as AMQP entities), which are connected into processing chains in the server to create the desired functionality as defined by AMQP parlance are [112], [115]:

- The **exchange** receives messages from publisher applications and routes these to "message queues" within the server, based on a specific criteria, usually message properties or content.
- The **message queue** stores messages until they can be safely forwarded to and processed by a consuming client application (or multiple applications).
- The **binding** defines the relationship between a message queue and an exchange and provides the message routing criteria.

AMQP allows -for administrative convenience- exchanges, message queues and their corresponding objects (bindings, user permissions, policies) to be logically grouped into completely isolated environments within the same server (broker) known as **virtual hosts** granting resources separation, and making it possible for a single broker to host and manage multiple independent environments (multi-tenant system). Thus, a Virtual Host (vhost) is a collection of exchanges, message queues and associated objects forming independent server

domain, comprising its own name space with one authentication scheme to be shared between all virtual hosts on the same server. AMQP offers no mechanisms for creating or configuring virtual hosts. However, this is done in an undefined manner within the server and is entirely implementation-dependent [112], [114].

The power of AMQP comes from its ability to create queues, exchanges and bindings at runtime, and to chain these together in ways that go far beyond a simple mapping from an address to another. This feature makes AMQP a programmable protocol in the sense that it provides runtime-programmable semantics to the application developers giving them a lot of freedom, through two main aspects:

- Arbitrary exchange and message queue types that can be created by the programmer and added as server extensions at runtime via the protocol, to extend the predefined standard types.
- User defined bindings that can be created at runtime via the protocol to wire exchanges and message queues together to create any required message-processing system.

A functioning implementation of the AMQ-Model has the following flow of events as generally depicted in figure 2.22:

1. *messages* published by the *publishers* are sent to a broker's *exchange*;
2. the *exchange* accepts the messages;
3. and then routes them to the *message queues* based on rules defined in the *bindings*;
4. *messages* stay in the *queues* and could be buffered in memory or on disk for *consumers* that are not able to accept them fast enough, so they can fetch/pull *messages* from *queues* on demand.
5. thereafter the *broker* forwards the messages through the *queues* to different subscribed *consumers* applications based on user defined criteria.

2.6.2 Message Acknowledgements

From the point of view of AMQP designers, networks are not reliable enough to deliver all messages every time, and applications might not be able to process some messages, therefore the AMQ-Model has the notion of message **acknowledgements**, which is a formal signal from the client application to the broker (a message queue within the broker) indicating that it has successfully processed a message. When acknowledgements are enabled, a message is not completely removed from a messages queue until the broker receives a notification for that message. AMQP defines two possible acknowledgement models [112], [114]:

- *Automatic*, in which an acknowledgement is sent automatically from the client to the message broker as soon as the message is delivered to an application, and the server removes the content from the queue right after it gets that notification.
- *Explicit*, in which the client chooses when to send the acknowledgement for each message (or a group of messages) that it has processed.

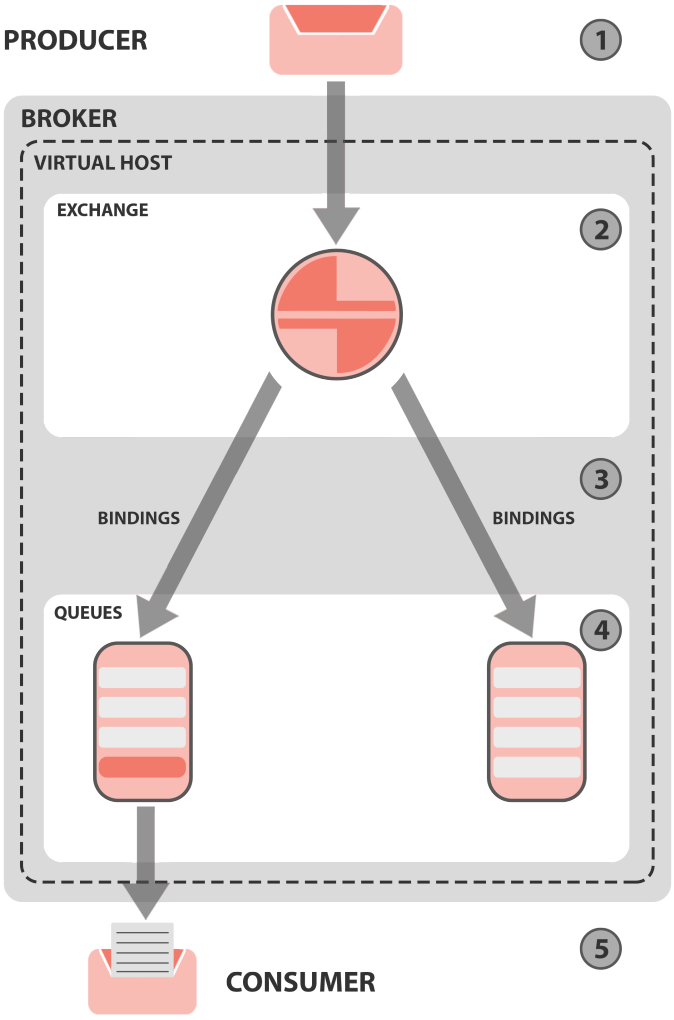


Figure 2.22: Overall Advanced Message Queuing Model

2.6.3 Connections and Channels

Another very important idea to mention is the concept of **Connections** and **Channels** (figure 2.23). A connection in AMQP is a long-lived network connection (e.g. TCP/IP socket connection) with all the underlying tasks such as initial authentication, IP resolution, and networking between the client (publisher/consumer) and the broker server, more specifically a single virtual host on the broker server. However, in many cases, an application needs to have multiple connections to the broker, and even though opening several connections to one or more AMQP servers from a single client is entirely applicable, it is undesirable to keep many simultaneous open TCP connections, because doing so, heavily consumes system resources since opening new TCP connections is considered to be an expensive procedure in terms of processing power, network resources and firewall configuration compared to using an already open connection and the disparity grows even faster with frequent opening and closing connections. That being the case, AMQP provides a way to multiplex a "heavywight" TCP connection into several "lightweight" connections and forgoing the need to reauthorize

and open a new TCP stream, causing the protocol to be more resources and firewall "friendly" since it significantly reduces the need of opening new TCP connections, hence, port usage becomes more predictable. This "lightweight connection" is called a channel which is defined as a bi-directional stream of communications between two AMQP peers that only exists in the context of a connection and can be multiplexed and maintained with other channels under the same TCP connection, while each channel is independent of the others and can perform different functions simultaneously with other channels, the available bandwidth being shared between the concurrent activities. All channels within a single connection work with the same virtual host and every protocol operation the client performs is transmitted over a channel, therefore each operation's frame is numbered with a *channel number* to be recognized by the clients and the brokers allowing safe interleaving of different channels' frames over one connection and at the same time, a strict sequence of frames to be retrieved for any given channel [112], [114].

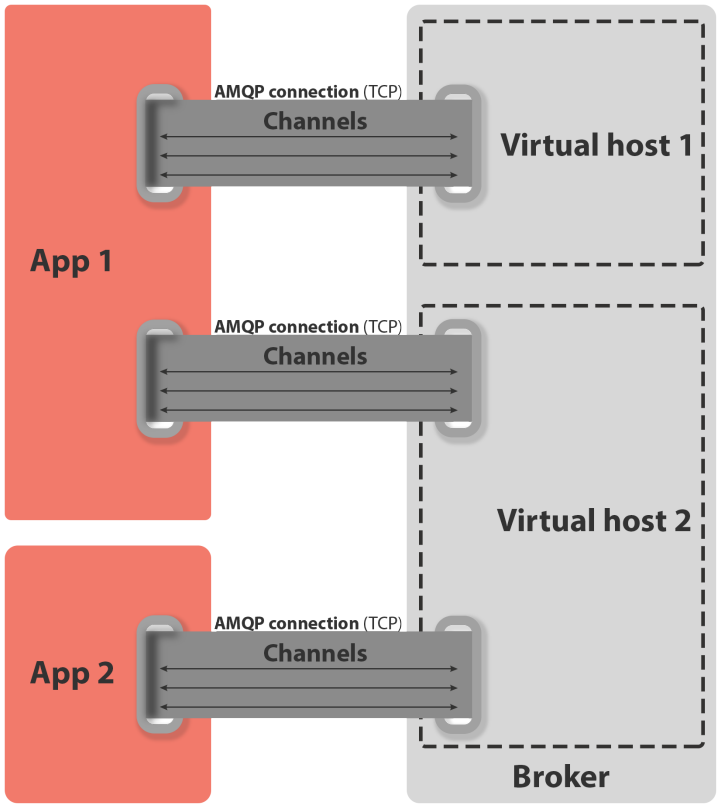


Figure 2.23: AMQP Connections and Channels

2.6.4 Protocol Commands (Classes & Methods)

Taming the complexity of middleware was a main challenge for the AMQP team while designing the protocol structure, and sending instructions between connection peers was at the heart of this challenge. Their approach was to model a traditional API and structure the

protocol as **classes** which comprise **methods**, and to define each method to do exactly one thing, and to do it well. A method, by definition, is a specific type of AMQP command frame that passes instructions from server to client (client-side method) or from client to server (server-side method), while a class is a logical collection of methods that covers a specific functional domain and operates over a specific type of functionality of the protocol structure. The following protocol classes are defined in the AMQP specifications:

1. **The Connection class** contains methods to work with socket connections, such as the *open* method to open a connection to a virtual host, and the *close* method to request a connection close.
2. **The Channel class** contains methods to work with channels, such as the *open* method to open a channel for use, and the *flow* method to enable/disable data flow from a peer.
3. **The Exchange class** contains methods to work with exchanges such as the *declare* method to verify if an exchange exists and create one if needed, and the *delete* method to delete an exchange.
4. **The Queue class** contains methods to work with queues such as the *bind* method to bind a queue to an exchange, and the *purge* method to remove all messages from a queue which are not awaiting acknowledgment.
5. **The Basic class** contains methods to work with basic content and operations such as the *publish* method to publish a message to a specific exchange, and the *consume* method to asks the server to start requesting messages from a specific queue.
6. **The Transaction class** allows *publish* and *ack* operations to be batched into atomic units of work, that is, all the operations grouped into a transaction have to be completed successfully or none of them will. This class contains methods to perform the aforementioned goal such as the *select* method to sets the channel to use standard transactions, and the *commit* method to commit the current transaction.

Methods in some cases can form logical pairs, and each pair consists of two methods, a "request" that is normally sent by a client and a "response" that is normally sent by the server to confirm the accomplishment of the requested operation, as an example, the methods *declare* and *declare-ok* of the exchange class form a pair that starts by the client sending a "request" to the broker to declare a new exchange using the *declare* method, and if the operation succeeds on the broker side, it sends a "response" with the *declare-ok* method. Not all the AMQP methods have counterparts ("response" method), while some others have more than one possible "response". For example, the most widely used method of the basic class being the *publish* has no corresponding "response" methods, and the *get* method of the same class that provides a direct access to the messages in a queue, has two possible "responses" being the *get-ok* method that delivers a message to the client following a get method, and the *get-empty* method to tells the client that the queue has no available messages. Using this analogy, two distinct method dialogues can be defined:

- Synchronous request-response, in which one peer sends a "request" and the other peer sends a "response", and these methods are used for functionalities that are not performance critical.

- Asynchronous notification, in which one peer sends a method but expects no reply, and these methods are used where performance is critical.

2.6.5 Exchanges and Exchange Types

Messages in AMQP are not sent directly to a queue; instead, the producer publishes messages to an exchange. An exchange is a broker's named entity exists within a virtual host on an AMQP server, it acts as a routing agent that receives messages from a producer application and optionally routes these either to the broker's internal services or to message queues within the same vhost, based on the message's routing information (routing keys) included in the messages header and sent by the publisher; on a certain criteria specified by the *bindings*; and on the exchange internal structure and algorithms (exchange type). It is possible as well that an exchange routes a single message to different queues in parallel, creating multiple instances of the same message that are independently consumed; or that no matching queue can be found for the message and is simply dropped. Generally, and regardless of the functionality, exchanges can be configured upon creation to have one of the following attributes:

- Durable: survive a server (broker) restart and last until explicitly deleted by the user.
- Temporary: last until the server shuts-down and they have to be re-declared when the server becomes online again.
- Auto-deleted: last until they are no longer in use.

At start-up, each AMQP server pre-creates a set of exchanges that can be used -but not deleted- by client applications. In addition, applications are free to create, share, use and delete exchange instances for their private use, within the limits of their authority. AMQP vocabulary neither includes a method with the name "create" nor with "delete", instead it uses an assertive "declare" and "destroy" methods which respectively mean "create if not present, otherwise continue" and "delete the exchange irreversibly".

It is plausible that different applications and use cases have different requirements in terms of message distribution and routing scheme, thus AMQP defines a number of standard and mandatory exchange types, and each type implements a specific matching method and algorithm. It also recommends other exchange types and keeps the door open for each server implementation to add its own types. The standard exchange types as defined by the AMQP protocol specification [112] are:

- **The Direct Exchange** uses one message attribute -the *routing key*- to deliver messages to queues by direct comparison between the *routing key* the queue uses to bind to the exchange (e.g. K) and the *routing key* encapsulated in the message header (e.g. R), and only passing the message to the queue if both routing keys are exactly the same (K = R). For instance, figure 2.24 shows a simple scenario of using one direct exchange for "etsi_standards" with two queues bound to it, Queue A (etsi_its_queue) with the routing key "etsi_its", and Queue B (etsi_sim_queue) with the routing key "etsi_sim". When a message with the routing key "etsi_its" arrives at the direct exchange from the Producer; the exchange routes it to the queue with the routing key that exactly matches

the routing key of the message, in this case to Queue A (etsi_its_queue). Brokers pre-declare at least two direct exchanges in each virtual host: one named **amq.direct** that works exactly like any other user declared exchange, and one with no public name (empty string), but with a very special feature that makes it serve as the **default exchange** for *publish* methods, because every queue that is created is automatically bound to it using the queue name as a routing key. This feature might make it seem as if it is possible to deliver messages directly to queues, while technically this is not happening.

- **The Fanout Exchange** unconditionally passes a copy of all the messages it receives from the producers to all the message queues bound to it (no arguments binding) regardless of the message routing key. Fanout exchanges are ideal for the broadcast routing of messages, as graphically represented in figure 2.25 where a fanout exchange for "5g news" is broadcasting copies of all the messages it receives from the producer to all three queues bound to it. An exchange named **amq.fanout** must be declared by the broker in each virtual host at start-up.
- **The Topic Exchange** routes messages to one or multiple queues based on matching between a message routing key and the *routing pattern* that was used to bind a queue to an exchange. The routing key used for a topic exchange consists of zero or more words delimited by periods (**.**). Each word may contain the letters A-Z and a-z and digits 0-9. The routing pattern follows the same rules as the routing key, with the additions that it can contain an asterisk ("*") to match one word in a specific position of the message routing key; and a hash symbol ("#") to match zero or more words. Thus a routing pattern (*.v2x.#) matches the routing keys (it2s.v2x) and (it2s.v2x.pt) but not (v2x.pt). It is possible to determine all bindings for a given routing key, and so to rapidly find the message queues for a message. Figure 2.26 depicts a simple example of a topic exchange for "it2s_projects" with three queues: Queue A (portugal_projects), Queue B (all_projects) and Queue C (partners_projects) bound to it using the routing patterns (it2s.eu.pt.#), (it2s.#) and (it2s.eu.*.co) respectively. A message with a routing key (it2s.eu.pt.pasmo) will be forwarded to Queue A and Queue B, while a message with the routing key (it2s.eu.scoop.co) will only be routed to Queue C. AMQP specification [112] suggests one use case of topic exchange that is to hold a set of all known routing keys, and update this when publishers use new routing keys which makes it a good option for distributing data relevant to specific geographic location. This exchange type is optional but if implemented, at least one topic exchange named **amq.topic** must be pre-declared within each virtual host.
- **The Headers Exchange** routes messages based on attributes taken from the message header property rather than a routing key, and designed to be used in cases where multiple routing attributes are easier to be expressed as message headers than a routing key. A message queue is bound to a header exchange through a table of arguments, and each field of the table is a pair (*name-value*) holding the name of the message header attribute to be matched and the value it should hold. The table of arguments may

contain one or more entries with the *value* entry of each field being optional, and in the case of using multiple entries table for binding a queue to a header exchange, one extra piece of information is needed by the broker to achieve more flexible and efficient routing. In reality, this extra piece of information is a special bind argument that will control how the rest of the matching algorithm is executed. The name of this argument is "x-match", it is passed as a (*name-value*) pair in the arguments table while creating the binding between a queue and a header exchange, it can take one of two values:

- "all" mandates that all the other pairs in the arguments table must match the headers property of a message for that message to be routed (i.e. an AND match).
- "any" mandates that the message should be routed if any of the fields in the headers property match one of the fields in the arguments table (i.e. an OR match).

A field in the bind arguments matches a field in the message if either the field in the bind arguments has no value and a field of the same name is present in the message headers or if the field in the bind arguments has a value, and a field of the same name exists in the message headers and has that same value.

Figure 2.27 shows an example of a header exchange for "pasmo" with three queues:

- Queue A, bound to the exchange with arguments (*name-value*): format=pdf, type=report, x-match=all.
- Queue B, bound to the exchange with arguments (*name-value*): format=db, type=radar_data, x-match=any.
- Queue C, bound to the exchange with arguments (*name-value*): format=pkg, type=arch_pkg, x-match=all.

Three messages are published and three different scenarios can be realized:

- Message 1 is published to the exchange with header arguments (*name = value*): format = pdf, type = report, therefore it is delivered to Queue A since all (*name-value*) pairs match.
- Message 2 is published to the exchange with header arguments (*name = value*): format = db which matches one field of the binding arguments, and since the x-match field is equal to the value "any", therefore this message delivered to Queue B.
- Message 3 is published to the exchange with header arguments (*name = value*): format = pkg, and even though this argument matches one of the binding arguments. However, this message will not be delivered to any queue because the queue is configured (using x-match=all) to match all of the headers.

- **The System Exchange** is an optional exchange type that passes messages to system services with a name that matches the messages routing key.

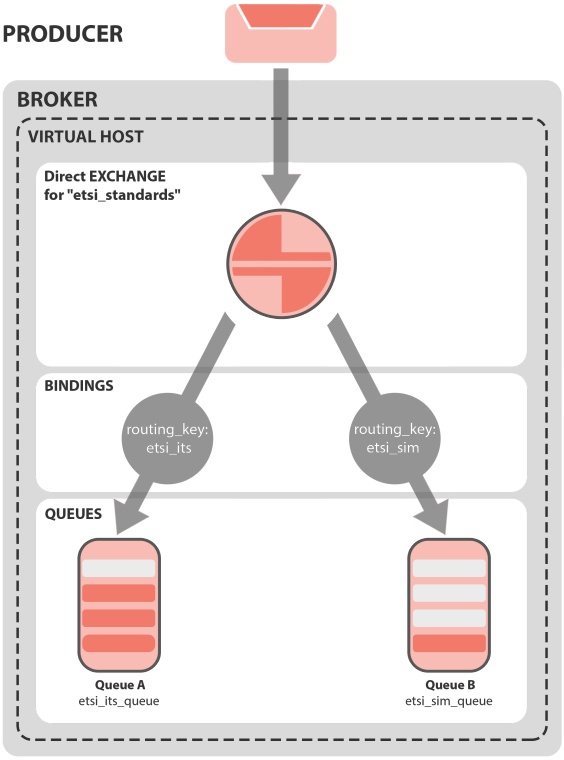


Figure 2.24: AMQP Direct Exchange

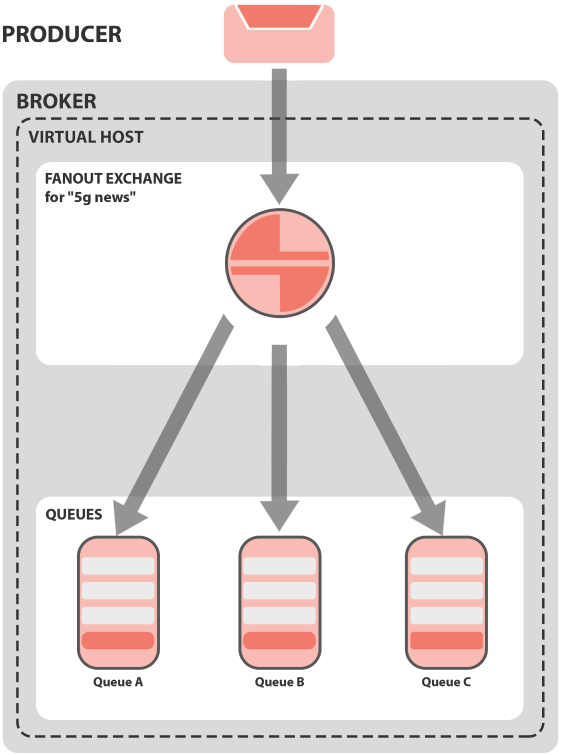


Figure 2.25: AMQP Fanout Exchange

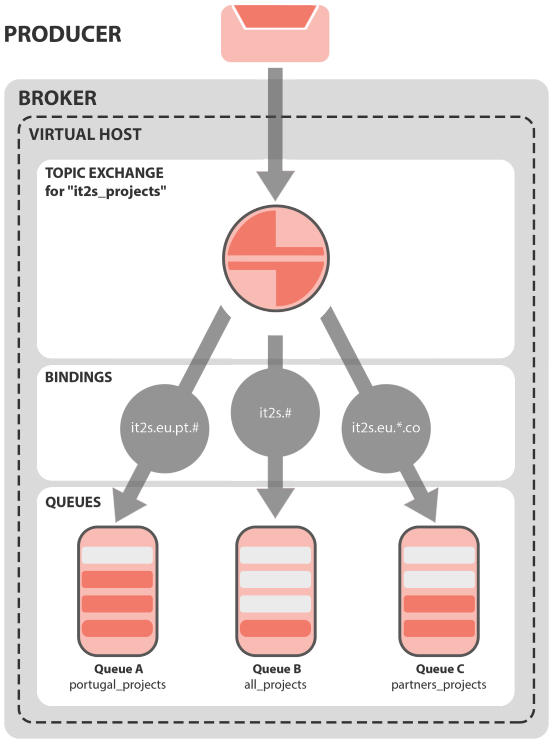


Figure 2.26: AMQP Topic Exchange

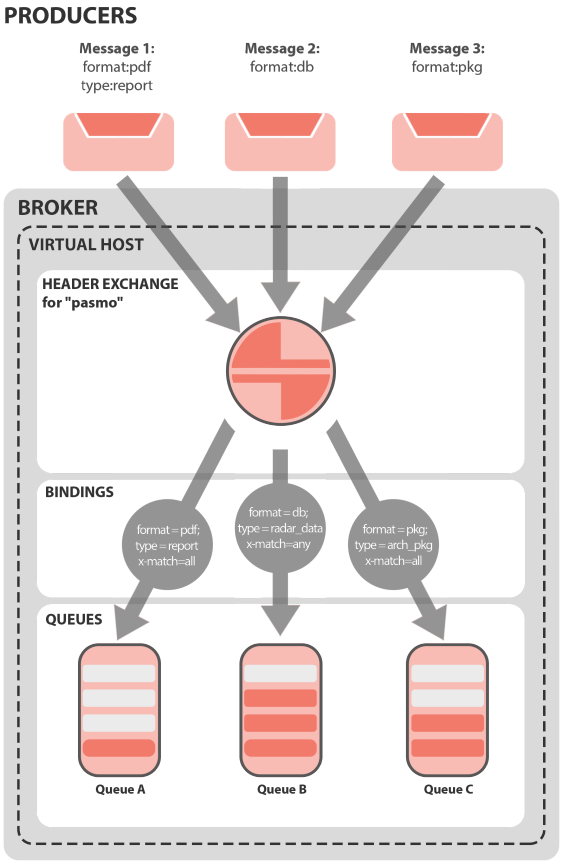


Figure 2.27: AMQP Header Exchange

2.6.6 Queues

AMQP queues are named entities that can be freely created, shared, used and destroyed by client applications on the server disk or memory or some combination of both to hold messages and forward them to consumer applications, or more accurately, to registered consumer applications that *subscribe* to the queue. When creating a message queue, a client application can select some important properties [112], [114], [116]:

- **name** to reference the queue by applications on a per-virtual host basis. The queue name can be set by the user, or left unspecified to be generated by the server and provided to the client.
- **exclusivity** determines if the queue belongs to only one connection or if it can be shared by multiple connections. If a queue is set to be exclusive, it belongs only to the current connection and is to be deleted when the connection closes.
- **durability** is the property which affects the queue's resistance to server restarts and zero subscriber cases. Similarly to exchanges, a queue can be *durable*, meaning that it is persisted to disk and automatically becomes present and active after a server restarts; *temporary* that is private to one consumer and deleted when that consumer disconnects (*unsubscribes*); or *shared* that can be used by multiple consumers and is deleted when the last consumer disconnects.

Technically, a queue is a named FIFO that can hold independently different types of content (messages) on behalf of one or a set of consumer applications, and then distribute the messages between these clients following a rule that a message is never sent to more than one client, unless it is being resent after a failure or rejection and consequently to round-robin the messages to the consumers in the presence of multiple readers from a queue. Hence, it can be seen that a queue in this case does not exhibit true First-In, First-Out (FIFO) characteristics. Using message prioritization or implementation-specific delivery optimisations can lead queues as well to drift away from the FIFO properties, and in such cases a queue may be described as "weak-FIFO".

2.6.7 Bindings

A Binding is an AMQP entity created by the client application to defines the relationship between a message queue and an exchange, it instructs the exchange how to route message and to which queues. Applications can create and destroy as many bindings as they need to drive the flow of messages into their message queues, and when a message queue is destroyed, its bindings are also destroyed.

CHAPTER 3

State of the Art

Future ITS will see greater usage of Information and Communication Technologies (ICT) to enhance even more the efficiency, safety and comfort on the roads [19], with safety - understandingly- is being given special attention to fulfill the essential constraints of road safety applications, namely *low latency*, *high reliability* and *high throughput*.

For the time being, Short Range Communication (SRC) standards based on IEEE 802.11 standard and Cellular-V2X (C-V2X) [95] based on Device to Device (D2D) Pro-Se services (since 3GPP Release 12 [94] and enhanced for vehicular use cases in Release 14 [95]), are the competing technologies to support direct V2V broadcast communications. Even though C-V2X enables vehicles to communicate directly (in the 5.9 GHz band) in the absence of cellular network coverage and to reconnect to the network when the vehicle is back in coverage; C-V2V is still not considered as a mature alternative for vehicular safety applications as compared to the localized SRC, due to the probable authentication and resource allocation control by eNBs to use this ability (at least when being inside coverage of an eNB) [117], [118], which introduces unwanted overhead and delays. Stakeholders of the wireless and vehicle industry are still in debate with the unique advantages of each technology. For instance, no agreement have been reached yet on whether cellular technologies are ready to be used for safety critical ADAS systems due to technical, political and performance related indicators, which compete against cellular technologies use in safety applications, such as cellular network coverage, reliability, liability and security, in addition to the need for mobile operator subscriptions, and roaming agreements, as for today [119]. Moreover, in the near future, competition is expected to be even stronger with the next generation "New Radio" communications technology (5G NR-V2X) under 3GPP Release-16 in June 2020 [120] and the NGV (IEEE 802.11 bd) by December 2021 [65], each is expected to deliver more capacity, lower latency and higher speeds than its respective preceding. However, since IEEE 802.11bd and NR V2X are still under

development with no functioning deployments so far, they are both out of the scope of this chapter.

This chapter will focus at the presently deployed technologies starting by explaining the term V2X, comparing the main characteristics and limitations of major V2X technologies, that is LTE and access technologies based on the IEEE 802.11p (namely ITS-G5 in Europe and WAVE in the US); afterwards, it will explain V2X hybrid architectures and mobility management concepts; and finally lists the most remarkable work on Radio Access Technologys (RATs) interworking by communication interface selection algorithms and handover mechanisms; and in lastly will be presented, studies on hybrid vehicular communication combining LTE-V2V and the traditional SRC technologies.

3.1 Vehicle-to-Everything Communication

Vehicle-to-Everything (**V2X**) communications (figure 3.1) indicate the information interchange between a vehicle on one side and any other Intelligent Transport Systems entity on the other side. V2X applications may contain the following types:

- Vehicle-to-Vehicle (**V2V**) transports messages containing V2V application information.
- Vehicle-to-Infrastructure (**V2I**) transmits messages containing V2I application information between a vehicle and transportation system infrastructure (e.g. signs, traffic lights, RSUs or locally relevant application server).
- Vehicle-to-Network (**V2N**) supports communication with a server supporting V2N applications via private or public access networks, core networks (e.g. Internet)
- Vehicle-to-Pedestrian (**V2P**) enables communication with Vulnerable Road Users VRUs (e.g. pedestrians, cyclists)

These types of V2X applications can use "co-operative awareness" to provide more intelligent services for end-users, which means that entities, such as vehicles, roadside infrastructure, application server and pedestrians, can collect knowledge of their local environment (e.g., information received from other vehicles or sensor equipment in proximity) to process and share that knowledge in order to provide more intelligent services, such as cooperative collision warning or autonomous driving. This level of cooperation on the roads has the potential to broaden the scope of the services that an ITS offers to its users by enabling novel ITS applications for road safety, traffic efficiency, infotainment as well as vehicles manufacturers' services, each of which imposing its own technical and performance set of requirements [121]–[123].

In concordant with its initial design purpose that focuses on low latency, IEEE 802.11p is considered as the dominant V2V standard, and it has been confirmed by several tests and performance evaluations that IEEE 802.11p based RATs work more efficiently in vehicular communication networks than all the other traditional WiFi protocols of the IEEE 802.11 family, even in large scale deployment due to its ability to operate in OCB mode. However, despite all the advantages that 802.11p has in vehicular environments over other protocols of the same family, the major technical downside of SRC based access technologies is the

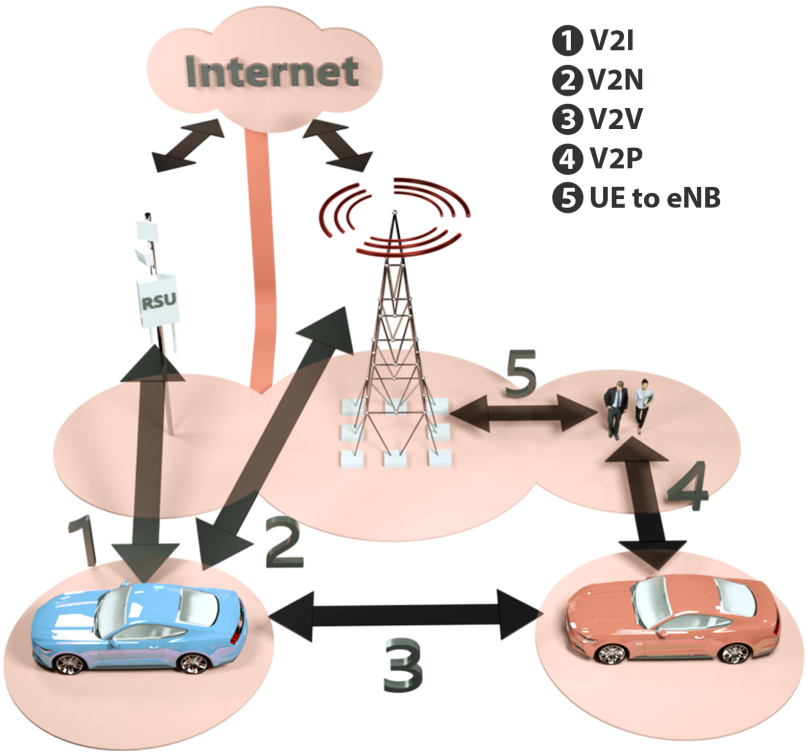


Figure 3.1: Communication in V2X environment

inherent short communication range which limits its usability in some cases that require covering areas out of SRC reach or providing a stable in-vehicle Internet access; in addition to its low data rates, large channel access delay and high collision probability in vehicle dense scenarios caused by the contention-based CSMA/CA Medium Access Control scheme used by SRC standards which results in unreliable broadcasting that strongly suffers the hidden terminal problem [124]. The problem whereby a transmitting node can fail in its attempt to transmit data because of destructive interference which is only detectable at the receiving node, not the transmitting node due to the lack of any handshaking or acknowledgment mechanisms, not to mention the regional spectrum incompatibility issues and high investment and deployment costs of all the required infrastructure (e.g. Access Points (APs), gateways) along thousands of miles of roads, with the absence of a clear sight business model which obstructs the wide adoption of SRC by governments and roads operators. IEEE 802.11ax amendment was launched in May 2014 [125] (and yet under development) mainly focusing on achieving enhanced throughput-per-area in high-density scenarios [126], [127]. Some medium access control schemes have been proposed based on distributed TDMA, contention window size optimization and other cooperative methods to improve the CSMA/CA and provide a reliable broadcast service for high-priority V2X road safety applications, but none of these protocols have been standardized so far [128]–[133].

On the contrary, cellular communication systems, namely LTE, is a widely deployed technology that benefited from the preceding 3G infrastructure, and it is able to provide higher ranges than SRC (even with SRC multihop routing) thanks to its large-cell-coverage-

range that can be reached by the eNBs resulting in longer eNB-vehicle contact time which in turn reduces the handover rate, on top of its higher data rates and suitability for communication between objects with speeds up to 200 km/h [134], but due to its centralized-control nature, LTE is unable to fully support low-latency V2V communication of safety-critical vehicular applications, which is estimated to be 5 times less than the current LTE networks end-to-end latency. Furthermore, even though modern LTE-Advanced releases support communication in D2D mode, this solution was developed for public safety services with relatively tolerant latency and reliability requirements, besides its relatively complex resource allocation principles which are often dependable on eNBs for resource control and resource selection mechanism, thus it is yet not capable of meeting the ones required in time critical vehicular scenarios. 3GPP standards already support multicast and broadcast features under the Multimedia Broadcast and Multicast Services (MBMS) and evolved MBMS (eMBMS). In broadcast mode, the broadcasting vehicle needs to send the V2X messages (including the vehicle's location information) to a V2X server, that in turn forwards them to all the vehicles in its cell which includes vehicles that are out of the zone of relevance of a specific safety message. This issue has some presented multicast solutions in 3GPP standards [117], [135] to define how to decide the V2X dissemination area, and how to transmit different V2X messages in different areas. However, these solution might be costly in terms of latency and control signaling overhead associated with the join and leave procedures of the eMBMS, which are necessary to create a multicast group [117], [136]–[142]. Table 3.1 lists the major V2X technologies along with their respective standards, and table 3.2 highlights the main features of SRC and LTE from the perspective of vehicular communications. A comparative analysis of the two technologies based on chosen common performance characteristics can be found in [143].

Technology	Region	Standard
802.11p	US	IEEE 1609.1/2/3/4, SAE J2735 and J2745/x
802.11p	Europe	ITS-G5
802.11p	Japan	ARIB STD-109
Cellular LTE	Global	3GPP TS 22.185, TS 23.285, TS 36 series
Cellular 5G	Global	3GPP TS 22.186, TS 23.501, TS 38 series

Table 3.1: Major V2X Technologies

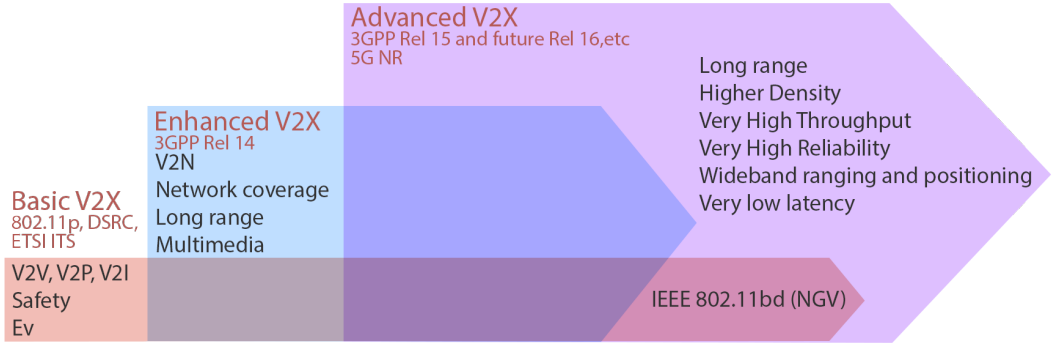


Figure 3.2: V2X Evolution

Characteristics	SRC	LTE
Maximum range	1 km	100 km
Maximum bandwidth	10 MHz	Variable up to 100 MHz
Connectivity	Intermittent	Pervasive
Capacity	Average	Very high
Frequency band	5.9 GHz	450 MHz–4.99 GHz
Peak data rates	6-27 Mbps	300-1000 Mbps
V2V connectivity	Ad-hoc	D2D (LTE-A)
V2I connectivity	Yes	Yes
Market penetration	Low	High

Table 3.2: V2X Technologies specifications

3.2 Short Range–cellular hybrid V2X communications architecture

Due to their fundamental structural differences, most of the available work that compares the performance of LTE and SRC based technologies in vehicular environments, do so based comparing a set of performance evaluation metrics, i.e. throughput, packet loss and latency. Such as the 2019 study [142], which found that the IEEE 802.11p is able to satisfy the necessities of many road safety applications with less than 25 ms end-to-end delay and 1.46 Mbps throughput at low vehicles density; and that LTE can reach higher throughput (2.58 Mbps) but cannot meet the time critical V2V latency requirements. The comprehensive survey presented in [144] that compared WiFi, DSRC, UMTS, LTE, and LTE-A technologies for usage in vehicular communication has reached similar results that LTE outperforms localized communication in terms of capacity and throughput, but it presents higher delays and that

it needs further improvements towards the device-to-device communication. That being the case, all the reviewed studies conclude that no single RAT is solely able to measure up to all ITS needs simultaneously, and for this reason, **hybrid** solutions have been proposed for vehicular networks.

At the present time, a combination of localized communications for direct short range (ad-hoc) communication; and cellular networks acting as an Internet access for various infotainment applications [145]–[147] and a vehicular data backup for supporting indirect long range V2V communication, is yet the most common approach of hybrid V2X, and it is the main focus of this literature review. Additionally, a cellular network can be used as a backbone network for disseminating control messages to support pure short range position-based ad-hoc routing protocols in situations of surroundings information scarcity (e.g. vehicle traffic flow condition) [148], [149], or even as suggested by [150] to opportunistically devote a part of the short range communication spectrum to cellular networks when cellular data traffic peaks and V2X data traffic plummets. According to the authors of [53], the scene is not going to change soon since it is not expected that either of these technologies will be able alone to support the vast diversity of V2X applications for a congested or highly dynamic traffic scenario due to each technology's own limitations. Therefore, collaboration between SRC and cellular systems is an issue of focus for many researchers and engineers in an attempt to bring positive effects into the overall performance of Intelligent Transport Systems by leveraging the advantages of both technologies.

In hybrid vehicular communication systems (SRC-LTE), a node is either static (i.e. eNB, RSU) or mobile (i.e. OBU). The possible ways of interaction between these nodes can be reached through very diverse approaches, and different criteria can be followed to build the RATs interworking model. The nodes can be conceptually arranged in a *flat* or *hierarchical* architecture:

- In a *flat* architecture, all nodes (mainly mobile nodes) are assumed to be equal in terms of their ability to use the supported RATs, and the choice of which technology to utilize in a specific case is individually made within each node. The decision can be based on the type of transmitted data (data-centric) [148], [149]; on particular performance metrics (network load, network coverage, vehicle's sensors data) which reflect the expected QoS of each technology (performance-centric) [146], [151], [152]; or on a certain algorithm that deploys both concepts [153].
- Hierarchical architectures group the network nodes into different hierarchical levels with different privileges given to the nodes in each level in terms of their accessible communication technologies. Therefore, using a certain RAT (e.g. LTE for Internet access) can only happen by passing through specific nodes within one hierarchical level. A hierarchical architecture can be; *fixed* where a node permanently belongs to a specific hierarchical level that might impose some restrictions on using a specific technology for V2X communications; or *dynamic* where nodes are not pre-classified and all nodes are initially assumed to be homogeneous (i.e. support both LTE and SRC), then the hierarchical levels are dynamically set up during network operation based on variations

in some network parameters (e.g. network topology, traffic load).

An example of a fixed hierarchical architecture is presented in [154], where a cloud-assisted scheme is proposed. Besides the distributed cloud servers tier, the hybrid architecture consists of two tiers of nodes. The high-tier nodes are the public transportation vehicles (buses) that are equipped with both cellular and short range communication interfaces to serve as Internet gateways for the low-tier nodes that can be any ordinary vehicle and are only able to use SRC. Similar framework is proposed by Ansari, Boutaleb, Sinanovic, *et al.* [155], albeit that the low-tier nodes are also assumed to support LTE and SRC; they use SRC for all types of communications by registering to a nearby high-tier node, and only fall back to use LTE if no high-tier node is reachable.
One way to attain a dynamic hierarchy is by deploying a "node clustering scheme" [156], that groups adjacent nodes into a set, named *cluster*; chooses one node to be a cluster head while the remaining ones are called cluster members. The cluster head is in charge of maintaining the cluster and managing its network resources, hence it may choose some of the cluster members as gateways for handling communication with other clusters [157], [158] and possibly to aggregate and relay information from/to the cellular network among other members of the cluster to reduce the network data traffic (e.g. video streaming) [159], [160].
While a fixed hierarchy provides a straightforward and stable architecture, it lacks adaptability and robustness to network dynamics (e.g. disconnections), and sometimes, it is not possible to implement a public-vehicle-based hierarchy outside of urban areas (where public vehicles are sparse) [154]. Contrastingly, a cluster-based dynamic hierarchy enhances such flexibility to network changes, but creating and maintaining clusters require explicit overhead messages exchange, which can notably increase in a highly dynamic vehicular network. Therefore, forming stable and long-lasting clusters is an important point to be considered when designing a cluster-based SRC–cellular network [161].

3.3 MOBILITY MANAGEMENT

Mobility management has always been a point of focus for wireless protocols, aiming to provide seamless communication to mobile nodes in different conditions. Ad-hoc networks brought more challenges to this field due to their infrastructure-less temporary nature. However, Vehicular Ad hoc Networks (VANETs) have risen the challenge level even more, because of vehicular environment's special characteristics:

- High mobility which results in recurrent nodes density variations, consequently affecting links stability (e.g. in case of clustering [161]), causing network fragmentation, topology changes and data flow disconnections [162]–[164];
- High vehicles speed (e.g. on highways) and SRC limited range might especially affect the V2I communication, since the interaction time between a moving vehicle and a fixed unit decreases as the vehicle's speed increases [165];
- Buildings and surrounding obstacles in urban areas can be related to intermittent connectivity due to shadowing and channel fading [159], [166], [167].

It is discernible that vehicles' mobility imposes technical challenges in handling V2V connections between vehicular nodes and V2I connections between vehicles and (RSUs/cellular base stations), as vehicles move in and out of the coverage areas of other vehicles, RSUs, and cellular base station [152]; signal quality degradation due to relative mobility, channel fading, or interference [152], [168]; or even when being in a coverage overlapping region, but one RAT could be able to offer better performance (e.g. higher throughput, lower cost, shorter delay) than the other [152]. Therefore, mobility management is a crucial issue in designing SRC–cellular hybrid solutions for V2X communications, typically by employing efficient **handover strategies** and **network selection schemes**.

3.3.1 Handover Strategies

As defined by the Internet Engineering Task Force (IETF) [169], a handover is the process by which an active mobile node changes its point of attachment to the network, or when such a change is attempted. There are different types of handover, classified according to different aspects involved in the handover procedure such as the scope, mobility types and performance characteristics. For instance, a handover can be either:

- *Horizontal* when a mobile node moves between points of attachment of the same type (use the same access technology) such as LTE to LTE; or *Vertical* when a mobile node moves between points of attachment of different type (use different access technologies) such as LTE to WLAN.
- *Hard* when a mobile node starts communication with the new point of attachment only after breaking the connection with the previous one (break-before-make); or *Soft* when the new point of attachment is reached before breaking the connection to the previous one (make-before-break).
- *Mobile-initiated* when the mobile node makes the initial decision to start a handover; or *Network-initiated* when the network makes the initial decision to start a handover.
- *Mobile-assisted*, *Network-assisted* or *Unassisted* based on where the handover execution information and measurements are collected.
- *Proactive* when the handover is planned and some signaling is done in advance; or *Reactive* when the handover is unplanned and no signaling is done before the mobile node moves to the new point of attachment.

In the literature, studies on hybrid vehicular communications focus on *Horizontal* and *Vertical* handovers. For example, a horizontal handover is used to transfer a vehicle's video streaming session to the internet from one RSU to another, as the vehicle moves between their coverage areas [170], [171], while it requires a vertical handover to transfer the same session between an RSU and an eNB [172]. It is noticeable that vertical handover includes handling different RATs with different frequencies, bandwidths, modulation and coding schemes, hence it is prominently more challenging to perform than the horizontal one. Moreover, an efficient vertical handover mechanism resides at the core of a successful hybrid vehicular communications

system that is capable of transferring an ongoing V2X transmission from one RAT to another while minimizing the packet loss and the handover delay.

3.3.2 Network Selection Schemes

A handover usually starts by some sort of measurements and handover-related information collection [169] that will assist in deciding whether a handover is needed, and about when/where to handover, by jointly analysing this collected information, in addition to a set of predefined thresholds and measures known as *handover triggers* within the network selection scheme, in order to make a handover decision. In short, network (protocol) selection is the process of making a handover decision based on handover triggers. These triggers can be oriented towards enhancing the user experience (user-centric), or towards improving the overall network performance (network-centric). User-centric handover triggers are often related to the user's provisioned QoE; QoS general metrics (i.e. throughput, end-to-end delay, packet loss); some application-specific requirements (e.g. video streaming and online gaming video quality); or even the possible financial cost of using a certain RAT (i.e. SRC-based services are mostly free as compared to the subscription-based cellular data plans) [151], [159], [173]. On the other hand, network-centric triggers can use load balancing methods to distribute the load between cellular and SRC networks according to each technology's own capacities [152], or for users fairness issues [152], [168]. IT is also possible to prioritize specific types of vehicular data traffic (e.g. safety-related messages) [174]; or aims for maximizing the overall network throughput to cope with the increasing user demand for bandwidth and required latency [175]. In order to make an informed handover decision, network selection schemes may use user-centric, network-centric handover triggers, or define an objective function that combines both types [168], [174].

Handover-related information can be either collected in a distributed or centralized fashion. In the former, each vehicle relies in its own information collection capabilities, such as using the Received Signal Strength Indication (RSSI) of the available networks [151], or the vehicle's sensors information (e.g. speed) [153] as measures of networks' ability to support QoS requirements; while in the centralized approach (mainly hierarchical architecture), a network entity (central server, cloud) collects, process and disseminate handover-related information among the vehicles through cellular or localized communication networks [146], [168], [176]. In like manner, network selection schemes can be *distributed* when the selection algorithm runs locally within each node, autonomously from other vehicles [153]; *centralized* when a central entity (server, cloud) executes the network selection algorithm and shares optimized results for each node [177]; or incorporate both approaches in order to make more informed decisions about the network load and fairness [152], QoS requirements for different flow classes [178], or available data rate per vehicle per access technology [168].

Local-information-based fully distributed scheme is more prone to making inaccurate handover decisions and might suffer of fairness issues, due to limited-scope and inaccurate information that some vehicles might have, causing each vehicle to ignorantly try to optimize its own experience, regardless of the total network effect. On the other hand, involving a central

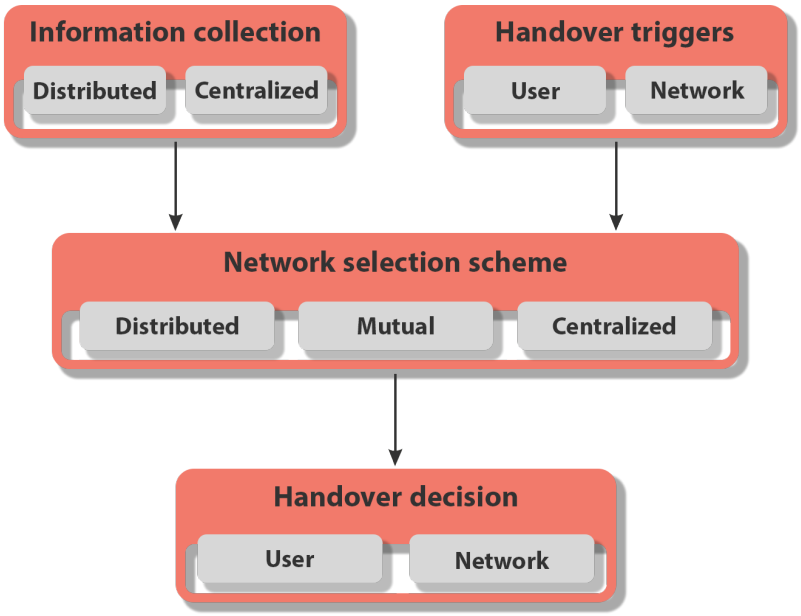


Figure 3.3: Network selection logic based on handover triggers

entity, either as a controller or as a centralized information base, has the potential to improve the overall system performance by fairly distributing the available resources, performing a proactive well-informed handover decision and pushing the prediction to the vehicle that needs it.

Although the handover information collection and the network selection can be centralized, the handover decision can still be left for the individual nodes to make [168]. This approach gives each node the freedom of decision and the robustness against disconnections from the central controller that is running the selection scheme. In case of a disconnection, a node can decide to initiate a handover based on local information (e.g. RSS from neighboring RSUs/BSs).

In [153], an intelligent interface selection scheme for the multi-interface terminal that is providing both LTE and DSRC based access is proposed. V2V communication is assumed to be possible only through DSRC which means that the OBU has to send all the V2V applications' data through the WAVE supporting interface. V2I is handled in a different way, the authors differentiate between two cases:

- when the OBU is located in the coverage area of one of the two technologies.
- when the OBU is located in a coverage interference area where both DSRC and LTE are available for transmitting data.

In the first case, the OBU just sends the data either to the RSU or to the eNB depending on the available technology at the transmission time. While in the second case the OBU can approach both the RSU and the eNB. Here is when the OBU has to choose which interface is more suitable to send the data or in other words, which interface provides better performance in each situation. To define this, the authors studied several performance evaluations of both LTE and DSRC, extracted 4 primary parameters to evaluate the interfaces performance and

named them “Communication Condition Indexes (CCIs)”, with their corresponding thresholds as shown in table 3.3, and designed a system that uses LTE by default and switches to WAVE when all these thresholds are exceeded (i.e. when the performance of DSRC is superior to the performance of LTE), thus allowing the OBU to choose the optimal interface that achieves the most possibly effective communication in each conditions.

CCI	Threshold
Distance between RSU and OBU	< 1000 m
Beacon Transmission frequency	< 10 Hz
Number of peripheral nodes	< 25 devices
Current speed	< 100 km/h

Table 3.3: CCIs and the corresponding thresholds

Gopinath, Wischhof, Ponikwar, *et al.* [97], [179] for the purpose of making an adaptive context-aware optimal communication mode selection on a per-packet basis, implemented an intermediate communication layer right under the applications layer to handle the selection process. The “Hybrid Overlay Protocol (HOP)” layer was proposed to observe the current context of the vehicle and assign ITS applications to the appropriate access technology and communication mode based on these applications’ requirements. The HOP layer communicate with the applications to determine their requirements (e.g. data payload, maximum allowed latency, dissemination range, etc) and store these requirements in form of Requirement Indicators (RIs). Simultaneously HOP layer characterizes the vehicles context by calculating several values termed Context Indicators (CIs) based on the vehicle’s local sensors (e.g. vehicle velocity) and other information received from the lower communication stack layers (e.g. data rate). The decision is made then by matching the calculated CI’s with the target RI’s.

Roman in his PhD research [180] has introduced a “Multiple Interface Scheduling System (MISS)” for packets scheduling, to be implemented in a novel intermediate shim layer located between the network layer and the MAC layer, that allows the higher layers (network, transport and application) to function independently of the used RAT, without the need to modify the fixed infrastructure, the routing protocols, nor the RAT standards. Roman’s approach focuses on the uplink; considers the vehicle as a data source that is able to simultaneously use cellular LTE and WiFi 802.11p. It relies on an adaptive scheduling algorithm -hosted in the MISS shim layer- that comprises automatic wireless interface selection, intelligent bandwidth aggregation and allocation, seamless Quality of Service QoS support in addition to context-aware packet scheduling in order to dynamically select the transmitting RAT or even jointly use both interfaces to achieve a level of reliability and throughput that cannot be reached by using a single RAT.

Although IEEE 802.11p is the de facto technology for vehicular communications, a research team from Spain and Morocco carried out a hybrid V2X communication approach based on the WiFi and 4G technologies to be used as communication mediums between vehicles [181]. The study that was published in 2018 IEEE International Conference on Vehicular

Electronics and Safety (ICVES) was aiming to provide an efficient manner of maintaining the vehicular communications (V2V, V2I, and V2P) on the road and to guarantee the quality of information to have a good reception of messages through the use of different ways to connect to the network according to the availability and pre-adjusted configurations, in addition to supporting connection through Virtual Private Network (VPN) for higher security. The proposal considers WiFi as the primary connection option and LTE as the secondary one. The system starts by scanning the active available WiFi networks, then it checks their signal quality and stores the names of the active networks and their signal strength and connects to the network of the strongest signal if it was higher than a predefined threshold, otherwise it connects to the cellular network. The system supports connecting to secured WiFi networks with authentication as well as switching between WiFi and 4G based on the comparison result of the WiFi signal strength with the manually set threshold. The proposed approach has been validated by performing real experiments with automated electric golf cart, and several WiFi access points in the university campus; to test V2I communications by measuring the Round Trip Time (RTT) for both technologies in different scenarios by changing the percentage of experiment time that the 4G is active, and the obtained results showed that the RTT values for a 4G connection are higher than for WiFi, while WiFi RTT values present more outliers and higher variance, and that combination of both can achieve the performance and the efficiency of this hybrid approach and guarantee higher connection stability in case of intermittent WiFi connection.

CHAPTER 4

System Architecture

In light of the reviewed literature and taking into consideration the available hardware and software, this dissertation proposes a hybrid vehicular communication model that uses localized communication (ITS-G5) and cellular based communication (LTE) to send and receive all types of ITS messages. in addition, it presents a shim sublayer to manage, adapt and forward traffic between the radio access interfaces and higher protocol stacks. Also, a distributed interface selection algorithm is suggested, aiming to reduce the financial costs related to using cellular networks and restrict their usage to cases where ITS-G5 is not capable of offering a reliable message delivery. The proposed model and the overall system architecture are detailed in this chapter.

4.1 Duplicated transmission via cellular interface

The first main milestone of this work is to design and build a system that sends and receives ITS packets (up to the size of Maximum Transmission Unit (MTU)) simultaneously through ITS-G5 and LTE, in compliance with the Duplicated Transmission via Cellular interface (DTC) interoperability principle (figure 4.1) defined by ETSI Technical Report (TR) 103 576-2 [182]. DTC assumes that each ITS-S is equipped with a single short-range radio technology (ITS-G5 in this case) and a a cellular radio technology (LTE) with a valid subscription to be able to participate in this scenario. Each ITS-S has to transmit and receive ITS messages via the short range radio technology and simultaneously via the cellular radio technology. Afterwards, cellular base stations or roadside ITS stations with a cellular communication technology re-transmit the received ITS messages to all ITS stations under their area of coverage via the cellular downlink. This hybrid solution elevates the message delivery probability and enables indirect interoperability through intermediary cellular network entities, since ITS-Ss with dissimilar protocols and service providers can exchange information via cellular communication

(i.e. interworking). However, it brings higher implementation complexity than using a single localized radio, to support a cellular radio technology. Latency is the key obstacle for the transmission of safety related packets via LTE, especially as cross-network communication among mobile operators would be required. This issue might be addressed by the deployment of edge computing, such as the Mobile Edge Computing (MEC) that is localized close to the source/destination of messages, therefore the messages are transmitted/received at a local level, avoiding any unnecessary messages to be sent to a centralized data base or data centre, which helps to to reduce latency and better protect users privacy.

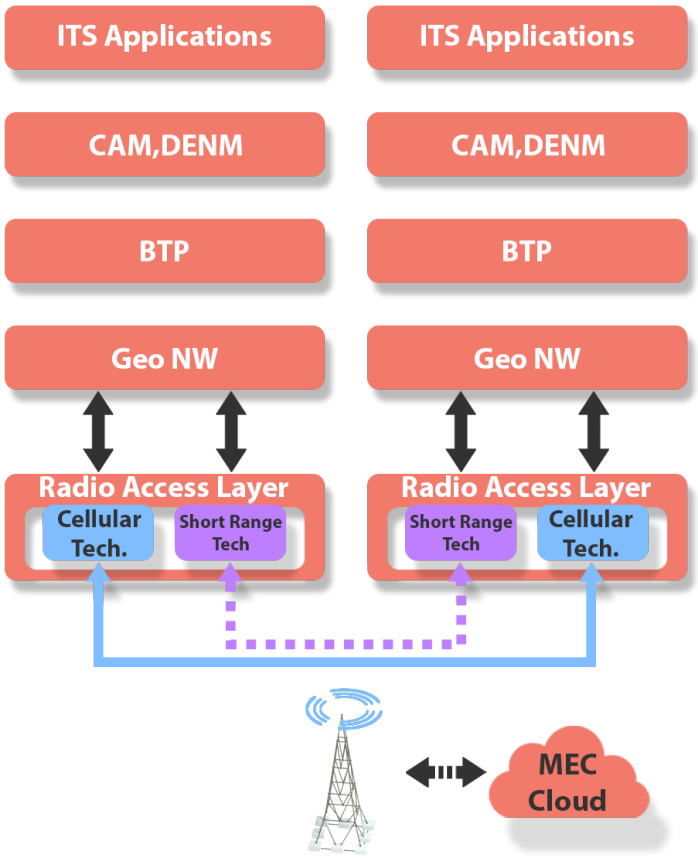


Figure 4.1: Duplicated Transmission via Cellular interface

By analysing the available hardware/software capabilities (detailed in the next chapter), taking into account the communication requirements of ITS, the DTC model shown in figure 4.1 is proposed. The short range communication channel will be established using the packet socket family (`AF_PACKET`) to allow passing the non-IP ITS packets directly between the userspace and the device driver level. Regarding the long range communication deployment over cellular networks (LTE), the implementation should support flexible, low delay exchange of (potentially) many messages from different sources to multiple destinations. These sources and destinations can be OBUs, RSUs, traffic management centers, or any user equipment that is able to use cellular networks. It should also be possible to filter messages on several

criteria, such as the type of message and the validity in time and geographical information. These requirements can be met by messaging systems or more specifically by message queuing protocols. In a typical message-queuing implementation, multiple queues are defined in a Message Queue server (broker); an application (client) can register with the MQ server and send (publish) messages and/or listen (subscribe) for messages placed into a specific queue. Depending on the MQ protocol and server software in use, many configurable options exist for the broker, clients, queues and for messages. However, a common aspect for message queuing protocols is that they separate the actual message being transmitted from the protocol and logic handling it.

Two MQ protocols were considered, namely MQTT and AMQP. The Message Queuing Telemetry Transport (MQTT) was originally designed as a simple and small-footprint solution for embedded and IoT devices with small memory and limited processing power. AMQP on the other hand, focuses on providing configurable, optimizable and flexible messaging solutions for high performance systems in a wide range of scenarios. On this account, AMQP has more features than MQTT, which allows for profound control over the messaging strategy and each entity's behavior, making it a more suitable solution for this work, even if all these features come for a price in implementation complexity [111], [183]–[185].

As observed in figure 4.2, which outlines the proposed DTC scheme, the LTE access layer is not directly reachable. In lieu, sending or receiving ITS messages using AMQP over LTE requires setting up a TCP/IP connection with the messaging broker, over which, data transfers can take place. This process of creating the connection might cause considerable delays and overhead at first, but afterwards, this connection becomes completely transparent for the exchanged messages.

At this level of the implementation, messages can be delivered in one of four uncontrollable communication modes, as illustrated in figure 4.3. Assuming that the both interfaces are locally functioning, the occurrence of any scenario only depends on external factors (e.g. cellular network coverage) and each technology's own capabilities:

- When both vehicles are out of the coverage area of cellular networks and the distance between them is less than the range of ITS-G5 (figure 4.3a), hence only direct communication is possible.
- When both vehicles are in cellular coverage and and the distance between them is larger than the range of ITS-G5, hence only indirect communication is possible (figure 4.3b).
- When only one of the communicating vehicles is in cellular coverage but they are in the range of ITS-G5, hence direct communication will handle message delivery between the two vehicles while the one located in cellular coverage area is still able to use LTE for other services (figure 4.3c).
- When both of the vehicles are in the cellular coverage and in the ITS-G5 range, duplicated messages will be received in both (figure 4.3d).

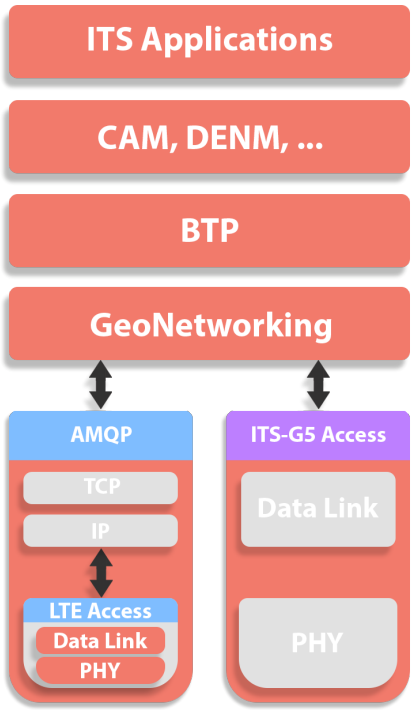
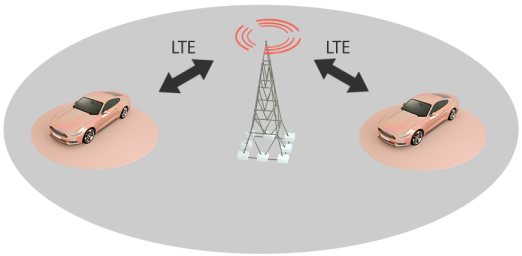
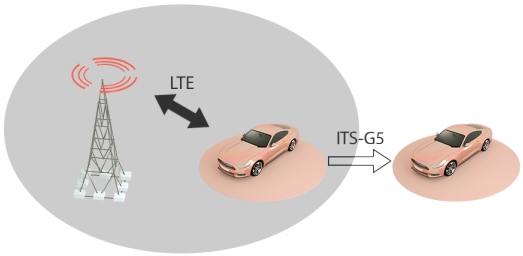


Figure 4.2: DTC implementation

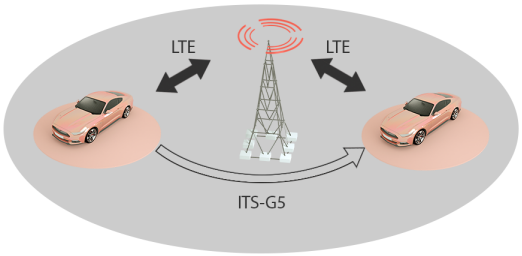
(a) Direct communication (out of cellular coverage)



(b) Indirect communication



(c) Direct communication (partial cellular coverage)



(d) Duplicated delivery

Figure 4.3: Communication modes for cellular and direct communication

4.2 Hybrid Routing Sublayer

The DTC hybrid option described in the previous section, offers a solution to boost the system's reliability and interoperability. However, the duplicated transmission of packets on both interfaces lowers the efficiency of spectrum usage, and can be related to additional delays and financial costs due to the centralized subscription-based nature of cellular communication. In addition, as for this current state, the proposed DTC approach does not take into account neither the differences between the two communication interfaces' implementations (e.g. different data types, different libraries and system calls), nor the assistance that the road infrastructure could offer to message delivery, and consequently, it requires the ITS applications to be directly responsible for handling the exchange of their packets, as well as the compatibility of their data with each access technology's communication interface specific implementation, which can complicate the applications' design and add unnecessary repeated utilization of system resources.

Having defined the limitations of the regular DTC, it is now possible to propose an architecture that aims to improve the performance of the said system by controlling the usage of cellular networks, taking into account the possible existence of a nearby RSU and the services it could provide to deliver ITS messages to areas out of reach of direct localized communication, allowing the reliable use of ITS-G5 for all ITS services. An RSU could be able to independently handle the delivery of ITS messages between vehicles in its coverage zone using ITS-G5 (figure 4.4); it could use the proprietary roadside network of the road operator to further broaden the dissemination area (figure 4.5); or in some cases, RSUs could be equipped with cellular communication which can be used to assist in message delivery. For this purpose, a new proposal was designed that introduces a cellular link controller algorithm to manage the wireless network connection via the cellular network.

The communication stack in figure 4.6 is based on the hybrid communication architectures proposed in ETSI Technical Report (TR) 103 576-2 [182]. However, this approach introduces a shim **Hybrid Routing Sublayer (HRS)** below the Networking and Transport layer of the C-ITS communication stack. This sublayer aims to isolate generating/decomposing messages, from the transmission/reception processes. Hence, it is meant to be able to handle and pass ITS packets between higher stack layers and the available radio access technologies, making the the existence of multiple Radio Access Technologies transparent to the upper layers of the protocol stack as shown in figure 4.7:

- The Message Preparation module includes functions to accept the ITS packets from the Networking and Transport layer, perform the required adaptations, add the Lower layer headers (i.e. MAC and LLC) and then pass the packets to the corresponding end module.
- The Message Handler module includes functions to accept the ITS messages received at both radio interfaces (by the receive modules), remove the lower layer headers, and pass the extracted packets to the upper layers (Networking and Transport).

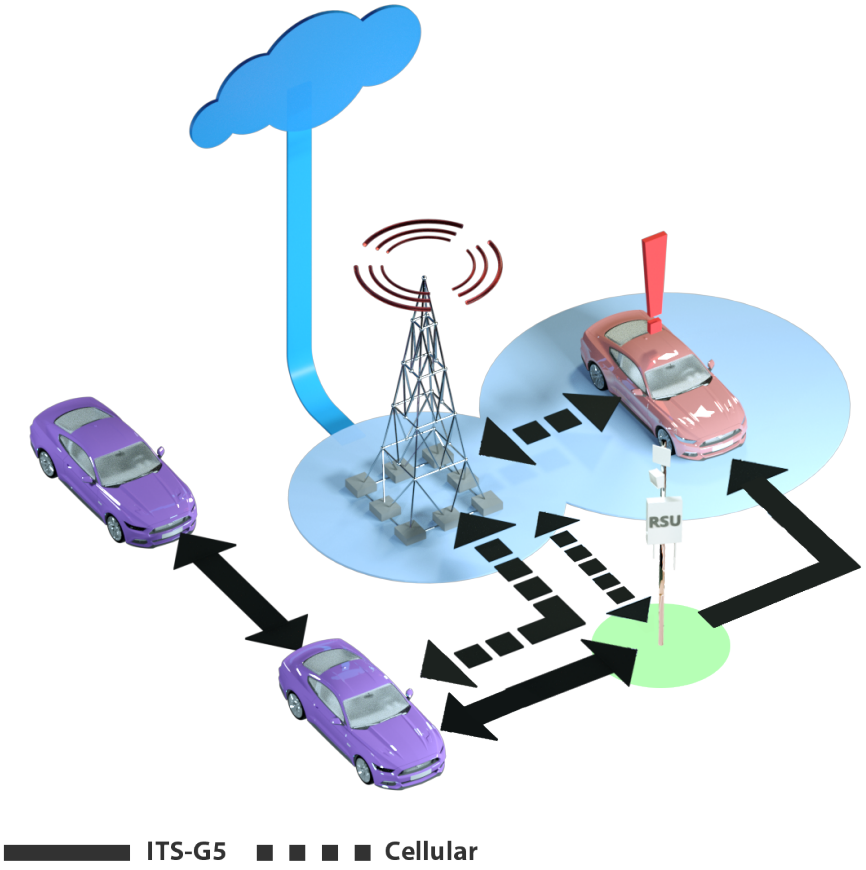


Figure 4.4: Hybrid Routing Sublayer

- The Beacon Detector module has a very specific task to detect a special type of GeoNetworking messages (will be explained in the next clause), that plays a major role in this sublayer's functionality.

In addition, for the intent of reducing the financial costs and improving the spectrum usage efficiency, the Hybrid Routing Sublayer contains the **Cellular Link Controller Module**, which is in control of the LTE connection state. That is, controlling whether any data is to be sent or received over the cellular connection; and limiting the usage of this channel to the cases when the ITS-G5 is unable to reach an RSU and therefore unable to guarantee reliable communication to broad areas. The logic behind the cellular link controller algorithm lies in evaluating the beacons reception stability at the short range communication interface (ITS-G5) and use the outcome of this evaluation to make a decision of switching the long range cellular communication ON or OFF. Beacons are periodic GeoNetworking packets that do not carry any payload, but a set of headers that hold specific ITS-S information, such as the ITS-S type (mobile or stationary) and position, in addition to other routing related information [18]. Beacons broadcasted by an RSU can be used by OBUs to detect being in the RSU's coverage area, hence determining their ability to benefit from the ITS services offered by this RSU. This implementation deploys beaconing at a fixed rate of 10 beacons per second

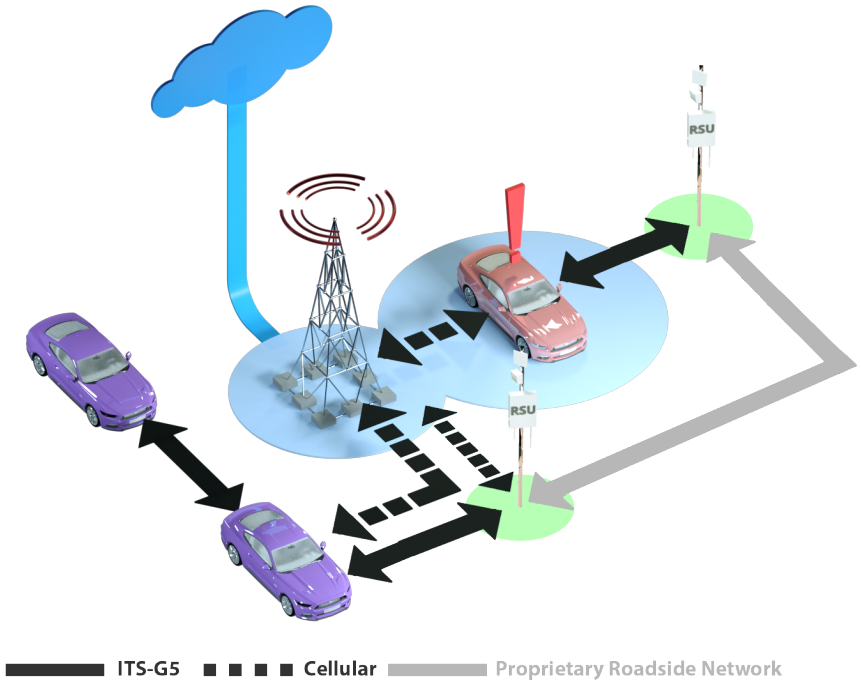


Figure 4.5: Hybrid Routing Sublayer

(every 100 ms), which means that in a typical case, one second is the time that each OBU should take to be able to detect 10 beacons. However in practice, this paradigm is almost never met due to OBU's internal delays (e.g. communication stack procedures, processor scheduling), or external factors that might cause the loss of some packets (e.g. a bus or a truck transitorily blocking the the signal between the OBU and the RSU), meaning that even in a practical stable beaconing situation, the beacon reception rate is below the theoretical one. The beaconing stability evaluation is based on comparing the expected beacons reception rate (taking into account the additional delays and the potential packet loss) and the practically observed rate at a specific case. For this purpose, a "Beacon Counter" and a "Timer" are introduced by the cellular link controller algorithm (figure 4.8). The "Beacon Counter" counts the number of beacons captured by the "Beacon Detector", and the "Timer" is used to set up the "Time out", that is the expected reception time for a certain number of beacons in a stable beaconing situation. The acceptable "Time out" is assumed to be 33% larger than the time of typical case (one second for receiving ten beacons), or in other words, a beaconing reception is considered stable, until reception rate that is 33% less than the typical value (ten beacons per second).

As a starting point for the cellular link controller module algorithm, a typical DTC scenario is assumed, where both interfaces are up and running. The module's initial state is to be waiting for new beacons arriving at the "Beacon Detector", with a zero "Beacon Counter" and deactivated timer. If no beacons by any RSU are detected, the system stays in the "Wait" state and DTC is carried on. When the first beacon is received, the "Beacon Counter" value will be incremented to one, which will, in turn, activate the "Timer" and go back to wait for

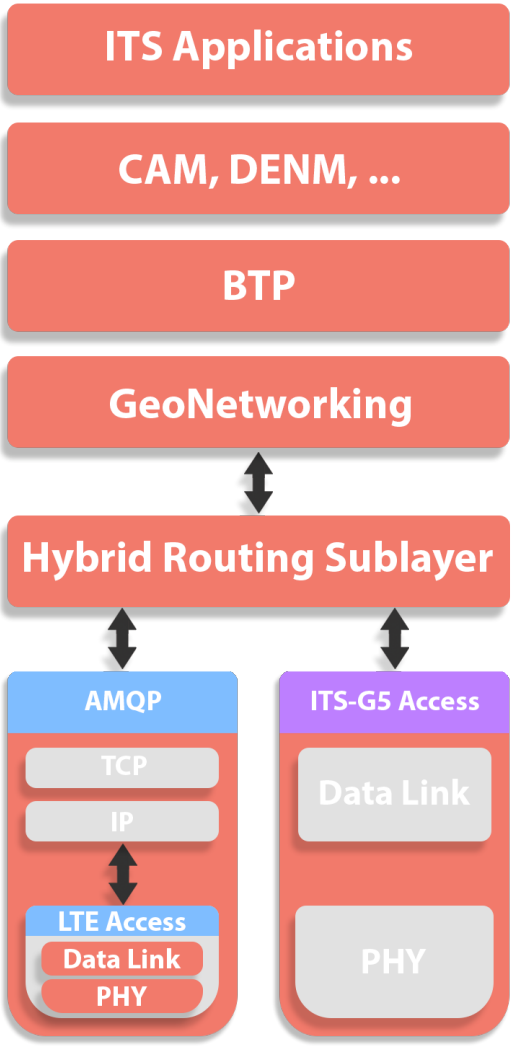


Figure 4.6: Hybrid Routing Sublayer

the "Beacon Detector" signals. As new beacons are being received, the "Beacon Detector" is increasing while the timer is running towards the "Time out" (1200 ms in this case). At this points, two scenarios can be distinguished:

- If the Beacon Detector catches nine more beacons after the first one (Beacon Counter = 10), before the Timer reaches the defined Time out; the module interprets this situation as *stable beaconing*, which means that the OBU is in the coverage area of an RSU and hence, it is possible to rely on localized communication (ITS-G5) and to temporarily disable the cellular connection (LTE). Therefore, the controller checks the state of the LTE connection:
  - In case it was already disabled (e.g. in a precedent iteration), the controller will directly reset the Beacon Counter and go back to the "Wait" state;
  - While if the LTE was found to be "ON", the controller will switch it off, then reset the Beacon Counter and continue waiting for new beacons.

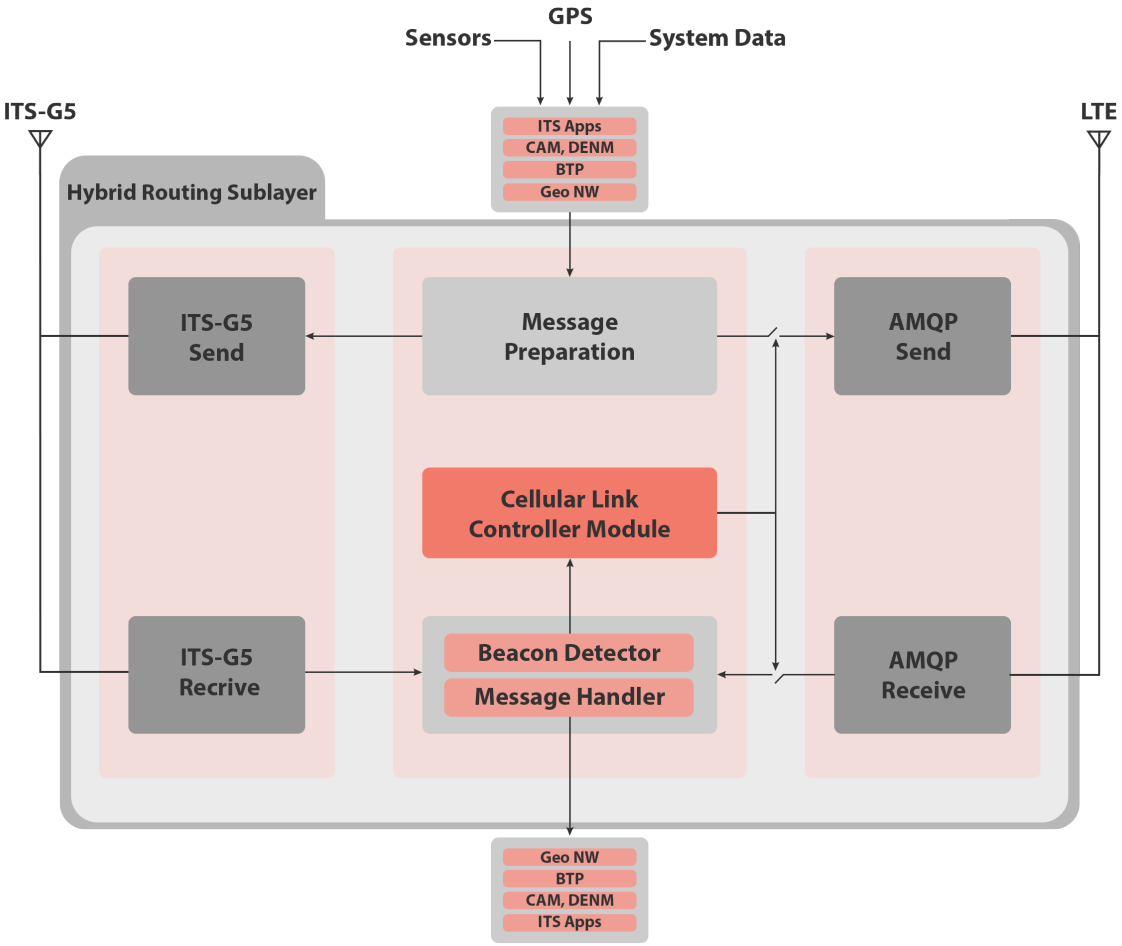


Figure 4.7: Hybrid Routing Sublayer Architecture

- If the Timer reaches the "Time out" value before the Beacon Detector receives a total of ten beacons, The module interprets this situation as *unstable beaconing* or *out of reach*, which means there is no nearby RSU that the OBU can reach and hence, the OBU cannot rely only on the short range communication to transmit or receive ITS messages but rather, ITS-G5 should be assisted by LTE. Therefore, the controller checks the state of the LTE connection:
  - In case it was already enabled (e.g. in a precedent iteration), the controller will directly reset the Beacon Counter and go back to the "Wait" state;
  - While if the LTE was found to be "OFF", the controller will switch it on, then reset the Beacon Counter and continue waiting for new beacons.

After the Beacon Counter is reset to zero, the module is put back in the "Wait" state, and a new iteration of the algorithm is started. The parameters of the algorithm can be adjusted to allow for more strict or tolerant use cases.

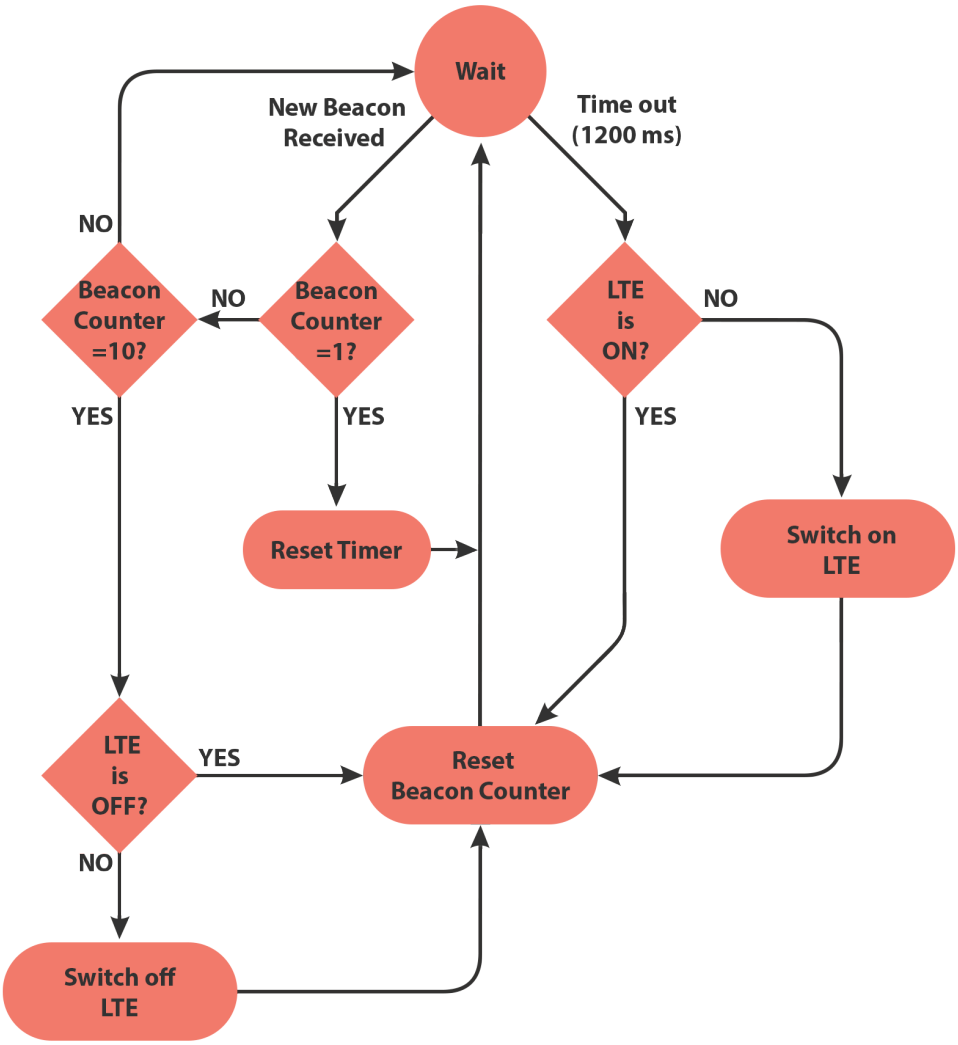


Figure 4.8: Cellular link controller algorithm

CHAPTER 5

Implementation

5.1 IT2S platform

This presented work takes advantage of a custom ETSI ITS-G5 station that was developed at Instituto de Telecomunicações - Aveiro prior to this Master's thesis work. Due to the high flexibility, configurability and low-level control of both software and hardware components, this platform was utilized to implement, test and validate the experimental work of this dissertation.

This station, named IT2S platform, is built on top of the the apu3c4 system board from PC Engines (5.2), making use of its features (table 5.1) and broadening them with a set of extensions such as GPS receiver, a WiFi module, an LTE module and with a pair of appropriate antennas for each module; all combined forming a platform that can operate either as an RSU or as an OBU. Moreover, the IT2S platform also supports Ethernet based connections which is a particularly important feature when the platform is used as an RSU, allowing the forming of back-hauling network to exchange information with other RSUs, with a gateway node or even to be accessed by the network operator for maintenance and updates.

Feature	**Description**
CPU	AMD GX-412TC SOC (4 cores - 1 GHz)
Memory	4 GB DDR3-1333 DRAM
Storage	60 GB SATA SSD
Power	12 VDC - (6 to 10 W)
I/O	DB9, 2 USB 3.0 external, 4 USB 2.0 internal
Expansion	3 mPCIe, GPIO header
Board size	152.4 x 152.4 mm

Table 5.1: PC Engines apu3c4 System Board [186]

Figure 5.1: IT2S Platform

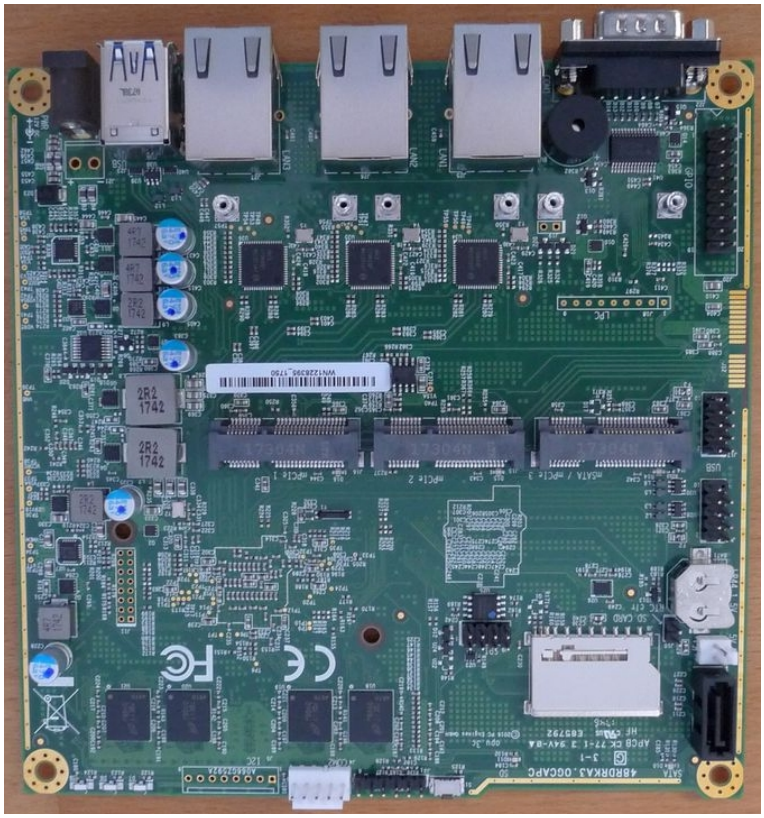


Figure 5.2: PC Engines apu3c4 System Board

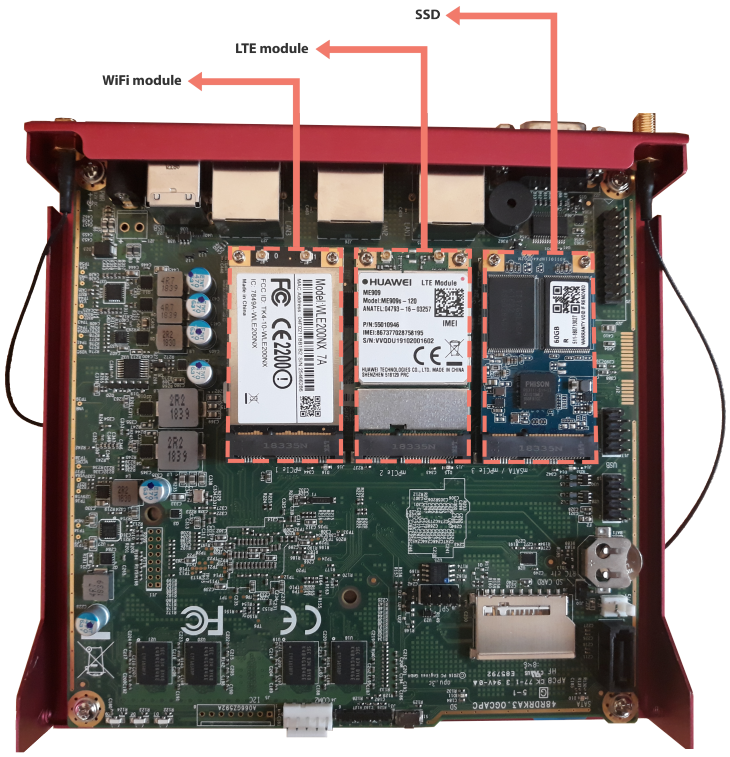


Figure 5.3: IT2S Platform Internal Components

5.2.1 Hardware Components

WiFi Module

Compex WLE200NX is a CE and FCC certified Dual Band 2x2 Multiple Input Multiple Output (MIMO) 802.11n full-size mini-PCI-express (mPCIe) Module, designed for dual band wireless access points in compliance with the the REACH and RoHS regulations, based on Atheros XB92 reference design using the Qualcomm Atheros XSPAN family AR9280 chipset as the main component, thus inheriting all its main features encapsulated in a Class 1C Electrostatic Discharge (ESD) sensitivity board [187].

Atheros AR9280 is a Single-chip, 802.11n wireless LAN client solution with integrated MAC, baseband and dual 2.4/5 GHz radios based on third-generation 802.11n compliant technology providing support for 2x2 MIMO with spatial multiplexing and Maximal Ratio Combining (MRC) to improve the signal quality of the receive end, while enabling PHY rate up to 300 Mbps and featuring Dynamic MIMO Power Save and Transmission Power Control (TPC) (IEEE 802.11h) to conserve system power. It is equipped with run-time user controllable General-Purpose Input/Output (GPIO) peripheral interfaces in addition to the ability of using an EEPROM through a memory interface as shown in its internal architecture in figure 5.4 [188].

Compex WLE200NX supports Wi-Fi Protected Access 2 (WPA2) encryption (IEEE 802.11i), 802.1X authentication, and Dynamic Frequency Selection (DFS) through two antenna chains that can be connected to the module by U.FL connectors. main technical specifications can be found in table 5.2.

Feature	**Description**
Host Interface	Mini PCIe 1.1 Standard
Frequency Bands	2.4 and 5 Ghz
Modulation Techniques	OFDM: BPSK, QPSK, 16-QAM, 64-QAM
Operating Temperature	–20°C to +70°C
Power Voltage	3.3 VDC
Antenna Connector	2x U.FL
Data Rates	Up to 300 Mbit/s (IEEE 802.11n)
TX Power	[+8, +20]dBm (2 chains)
RX Sensitivity	[-72, -94]dBm (2 chains)

Table 5.2: Compex WLE200NX Wi-Fi module features [187]

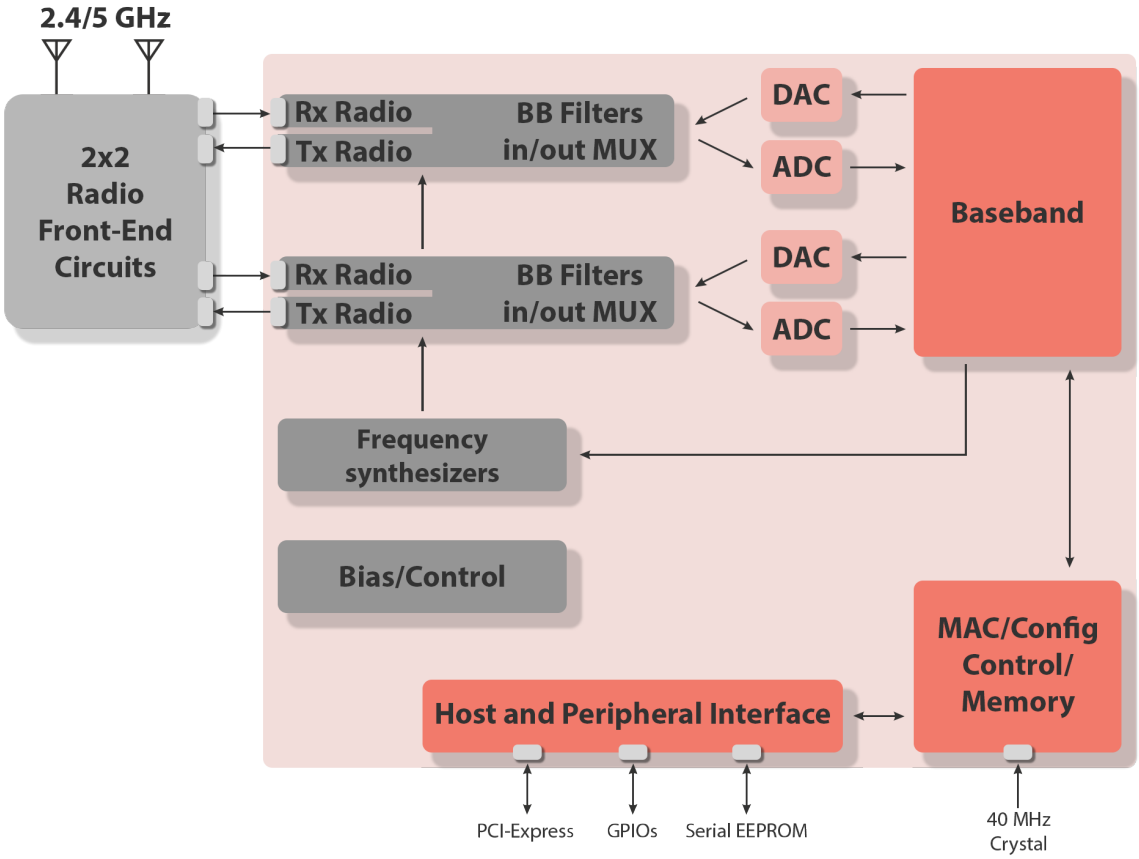


Figure 5.4: Atheros AR9280 Architecture [188]

LTE Module

Huawei ME909s-120 mPCIe LTE module is part of a family of small size and high-quality LTE modules that deploy Huawei Land Grid Array (LGA) standard design specifically developed for industrial-grade Machine to Machine (M2M) applications such as vehicle telematics, tracking, mobile payment, industrial routers, security monitoring and industrial Personal Digital Assistant (PDA). ME909s-120 module is RoHS compliant developed based on Huawei's Balong Hi6921M platform and it is the first full size mPCIe form factor LTE module based on Hi-Silicon chipset that supports the fourth category LTE User Equipment (cat4) over eight LTE bands in the EMEA region and Asia. The ME909s Mini PCIe module uses a Mini PCIe interface as its external interface as figure 5.5 shows the circuit block diagram of the ME909s Mini PCIe Adapter. The major functional units of the Mini PCIe Adapter contain the following parts:

- LGA Module;
- Control Signals;
- Antenna Connectors.

ME909s-120 is able to provide a maximum transmission speed in LTE (FDD) mode equal to 150 Mbps in the DownLink and 50 Mbps in the UpLink including enhanced features such

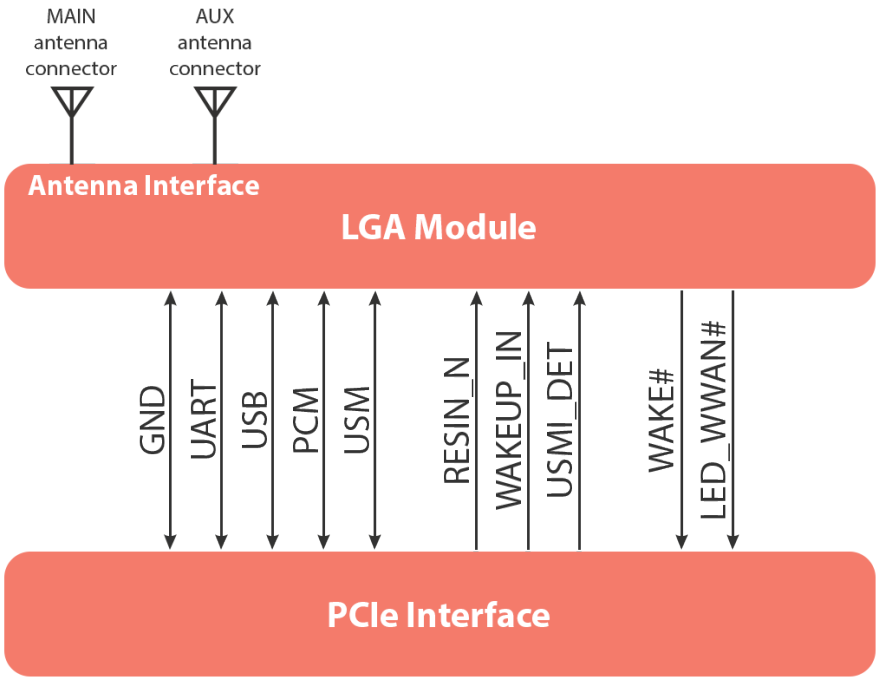


Figure 5.5: Circuit block diagram of the ME909s Mini PCIe module

as Firmware Over-The-Air (FOTA) (a Mobile Software Management (MSM) technology in which the operating firmware of a mobile device is wirelessly upgraded and updated by its manufacturer), Unstructured Supplementary Service Data (USSD) (a GSM communication technology that is used to send text between a mobile phone and an application program in the network), Pulse-Code Modulation (PCM) digital audio interface and Huawei enhanced AT commands. Huawei offers firmware support for various operating systems (Linux, Android, Windows 7, Windows 8, Windows CE, Windows Mobile) and in accordance with most 4G/LTE cellular modules manufacturers, it implements the USB Implementers Forum's MBIM interface that allows to communicate with the module over the interface. Table 5.3 gives a general function overview of the ME909s Mini PCIe module operating in the LTE mode according to Huawei official Hardware Guide [189].

Feature	Description
Host Interface	Mini PCIe 1.1 Standard
Operating Bands	B1/B2/B3/B5/B7/B8/B20, all bands with diversity
Operating Temperature	–40°C to +85°C
Power Voltage	3.2 VDC–4.2 VDC (typical value is 3.8 VDC)
Application Interfaces	USIM/PCM Voice/USB 2.0/UART
Antenna Connector	WWAN MAIN/WWAN AUX
Data Services	UL 50 Mbit/s; DL 150 Mbit/s @20M BW cat4
TX Power	[+23dBm, +32.5]dBm (3GPP TS 36.101[190] R8 Class 3)
RX Sensitivity	[-101, -111]dBm

Table 5.3: Huawei ME909s-120 LTE module features [189]

5.2.2 Software Components

To meet the requirements of supporting vehicular communication operations, allowing the ITS-G5 communication in OCB mode, an operating system kernel should allow the use of ITS-G5 channels; be equipped with a wireless stack and wireless module driver that support OCB communication in ITS-G5 channels. For this purpose, a custom Linux kernel 5.0.1 was compiled -within the *Arch Linux* distribution[1]- to support the utilization of patches and userspace tools that facilitate IEEE 802.11p-based communication even for wireless modules that does not natively support IEEE 802.11p, such as the Compex WLE200NX used in IT2S platforms which does not fully comply with IEEE 802.11p power requirements.

As mentioned in the previous section, the Compex WLE200NX used by IT2S platform has a Qualcomm Atheros AR9280 chipset, that is supported by Atheros *ath9k* FOSS wireless driver, which could be enabled while compiling the kernel, and modified to support OCB mode at the 5.9 GHz frequency band. These modifications commence by editing the Linux wireless regulatory database (wireless-regdb) [191] to enable the ability to configure the wireless device in OCB mode and tune it at 5.9 Ghz band. Wireless-regdb includes all the permitted frequencies in different countries, and for the changes in this database to be passed to the kernel, which in turn enforces them on drivers, the Central Regulatory Domain Agent (CRDA) userspace agent is triggered to update the kernel wireless core's definition of the regulatory permissions for a specific country [192]. After editing the wireless-regdb to enable OCB in Portugal; and passing these changes to the kernel using the `crda-80211p` Arch Linux package. The entries of the wireless-regdb can be printed using the `regdbdump` 4 system administration tool as in Code 5, which was executed on one IT2S platform showing that the regulation database of the platform implies activating OCB mode in Portugal but not in Canada, for example.

At this point, it is possible to use a Command-Line Interface (CLI) configuration utility for wireless devices, namely `iw`, to show and manipulate wireless devices and their configuration.

[1] Arch Linux is an independently developed, x86-64 general-purpose GNU/Linux distribution that strives to provide the latest stable versions of most software by following a rolling-release model. The default installation is a minimal base system, configured by the user to only add what is purposely required.

`iw` has the command syntax shown in Code 6, where the argument `OBJECT-TYPE` can be set to:

- `dev` for network interface configurations, hence the argument `OBJECT-NAME` will be the name of the targeted interface;
- `phy` for wireless hardware device configurations, hence the argument `OBJECT-NAME` will be the name of the targeted hardware device;
- `reg` for regulatory agent options.

The `COMMAND` argument specifies the action to perform on the object, and the set of possible actions depends on the object type. All the supported commands can be printed in command-line using `iw help`, some examples are listed in Table 5.4.
Two commands are of particular importance for the current work, namely `ocb join` and `ocb leave` that are applied on a network interface (a `dev` object) to join or leave the OCB mode network respectively (Code 8).

Setting up a network interface and a corresponding hardware to operate in OCB mode at specific channel, following certain configurations requires the use of the `iw` wireless management tool, in addition to the `ip` system administration tool which is used to show or manipulate routing, devices and tunnels, in addition to several other tasks like configuring and modifying the default and static routing, setting up tunnel over IP, listing IP addresses and property information, modifying the status of the interface, assigning, deleting and setting up IP addresses and routes [193]. `ip` has a similar syntax (Code 7) to `iw`, however it applies its `COMMANDS` on a different set of `OBJECTS` (e.g. link, address, route).
Code 9 shows the script which was created to configure the wireless interface in OCB mode at 5.9 GHz (G5-CCH) and 10 MHz bandwidth and 6 Mbps theoretical bitrate. The outcome of executing the script can be shown uising the `iw dev` command as in Code 10.

```
regdbdump <path-to-regulatory.bin>
```

Code 4: regdbdump to read regdb entries

Regarding the Huawei ME909s-120 LTE module, it is supported by the open source Linux in-kernel *cdc_mbim* driver, which stands for MBIM Communications Device Class (CDC); aims to support USB devices conforming to the "Universal Serial Bus Communications Class Subclass Specification for Mobile Broadband Interface Model" [194], the standard developed by the "USB Implementers Forum" is optimized for Mobile Broadband devices (i.e. 3G and LTE modems). The *cdc_mbim* driver creates a control channel between the kernel and the userspace, with the userspace end of this channel being a `/dev/cdc-wdmX` character device, which is used by the user to communicate with the device. Managing a cellular WWAN modem that speaks the MBIM protocol, implies that userspace MBIM management and device control applications are required, such as the `mbim-network` (Code 11) and `mbimcli` (Code 13) user commands included with the `libmbim` library [195]. `libmbim` is an open source

```
it2s-admin@pasmo-05:~$ regdbdump /usr/lib/crda/regulatory.bin

[...]

country CA: DFS-FCC
(2402.000 - 2472.000 @ 40.000), (30.00), (N/A)
(5150.000 - 5250.000 @ 80.000), (23.00), (N/A), NO-OUTDOOR, AUTO-BW
(5250.000 - 5350.000 @ 80.000), (24.00), (N/A), DFS, AUTO-BW
(5470.000 - 5600.000 @ 80.000), (24.00), (N/A), DFS
(5650.000 - 5730.000 @ 80.000), (24.00), (N/A), DFS
(5735.000 - 5835.000 @ 80.000), (30.00), (N/A)

[...]

country PT: DFS-ETSI
(2402.000 - 2482.000 @ 40.000), (20.00), (N/A)
(5170.000 - 5250.000 @ 80.000), (20.00), (N/A), AUTO-BW
(5250.000 - 5330.000 @ 80.000), (20.00), (N/A), DFS, AUTO-BW
(5490.000 - 5710.000 @ 160.000), (27.00), (N/A), DFS
(5725.000 - 5875.000 @ 80.000), (13.97), (N/A)
(5850.000 - 5925.000 @ 20.000), (30.00), (N/A), NO-CCK, OCB-ONLY
(57000.000 - 66000.000 @ 2160.000), (40.00), (N/A)

[...]
```

Code 5: regdbdump to read regdb entries

```
iw <OBJECT-TYPE> <OBJECT-NAME> <COMMAND>
```

Code 6: iw Command Syntax

```
ip <OBJECT> <COMMAND>
```

Code 7: ip Command Syntax

```
dev <devname> ocb join <freq in MHz> <5MHz|10MHz>

dev <devname> ocb leave
```

Code 8: iw OCB mode commands

```
sudo ip link set wlp4s0 up
sudo ip addr flush wlp4s0
sudo iw reg set PT
sudo ip link set wlp4s0 down
sudo iw phy phy0 set antenna 1 1
sudo iw dev wlp4s0 set type ocb
sudo ip link set wlp4s0 up
sudo iw dev wlp4s0 set bitrates legacy-5 12 sgi-5
sudo iw dev wlp4s0 ocb join 5900 10MHZ
sudo iw dev wlp4s0 set txpower fixed 2700
sudo ip add add 10.0.0.5/24 dev wlp4s0
```

Code 9: wireless interface configurations script

```
it2s-admin@pasmo-05:~$ iw dev

phy#0
    Interface wlp4s0
    ifindex 7
    wdev 0x1
    addr 04:f0:21:1b:b1:b2
    type outside context of a BSS
    channel 180 (5900 MHz), width: 10 MHz, center1: 5900 MHz
    txpower 17.00 dBm
    multicast TXQ:
        qsz-byt  qsz-pkt  flows  drops  marks  overlmt  hashcol  tx-bytes  tx-packets
        0        0        55213  489    0      0        0        17379358  79921
```

Code 10: wireless interface configurations script

Object	Command	Description
dev	<devname> info	Show interface information
	<devname> link	Current link information
	<devname> set type <type>	Set interface type
phy	list	Query device capabilities
	<phyname> channels	List all wireless devices
	<phyname> set txpower	Specify transmit power level
reg	get	Kernel's current regulatory domain information
	set	Notify kernel about current regulatory domain
	reload	Reload the kernel's regulatory database

Table 5.4: iw Commands

glib-based library that can be used to communicate with WWAN modems and devices via the MBIM protocol through two CLIs [195]:

- `mbmicli`: which offers a set of tools and options to do necessary configurations to trigger the data connection over the cellular network, make queries to oversee the module information and capabilities as well as modify the module parameters.
- `mbim-network`: which is simple network management of MBIM devices. This script requires a *profile* (configuration file) to work, which specifies the Access Point Name (APN), and optionally the authentication protocol, the APN user/password and the mbim-proxy setup option. The file that stores the profile can be found (by default) at following path `/etc/mbim-network.conf/`. The profile used for this implementation can be seen in Code 12.

Table 5.5 lists some examples of the MBIM options provided by the *libmbim* Linux library. Full documentation is available in the corresponding Linux Man Pages [196], [197].

The character device associated with the LTE module in the present implementation is `/dev/cdc-wdm0`, hence to start a network connection through the module, the `mbim-network` command should be executed as in Code 14, and Code 15 could be applied to check the connection status. The connection state can also be queried using `mbimcli` as shown in Code

16.

```
mbim-network <path/to/device> [COMMAND]
```

Code 11: mbim-network Command Syntax

```
it2s-admin@pasmo-05:~\$ cat /etc/mbim-network.conf

APN=internetm2m
PROXY=yes
```

Code 12: MBIM profile

```
mbimcli -d <path/to/device> [OPTION]
```

Code 13: mbibcli Command Syntax

```
it2s-admin@pasmo-05:~\$ sudo mbim-network /dev/cdc-wdm0 start
```

Code 14: mbim-network start

```
it2s-admin@pasmo-05:~\$ sudo mbim-network /dev/cdc-wdm0 status

Loading profile at /etc/mbim-network.conf...
    APN: internetm2m
    APN auth protocol: unset
    APN user: unset
    APN password: unset
    mbim-proxy: yes
Status: activated
```

Code 15: mbim-network status

```
sudo mbimcli -d /dev/cdc-wdm0 --query-connection-state

[/dev/cdc-wdm0] Connection status:
    Session ID: '0'
    Activation state: 'activated'
    Voice call state: 'none'
    IP type: 'ipv4'
    Context type: 'internet'
    Network error: 'unknown'
```

Code 16: mbimcli Connection State Query

MBIM CLI		Command	Description
mbim-network		start	Start connection
		stop	Stop connection
		status	Query connection status
mbimcli	Basic Connect	–query-device-caps	Query device capabilities
		–query-radio-state	Query radio state
		–query-visible-providers	Query visible providers
		–set-radio-state	Set radio state
		–change-pin	Change PIN
		–connect/–disconnect	Session Connect/Disconnect
	DSS	–dss-connect	Connect DSS session
		–dss-disconnect	Disconnect DSS session
	Application	-d, –device	Specify device path
		-p, –device-open-proxy	Request using 'mbim-proxy'
		-V, –version	Print version

Table 5.5: Linux MBIM Commands

5.3 Communication Interfaces

5.3.1 Short Range Communications using Packet Socket

In vehicular communications, ad-hoc networks that operate in OCB mode exchanging non-IP messages, that are essentially designed to be small, efficient with low overhead and tailored to the simple, single-hop broadcast over capacity constrained radio frequency channels. To exchange non-IP messages, processes (at user space) need to communicate directly with the network device driver, skipping the Transport (L4) and the Network (L3) layers, to avoid adding unwanted headers to outgoing messages or losing the incoming ones as they might be dropped by the kernel. Linux supports this ability of enabling a packet interface on device level using a special type of sockets known as *packet sockets*.

As mentioned in section 2.5, a socket is the software abstraction of the end-point of a communication channel provided to a process by the operating system. Thus, a packet socket is the end-point of an Ethernet or wireless channel, used to receive or send raw packets at the device driver (OSI Layer 2) level, allowing the user to implement protocol modules in user space on top of the physical layer.

To begin communication with an interface, a process issues a system call to create a socket using the `socket()` function (Code 17). As can be seen, the first argument is the constant `AF_PACKET` which selects the packet socket address family. The second argument (`socket_type`) is either `SOCK_RAW` that delivers raw packets to the calling process including the link-level header, and the sent frames must include a link-level header built by the user; or `SOCK_DGRAM` for "cooked" packets with the link-level header removed at receiving and added at sending, by the kernel. The third argument (`protocol`) is set to tell the data-link layer which frame types to pass to the socket out of the frames the data-link receives. `protocol` is an IEEE

802.3 protocol number as assigned by Internet Assigned Numbers Authority (IANA) [198], or a corresponding macro in network byte order as defined in `<linux/if_ether.h>` header file. All incoming packets of specified protocol type will be passed to the packet socket before they are passed to the protocols implemented in the kernel. For example, the command in Code 18, creates a low level packet interface (`AF_PACKET`), that enables exchanging raw packets including link-level header (`SOCK_RAW`), of any Ethernet type (`ETH_P_ALL`). If successful, a socket descriptor is returned and stored in the `sockfd` integer, which is used to refer to the created socket in the program. In case of error, the `socket()` system call returns the (-1) value without any error handling because packet sockets do not have the concept of a pending error [102], [199], [200]. `htons()` (Code 19) is a `C` library function to the unsigned short integer `hostshort` from host byte order to network byte order [201].

```
#include <sys/socket.h>
#include <linux/if_packet.h>
#include <net/ethernet.h> /* the L2 protocols */

packet_socket = socket(AF_PACKET, int socket_type, int protocol);
```

Code 17: Create a packet socket using socket() system call

```
int sockfd = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
```

Code 18: Create a packet socket to receive all Ethernet types

```
#include <arpa/inet.h>

uint16_t htons(uint16_t hostshort);
```

Code 19: htons C library function

By default, all packets of all interfaces of the specified protocol type are passed to a packet socket. To get packets only from a specific interface, `bind()` system call (Code 20) is used, specifying an address in a `struct sockaddr_ll` to bind the packet socket to an interface. `bind()` assigns the address specified by `addr` to the socket referred to by the file descriptor `sockfd`. `addrlen` specifies the size, in bytes, of the address structure pointed to by `addr`. On success, zero is returned. On error, (-1) is returned, and `errno` is set appropriately.

```
#include <sys/socket.h>

int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

Code 20: bind() system call

The `sockaddr_ll` structure (Code 21) is a device-independent physical-layer address that has the following fields [202]:

- `sll_family` is always `AF_PACKET`.
- `sll_protocol` is the standard Ethernet protocol type in network byte order as defined in the `<linux/if_ether.h>` include file. It defaults to the socket's protocol.
- `sll_ifindex` is the index of the interface assigned by the operating system to which the socket is associated. Value equal to (`0`) matches any interface.
- `sll_hatype` is an ARP type as defined in the `<linux/if_arp.h>` include file.
- `sll_pkttype` contains the captured packet type. This field has no meaning while sending and should be set on received packets for the user's own requirements. It can take the values:
  - `PACKET_HOST` for packets addressed to the local host;
  - `PACKET_BROADCAST` for physical-layer broadcast packets;
  - `PACKET_MULTICAST` for packets sent to a physical-layer multicast address;
  - `PACKET_OTHERHOST` for packets to some other host;
  - `PACKET_OUTGOING` for packets originating from the local host, that are looped back to a packet socket.
- `sll_halen` contains the physical-layer address length.
- `sll_addr[8]` is the physical-layer address.

```
struct sockaddr_ll {
    unsigned short sll_family;   /* Always AF_PACKET */
    unsigned short sll_protocol; /* Physical-layer protocol */
    int            sll_ifindex;  /* Interface number */
    unsigned short sll_hatype;   /* ARP hardware type */
    unsigned char  sll_pkttype;  /* Packet type */
    unsigned char  sll_halen;    /* Length of address */
    unsigned char  sll_addr[8];  /* Physical-layer address */
};
```

Code 21: sockaddr_ll Adress Structure

In order for a process to use a wireless interface, it needs to maintain a number of pieces of information, namely, the socket descriptor, socket configurations (socket options), interface index, interface hardware address (MAC address), in addition to one characteristic of the data-link layer that is the MTU. A data-link layer's MTU is the upper limit that the layer places on the size of a frame, and different data-link layers have different MTUs. [103], [200]. Packet socket options are configured using the `setsockopt()` [203] system call, and they can be retrieved using the call `getsockopt()` [204], shown in Code 22. The `level` argument should be set to `SOL_PACKET` as defined in `<linux/socket.h>`. A list of all the available options can be found in [202] and in the main sockets header `<sys/socket.h>`. In addition, all standard ioctls defined in `netdevice(7)` [205] and `socket(7)` [100] are valid on packet sockets.

Figure 5.7 outlines the implementation of the ITS-G5 communication channel using packet sockets for the sending and receiving ends, as follows:

```c
#include <sys/socket.h>

int getsockopt(int sockfd, int level, int optname, void *optval, socklen_t *optlen);
int setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen);
```

Code 22: getsockopt() and setsockopt() system calls

- **Send** starts by opening a packet socket of type `SOCK_RAW` (Code 18); then getting the index and the MAC address (to include it in the Ethernet header) of the interface to send on, using the `SIOCGIFINDEX` and `SIOCGIFHWADDR` request codes of the `ioctl()` system call [206] respectively. Thereafter, it prepares the `sockaddr_ll` structures, binds the socket to the interface using the `bind()` system call [207] (Code 20) and transmits the message using the `sendto()` system call [208] (Code 23), that is, the message of length `len` that exists in `buf` is transmitted through the socket defined by the socket descriptor `sockfd` to the address stored in the socket address structure `dest_addr` of length `addrlen`. For this implementation, the `flags` argument is set to (`0`). On success, this call returns the number of bytes sent, while on error, (-1) is returned, and `errno` is set appropriately [208].

```c
#include <sys/types.h>
#include <sys/socket.h>

ssize_t sendto(int sockfd, const void *buf, size_t len, int flags,
               const struct sockaddr *dest_addr, socklen_t addrlen);
```

Code 23: sendto() system call

- **Receive** also starts by opening a packet socket of type `SOCK_RAW` (Code 18), then getting the index of the interface to receive from, using the `SIOCGIFINDEX` request code of the `ioctl()` system call [206]; followed by preparing the buffers and structures to receive the packets. Afterwards, it binds the socket to the interface using the `bind()` system call [207] (Code 20), and starts receiving packets using the `recvfrom()` system call [209] (Code 24), that places the received message into the buffer `buf` of predifined size `len`. For this implementation, the `flags` argument is set to (`0`); the `src_addr` and `addrlen` are set to `NULL`.

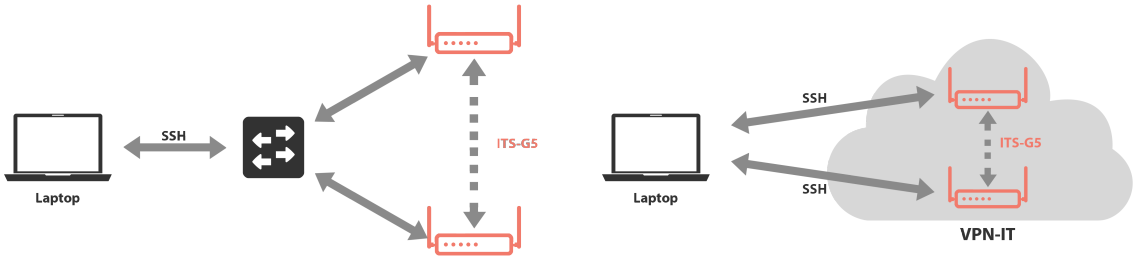
  `recvfrom()`, on success returns the number of bytes received, or (-1) if an error occurred, and `errno` is set to indicate the error [209].

```c
#include <sys/types.h>
#include <sys/socket.h>

ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags,
                 struct sockaddr *src_addr, socklen_t *addrlen);
```

Code 24: recvfrom() system call

Deployment and development of this communication channel took place on IT2S platforms. Remote logging and commands execution were conducted over an SSH connection, as shown in figure 5.6.



(a) In the lab **(b)** Outside the lab

Figure 5.6: Accessing IT2S platforms to deploy ITS-G5 communication channel

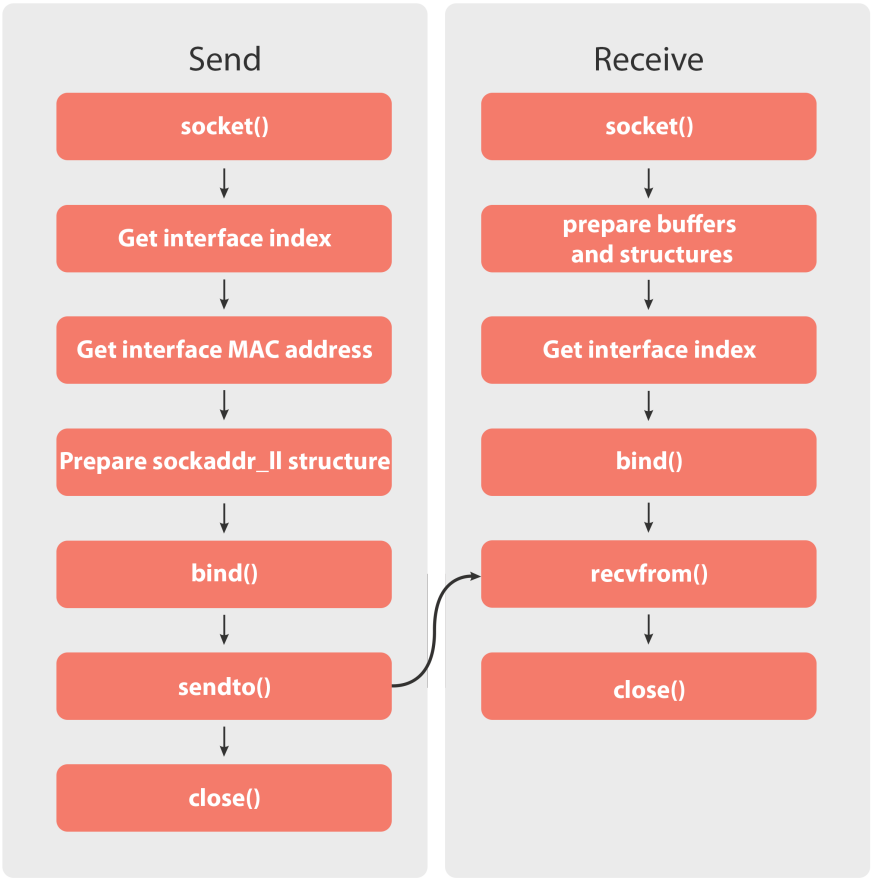


Figure 5.7: Localized communication using AF_PACKET raw socket

In order to automate and facilitate the processes at the raw packet sockets, the library `raw_sock` was created as part of the Hybrid Routing Sublayer. This library defines a set of data types, macros and functions (Code 25) that can be used to manage packet sockets;

prepare, check, encapsulate and retrieve MAC layer headers and packets, including byte ordering and memory management.

```
macaddr_t prepareMacAddrT();
unsigned int macAddrTypeGet(macaddr_t mac);
void display_packet(const char *text,byte_t *packet,unsigned int len);
void display_packetc(const char *text,byte_t *packet,unsigned int len);
void etherheadPopulateB(struct ether_header *etherHeader, macaddr_t mac,
                        ethertype_t type);
size_t etherEncapsulate(byte_t *packet,struct ether_header *header,byte_t *sdu,
                        size_t sdusize);
byte_t *getPacketPointers(byte_t *pktbuf,struct ether_header **etherHeader);
void getSrcMAC(struct ether_header *etherHeader, macaddr_t macsrc);
uint64_t hton64 (uint64_t hostu64);
uint64_t ntoh64 (uint64_t netu64);
void freeMacAddrT(macaddr_t mac);
```

Code 25: raw_sock library functions

5.3.2 LTE using AMQP (RabbitMQ)

Advanced Message Queuing Protocol is an open standard protocol designed to support messaging over middle-ware, by creating a functional interoperability between the client and the messaging middle-ware.
RabbitMQ is an open source message broker queuing server written in *Erlang*, that lets disparate applications share data via a common protocol, or to simply queue jobs for processing by distributed workers. RabbitMQ is currently under the wing of Pivotal Software. It is based around the *AMQP 0-9-1* open protocol, with official client libraries in C, Java, .NET, Erlang, as well as libraries for most other popular programming languages [210], [211]. RabbitMQ is the most widely used implementation of AMQP 0-9-1 due to many features and benefits, among which [210], [212]:

- *Open source* under the Mozilla Public License, which allows engineers and developers to contribute their own enhancements and add-ons. However, it still leverages the strength of Pivotal as a commercial and maturation supporter.
- *Reliable* as it offers a variety of features to let the user trade off performance with reliability, including persistence, delivery acknowledgements, publisher confirms and high availability.
- *Platform and vendor neutral* is an inherited feature of the platform and vendor-neutral AMQP, which makes possible the existence of clients libraries for a wide variety of programming languages on all major computer platforms, allowing all of them to share data across operating systems and environments.
- *Lightweight*, requiring less than 40 MB of RAM to run the core RabbitMQ application along with some additional plugins.
- *Client libraries for most modern languages*, making it a compelling broker to program for, since there is no language or vendor restrictions on developing applications to talk

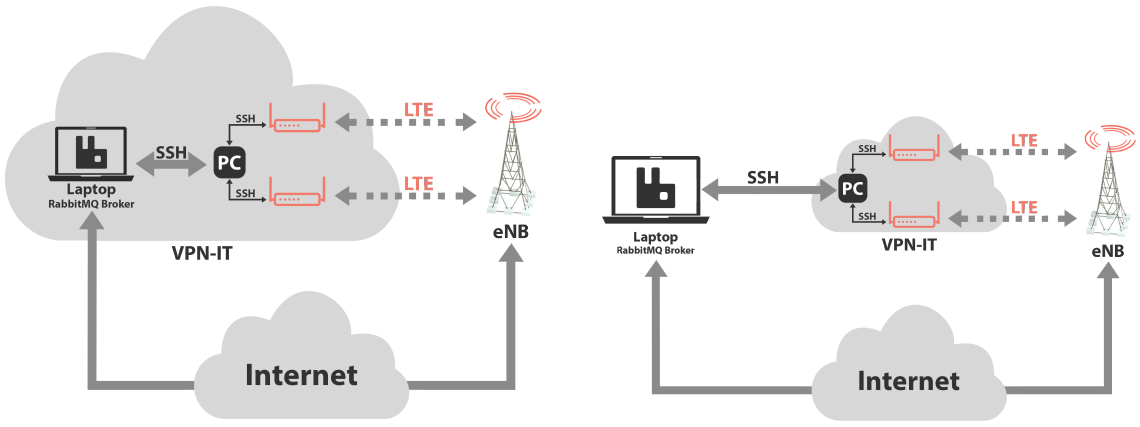
to RabbitMQ, which allowed the RabbitMQ community to create numerous clients, adaptors and tools in many programming languages [213].

- *Flexible Routing* as it features several built-in exchange types for typical messages routing logic.
- *Third-party plugins*, RabbitMQ ships with a variety of plugins extending it in different ways, but still provides a flexible plugin system that allows developers to tailor plugins for their own needs.
- *Multiple security layers* for authentication, authorisation, access control.

This presented work utilizes on the official RabbitMQ broker package for *Arch Linux* (rabbitmq 3.8.4-1) [214]; and the officially supported RabbitMQ C-language client library written by Alan Antonuk, available at [215].
Figure 5.9 outlines the implementation of the LTE communication channel using RabbitMQ for the producer and the consumer ends as follows:

- **Producer** starts by opening a TCP socket *connection* to the broker, followed by logging-in to the proper Virtual Host (vhost) and opening a *channel* to it. At this moment, the *Producer* is able to publish its messages to the broker on a specific *exchange* using a predefined *routing key* that is used by the *Consumer* to consume the messages.
- **Consumer** also starts by opening a TCP socket *connection* to the broker, followed by logging in to the proper Virtual Host (vhost) and opening a *channel* to it. Thereafter, the consumer declares a *queue* and binds it to an *exchange* by a *binding key* that defines which messages to be routed to the queue (the consumer) from the exchange. That is the case, the consumer is able to start receiving messages from the broker.

Deployment and development of this communication channel took place on IT2S platforms. Remote logging and commands execution were conducted over an SSH connection, as shown in figure 5.8.



(a) In the lab **(b)** Outside the lab

Figure 5.8: Accessing IT2S platforms to deploy LTE (AMQP) communication channel

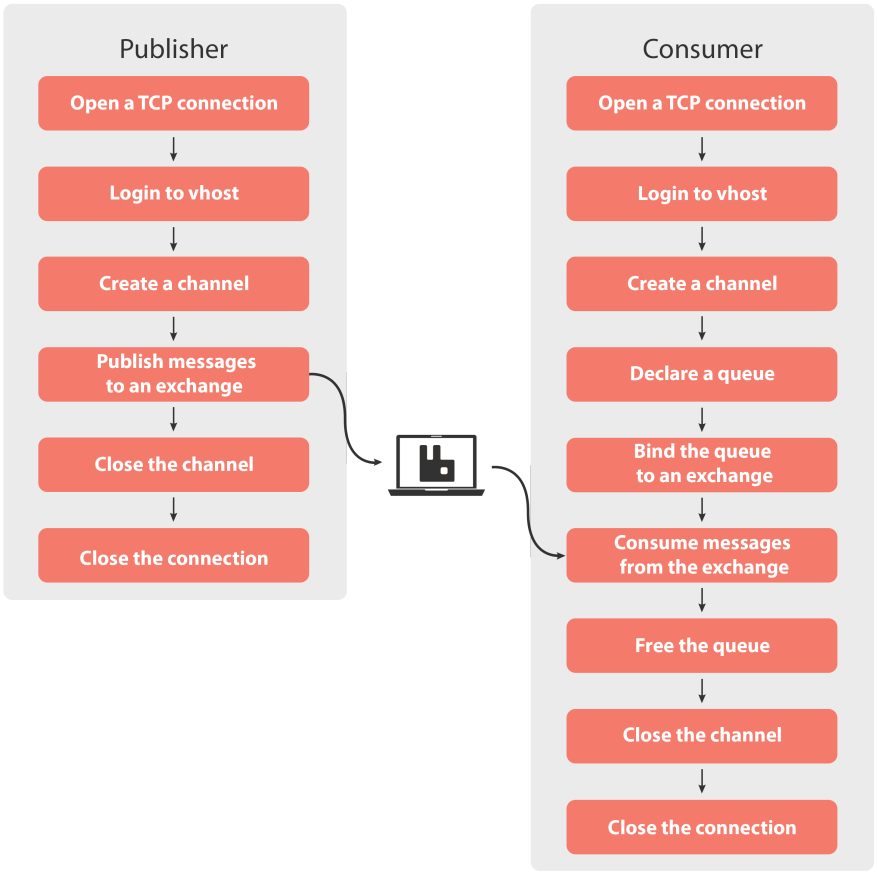


Figure 5.9: Long range communication using RabbitMQ

5.4 Cellular Link Controller Module

As mentioned in section 4.2, the Cellular Link Controller algorithm's initial state is a typical Duplicated Transmission via Cellular interface (dtc) scenario, where sending and receiving take place simultaneously on both, the ITS-G5 and the LTE interfaces. Practically, on Unix operating systems, this type of parallelism is achievable using *threads*. A thread, from a programming standpoint, can be defined as an independent stream of instructions that can be scheduled to run as such by the operating system, independently from its main program [216]. When compared to the cost of creating and managing a process (using `fork()` subroutine), a thread can be created with much less operating system overhead, and managing threads requires fewer system resources than managing processes [217]. Historically, hardware vendors have implemented their own proprietary versions of threads, but these implementations differed substantially from each other making it difficult for programmers to develop portable threaded applications. Hence, in order to take full advantage of the capabilities provided by threads, a standardized API was required. For UNIX systems, this interface has been specified in 1995, by the IEEE POSIX 1003.1c standard [218]. Implementations adhering to this standard are referred to as POSIX threads, or *Pthreads*. The POSIX standard has continued to evolve and

undergo revisions, including the Pthreads specification [219], [220]. Pthreads are defined as a set of `C` language programming types and procedure calls, implemented with a `pthread.h` header/include file and a thread library [218]. The subroutines comprising the Pthreads API can be informally grouped into three major groups [221], [222]:

- Thread management: comprises routines that work directly on threads (`pthread_t` data type) (e.g. creating, detaching, joining, etc.), and thread attributes (e.g. joinable. scheduling, etc.). Routines of this group are defined by the prefix `pthread_`
- Mutexes: routines that deal with threads synchronization, by controlling the main synchronization object in the pthreads library, which is called a "mutex" (`pthread_mutex_t` data type) Mutex is an abbreviation for "mutual exclusion" and it functions as a lock that limits the access to a specific shared variable or memory to one thread at a time. Mutex functions provide creating, destroying, locking and unlocking mutexes, in addition to mutex attribute functions that set or modify attributes associated with mutexes. Routines of this group are defined by the prefix `pthread_mutex_`
- Condition variables: address communications between threads that share a mutex, based on programmer specified conditions that are defined by variables of type `pthread_cond_t` (i.e. condition variables). This group includes functions to create, destroy, wait and send signals between condition variables based upon specified variable values, as well as, functions to set/query condition variable attributes. Routines of this group are defined by the prefix `pthread_cond_`

Routine	**Description**
pthread_create	Create a thread
pthread_join	Wait for thread termination
pthread_cancel	Cancel execution of a thread
pthread_mutex_init	Initialize a mutex object
pthread_mutex_lock	Lock a mutex object (blocking routine)
pthread_mutex_trylock	Try once to lock a mutex object (non-blocking)
pthread_mutex_unlock	Release the mutex object
pthread_cond_init	Initialize a condition variable
pthread_cond_wait	Block the thread on a condition variable
pthread_cond_signal	Unblock threads waiting on a condition variable

Table 5.6: Pthread Library Routines

Hence, implementing a DTC mode requires creating the four following threads that will run concurrently and independently of each other:

- `th_g5_send`: to activate sending packets over the ITS-G5 interface.
- `th_g5_rcv`: to activate receiving packets over the ITS-G5 interface.
- `th_lte_send`: to activate sending packets over the LTE interface.
- `th_lte_rcv`: to activate receiving packets over the LTE interface.

Implementing the Cellular Link Controller Module introduced in section 4.2 and depicted in figure 4.7, was made feasible by forcing the AMQP publisher and consumer to lock a mutex every time they publish/consume, and release (unlock) it right after the publishing/consuming is done. This way, it is possible to block the AMQP publisher/consumer threads according to the output of the controller algorithm.

In addition to above-mentioned sending/receiving threads, a "Timer" thread is created, while running the "Beacon counter" in the main thread. To achieve the required algorithm behavior, synchronization and interaction between all the threads were enforced using the `pthread_mutex_` and `pthread_cond_` routines shown in table 5.6.

CHAPTER 6

Tests and Results

6.1 Evaluation criteria

As stated earlier, traffic fatalities is one of the main causes of deaths worldwide and vehicular communications has emerged and been approved as a promising technology to improve the safety of drivers, passengers and pedestrians on the road. However, high mobility and unpredictable link conditions in transportation systems still impose a major challenge to fully exploit the potential advantages of these technologies into building more reliable and efficient ITS. In vehicular communications, time constraints have a very high impact on the reliability and robustness of safety related ITS applications, and if these constraints are not met, disastrous consequences may occur, possibly causing human, economic and environmental losses. For instance, in case of an accident, a warning message with sufficient time in advance has to be delivered to all the vehicles approaching the location of the hazard, in order for them to take appropriate measures avoiding a possible chain collision. Most of the Co-operative road safety applications (e.g. slow vehicle warning, wrong way driving warning, lane change assistance) require the latency time not to exceed 100 ms, while Some dynamic vehicle warnings are more demanding, such as the "pre-crash sensing warning" which impose a maximum latency time of 50 ms. Traffic Efficiency are generally more time tolerant, and the maximum allowable latency time can reach 500 ms is some applications, like the "intersection management" and the "limited access warning" (e.g. road work) applications [19].

Therefore, the design of ITS in general and specifically vehicular communication systems should take into account that strong time-critical constraints must be respected in this type of scenarios. Beyond that, this type of safety-critical applications must exhibit a high probability of providing continuous correct service, in order to guarantee that ITS activities are performed within firm bounds.

For the aforesaid reasons, the evaluation of the previously described implementation will be performed on the basis of simultaneously measuring the performance of both RATs in different communication scenarios and under different network loads using three performance metrics as follows:

- **Mean end-to-end delay**, computed for each test case as the average of end-to-end delays of all the received packets.
- **Throughput**, defined for each test as the ratio between the sum of the packet sizes received at the destination and the total receive time.
- **Packet Delivery Ratio PDR**, computed as the ratio between the number of received packets and the number of transmitted packets during a test time.

Each test consists of transmitting 10000 sequenced messages that are timestamped and counted at send and receive, in addition to tracking the execution time of the sending and receiving programs, then use the collected information to calculate the each packet's end-to-end delay, the total throughput, packet loss and the Packet Delivery Ratio (PDR). Testes were conducted for packet sizes equal to 50, 100, 200, 400, 600, 800, 1000, 1200, 1400, 1514 Byte, and for each packet size, packet rates equal to 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, and 10000 packetsec were included, resulting in 370 test for each RAT and a total of 740 tests evaluating the impact of packet size and transmission rate on the abovementioned metrics.

For the purpose of this test, the library `time_hdr` was created as addition to the Hybrid Routing Sublayer. The header file `time_hdr.h` defines; the `stampHdr` structure (figure 6.1) that holds all the test-related data; in addition to a number of functions (Code 26) to set and get the values of the structure's members, as well as functions to encapsulate a payload packet and to retrieve the packet at the receive end. The `stampHdr` consists of:

- `uint64_t sec`: 64-bit seconds timestamp, it stores the seconds of the current packet timestamp.
- `uint64_t nsec`: 64-bit microseconds timestamp, it stores the nanoseconds of the current packet timestamp.
- `uint32_t seq`: 32-bit Sequence field, it is used to store cyclically increasing sequence numbers, that can be used to identify lost packets and to associate timestamps at the receive end.
- `uint16_t len`: 16-bit Payload length, it stores the payload length, up to 65535 B.
- `uint8_t reserved`: 8-bit Reserved field, always set to `_0xAA_`.

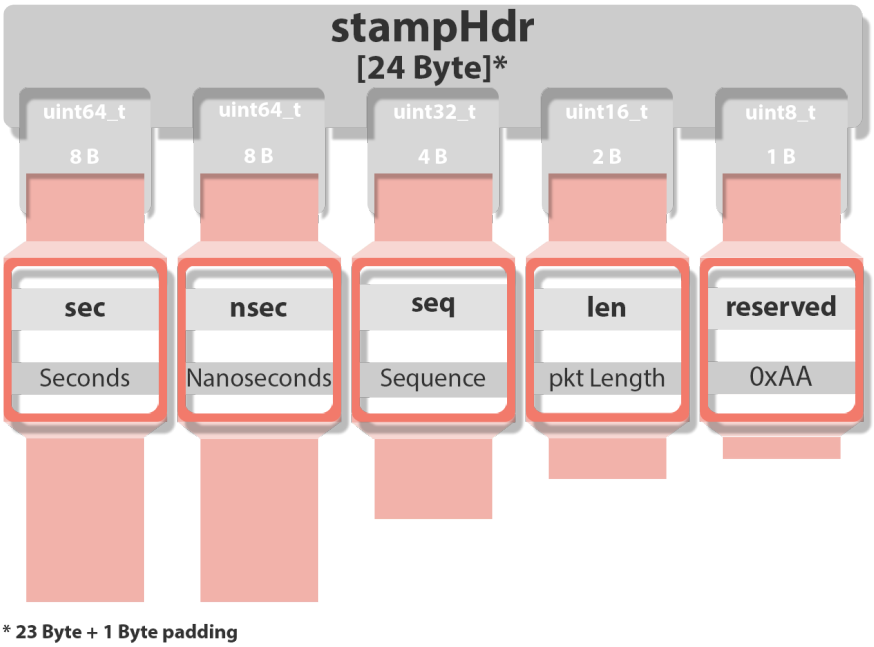


Figure 6.1: stampHdr Data Structure

```
void stampHeadPopulate(struct stamphdr *stampHeader, unsigned int seq);
void stampHeadSetTimestamp(struct stamphdr *stampHeader, struct timespec *tStampPtr);
void stampEncapsulate(byte_t *packet, struct stamphdr *stampHeader, byte_t *data,
                      size_t payloadsize);
void stampHeadIncreaseSeq(struct stamphdr *inpacket_headerptr);
void stampHeadGetData(byte_t *stampPacket, unsigned int *seq, unsigned short *len,
                      struct timespec *timestamp, byte_t *payload);
byte_t *stampGetPacketPointers(byte_t *pktbuf,struct stamphdr **stampHeader);
```

Code 26: time_hdr library functions

Concerning time analysis, Timestamps can be generated at different layers of the network stack:

- **Software Timestamping** is when packets are stamped at user space (application layer) using the Operating System (OS) time that is stored in memory. Practically, accessing memory/system time using normal kernel routines results in jitter which varies between systems and timestamped packet still need to pass through the lower layers of the network to reach physical channel, which also adds some inaccuracy.
- **Hardware Timestamping** is when packets are timestamped at MAC or PHY layer depending on the hardware type. Hardware-based timestamping effectively reduces the jitter caused at the OS level and reduces the error resulted by network layers.

Recent versions of glibc and the Linux kernel support various types of settable and nonsettable clocks that differs in precision, scope and robustness. Different types of clocks are

recognized by their `clockid` [223], [224]:

- `CLOCK_REALTIME` is a settable system-wide clock that measures real (i.e. wall-clock) time. Its time represents seconds and nanoseconds since the Epoch (00:00 1 January, 1970 UTC). It is affected by discontinuous jumps in the system time and also by the incremental adjustments performed by clock synchronization protocols (e.g. NTP, PTP).
- `CLOCK_MONOTONIC` is a nonsettable system-wide clock that represents monotonic time since some point in the past with nanosecond resolution. It is not affected by discontinuous jumps in the system time , but is affected by the incremental adjustments performed by clock synchronization protocols.
- `CLOCK_MONOTONIC_RAW` Similar to `CLOCK_MONOTONIC`, but provides access to a raw hardware-based time that is not subject to adjustments performed by clock synchronization protocols.
- `CLOCK_PROCESS_CPUTIME_ID` is a nonsettable clock that measures CPU time (CPU cycles) consumed by a process (including all the threads).
- `CLOCK_THREAD_CPUTIME_ID` is a nonsettable clock that measures CPU time consumed by a single thread.

Linux provides through the `<time.h>` header file, the ability to use a set of functions to get and manipulate date and time information, among which is the `clock_gettime()` function that is used to retrieve the time value used by the clock which is specified by argument `clockid` and stores it in the `struct timespec` pointed to by the argument `tp`. Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global variable `errno` is set to indicate the error [223].

```
int clock_gettime(clockid_t clockid, struct timespec *tp);
```

Code 27: clock_gettime()

```
struct timespec {
    time_t   tv_sec;        /* seconds */
    long     tv_nsec;       /* nanoseconds */
   };
```

Code 28: struct timespec

6.2 Test setup

The operation of the implemented system was successfully evaluated in a laboratory environment by utilizing the experimental setup depicted in figure 6.2. The *it2s* platforms in the lab are normally connected to the internal lab network -and consequently to Internet- via

Ethernet. However, the laptop which was used as a RabbitMQ broker was in some cases also connected to the same network, which would cause the packets to be routed from the RabbitMQ clients (running on the it2s platforms) to the Broker through the lab internal network without accessing the internet. In order to change this behavior and to force the platforms to use the LTE interface as a Default Gateway to access the Internet, modifications to the platforms routing tables were applied.

In an actual deployment, a publicly routable RabbitMQ broker is expected, and in this lab setup, a similar communication scheme is created by running an instance of *OpenVPN* daemon [225] on each of the platforms and on the laptop to connect to the Telecommunication Institute VPN (**VPN-IT**), allowing all of their interfaces (including LTE) to join the same private network and hence, to be discoverable and routable by each other through the Internet.

This test setup deploys the Secure Shell (SSH) protocol by using an SSH client program for remote logging and executing commands on the *it2s* platforms. The Personal Computer (PC) -which exists in the lab and is connected to the lab internal network- is used as a proxy between the laptop and the platforms so as to avoid forming a direct route between the laptop and the platforms when the laptop is physically in the lab; and to avert using the platforms' lte interfaces for transmitting *ssh* packets, but instead, devote them for RabbitMQ messages.

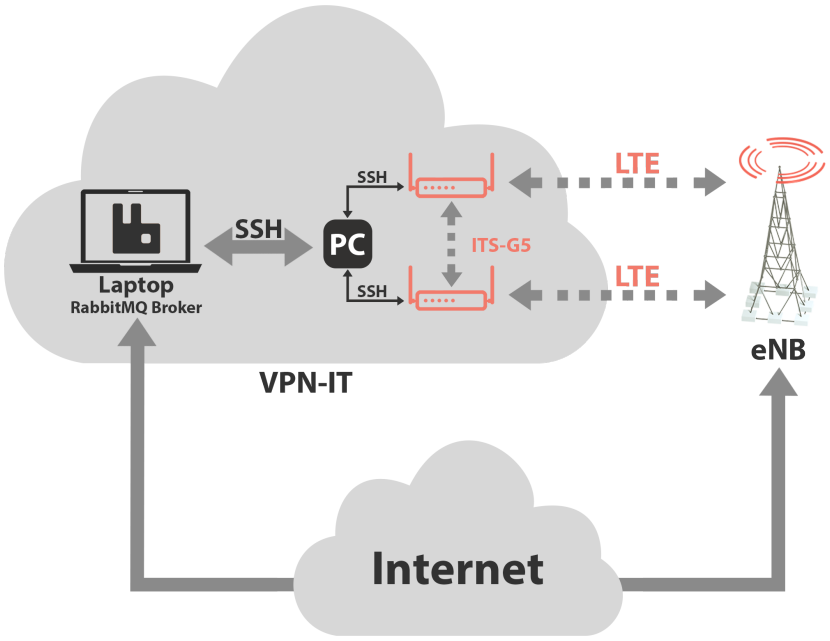


Figure 6.2: Test Setup

6.3 Synchronization

Continuous transformation of computer systems from large-scale isolated units to application specific distributed units is rising the challenge of time synchronization due to their associated

and time-critical actions. These kinds of transformation can clearly be seen in many industries like, automation, distributed measurement systems and power distribution systems. Several solutions like GPS and Network Time Protocol (NTP) emerged to overcome synchronization challenges. As the large-scale distributed systems are becoming more complex and modular, where each distributed module/node is communicating via standardized communication medium, the demand for precision and accuracy of time synchronization is increasing, while existing solutions are either limited in term of synchronization accuracy or requires specialized additional hardware and operational support to ensure the common time in distributed nodes. On the other hand, installation of additions time keeping networks and devices makes the system more complex and increases the troubleshooting and maintenance problems.

IEEE 1588 [226] Precision Time Protocol (PTP) developed to overcome these challenges, it provides a hierarchical (master-slave) self-organizing time synchronization protocol while utilizing the existing system for timekeeping applications. This standard defines a protocol that provides precise synchronization of clocks in packet-based networked systems. Synchronization of clocks can be achieved in heterogeneous systems that include clocks of different inherent precision, resolution, and stability. The protocol supports synchronization accuracy and precision in the sub-microsecond range with minimal network and local computing resources, and sub-nanosecond time transfer accuracy can be achieved in a properly designed network.

The widespread adaptation of IEEE 1588 Precision Time Protocol (PTP) is not just limited to industrial automation and measurement systems, the integration of PTP is also realized and standardized in many leading technologies/sectors like Audio-Video Bridging, Smart Power Grids and Financial Systems, and even the silicon vendors are also offering on-board sense of time support with hardware-assisted timestamping capabilities for the wide variety of embedded solutions. Additionally, PTP hardware support in mainline Linux kernel is becoming a catalyst in the development and adaptation of Linux based software solution for time synchronization applications in different sectors.

The first version of Precision Time Protocol (PTP) was standardized in 2002, and three years later, a first open-source software-only PTP implementation for Linux based systems was published. Later in 2009, an infrastructure for PTP hardware clock (Physical Clock (PHC)) control was introduced into the Linux kernel [227], while the first tool to synchronize the Linux system time to a PTP hardware clock was presented in 2011 [228].

Two main implementations of the IEEE 1588 Precision Time Protocol (PTP) are available, namely the *ptpd* [229] which only support software timestamping; and *linuxptp* [230] which support both software and hardware timestamping. This work uses the *linuxptp* implementaion for synchronization throughout the testing process. The initial design behind *linuxptp* was started in 2011 after integration of PHC API in Linux mainline kernel [228], but it was until 2015 when it was published as a full implementation of IEEE 1588 [230]. This Implementation closely resembles the "Two-Level PTP" method to synchronize the system with a PHC clock introduced by the authors of [231] in 2008. The two-level PTP method runs two levels of clock synchronization as shown if figure 6.3:

- At the network layer, PTP synchronizes the Network Interface Controller (NIC) time

(PHC) of the slave(s) with the NIC time of the master, across the network. *linuxptp* implementation of this level is called `ptp4l`;

- Inside each node, a PTP clock servo adjusts the NIC time (on the master) or system time (on PTP slaves). *linuxptp* implementation of this level is called `phc2sys`;

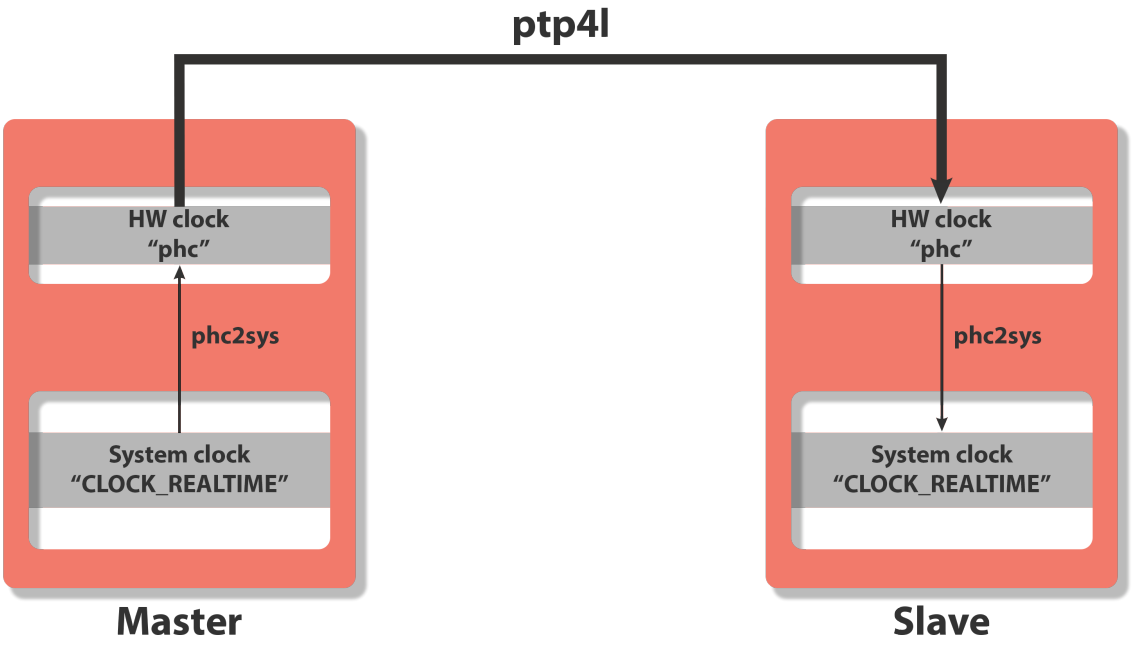


Figure 6.3: Linuxptp two-level PTP method

linuxptp also supports fully software-based synchronization -using `ptp4l`- when the network interface does not support hardware timestamping, therefore it is important to know whether a MAC supports hardware or software timestamping which defines *linuxptp* configuration. To query network driver and hardware settings and check whether a network interface supports PTP, the `ethtool` command can be used. Applying `rthtool` command on the IT2S platform (Code 31, 30, 31) shows that hardware timestamping is only supported by the Ethernet interfaces. Thus, and since all the transmissions will happen on the wireless ones, the "Two-Level PTP" method will be deployed through synchronizing the hardware clocks (`phc`) of a dedicated Ethernet interface (`enp3s0`) on each of the two IT2S platforms under test using `ptp4l` then synchronizing the system clock (`CLOCK_REALTIME`) of each platform to the corresponding hardware clock using `phc2sys`. This approach provides higher flexibility and precision in comparison to using direct software-based synchronization.

Synchronizing hardware clocks -using `ptp4l`- without any software clock intervention can be very accurate as can be seen in figure 6.4, which depicts the offset between the hardware

```
it2s-admin@pasmo-05:~$ ethtool -T wlp4s0

Time stamping parameters for wlp4s0:
Capabilities:
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
PTP Hardware Clock: none
Hardware Transmit Timestamp Modes: none
Hardware Receive Filter Modes: none
```

Code 29: WiFi Interface Time Stamping Capabilities

```
it2s-admin@pasmo-05:~$ ethtool -T wwp0s19u1u3c3

Time stamping parameters for wwp0s19u1u3c3:
Capabilities:
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
PTP Hardware Clock: none
Hardware Transmit Timestamp Modes: none
Hardware Receive Filter Modes: none
```

Code 30: LTE Interface Time Stamping Capabilities

```
it2s-admin@pasmo-05:~$ ethtool -T enp3s0

Time stamping parameters for enp3s0:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 2
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        all                   (HWTSTAMP_FILTER_ALL)
```

Code 31: Ethernet Interface Time Stamping Capabilities

clocks of the two it2s platforms' Ethernet interfaces connected directly via an Ethernet cable over a period of two hours.

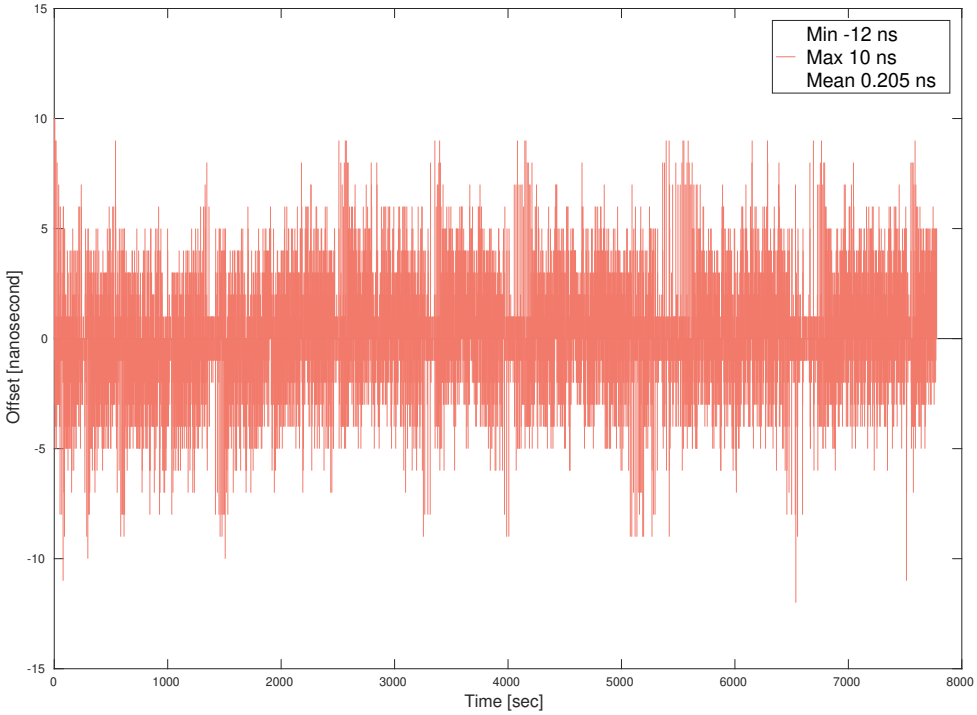


Figure 6.4: Isolated Hardware Clocks Synchronization

During this evaluation process, and since only the relative time difference between the two IT2S platforms is important, it was decided to disable any network synchronization services in order to keep the platforms' clocks as stable as possible during the testing period. Applying the "two-level PTP" method to synchronize the tested platforms passes through the following phases:

- Synchronizing the hardware clock of the master to its system clock (the overall time reference) using `phc2sys` (Figure 6.5);
- Running `ptp4l` between the hardware clocks to impose the master's system clock to the hardware clock of the slave (Figure 6.6);
- And finally using `phc2sys` in the slave platform to synchronize its system clock to the master's system clock (Figure 6.7).

Figure 6.8 shows the total offset between the two system clocks as a result of combining the delays of the three previous stages, which in this case has (the offset) an absolute average value equal to (28 ns) with 99.45% of its samples being below 100 nanosecond. Time data in

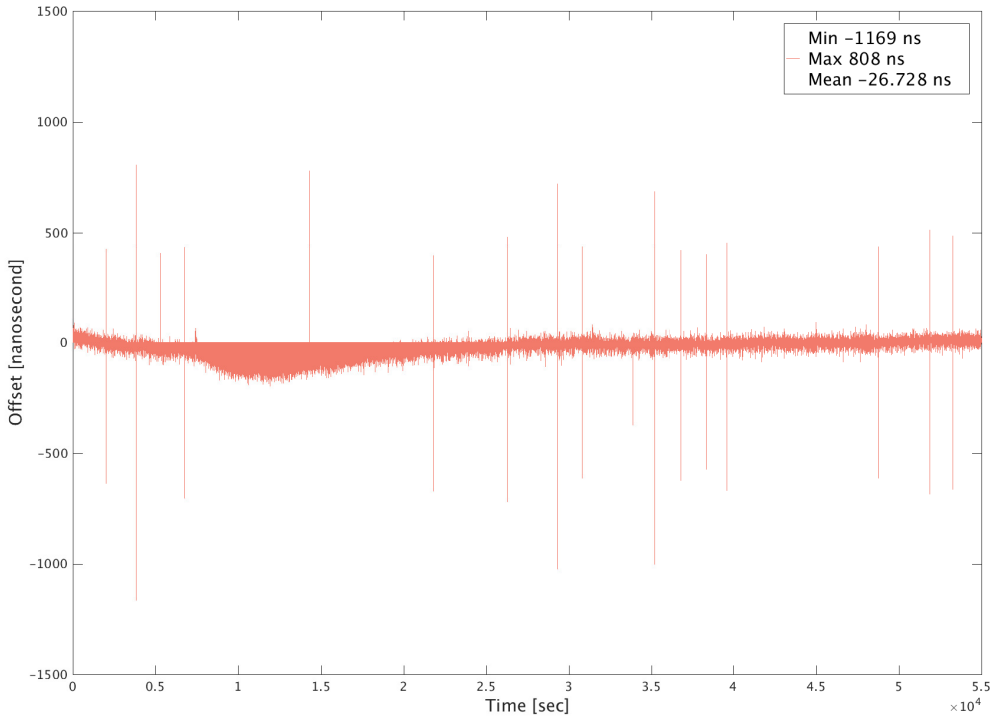


Figure 6.5: Master's clocks Synchronization

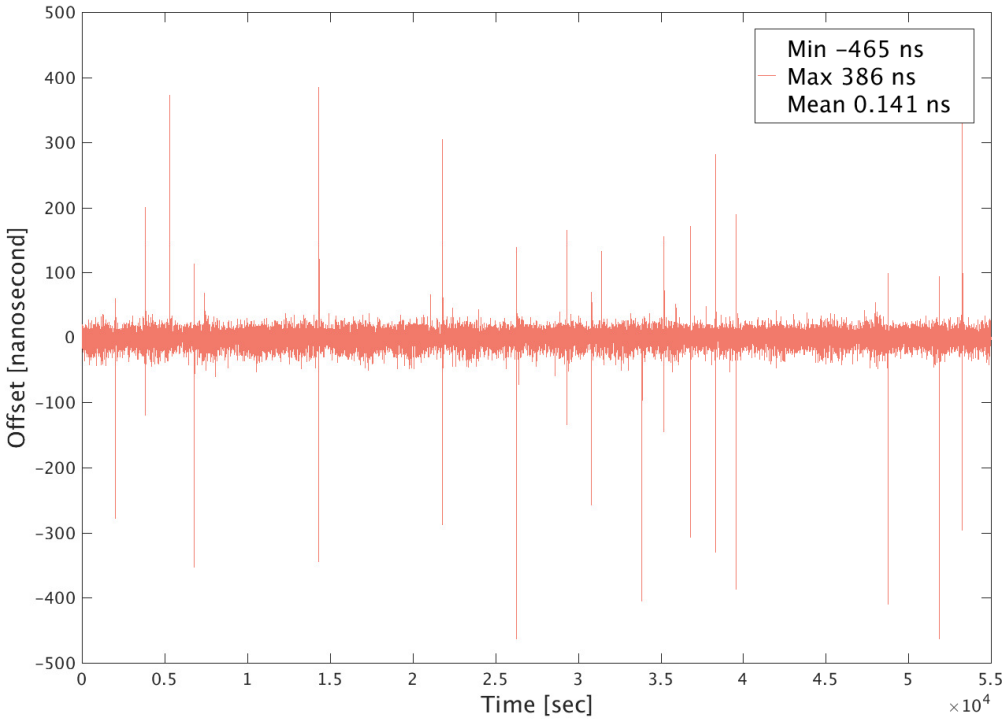


Figure 6.6: Hardware clocks Synchronization

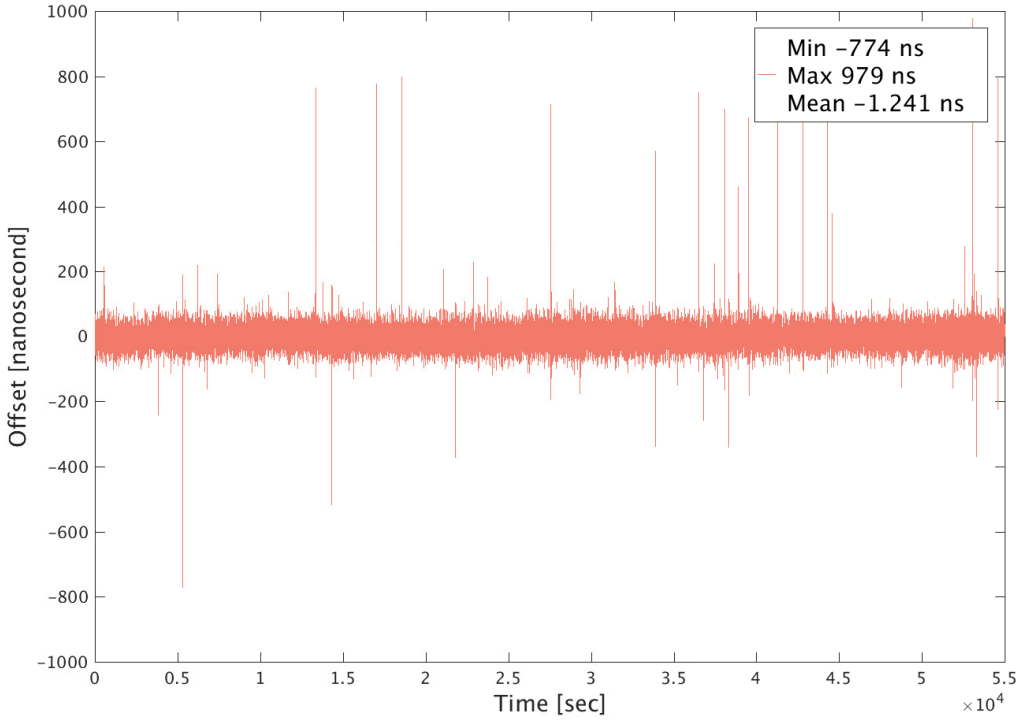


Figure 6.7: Slave's clocks Synchronization

figures 6.5, 6.6, 6.7 and 6.8 shows 15 hours of synchronization that was recorded concurrently with the evaluation tests

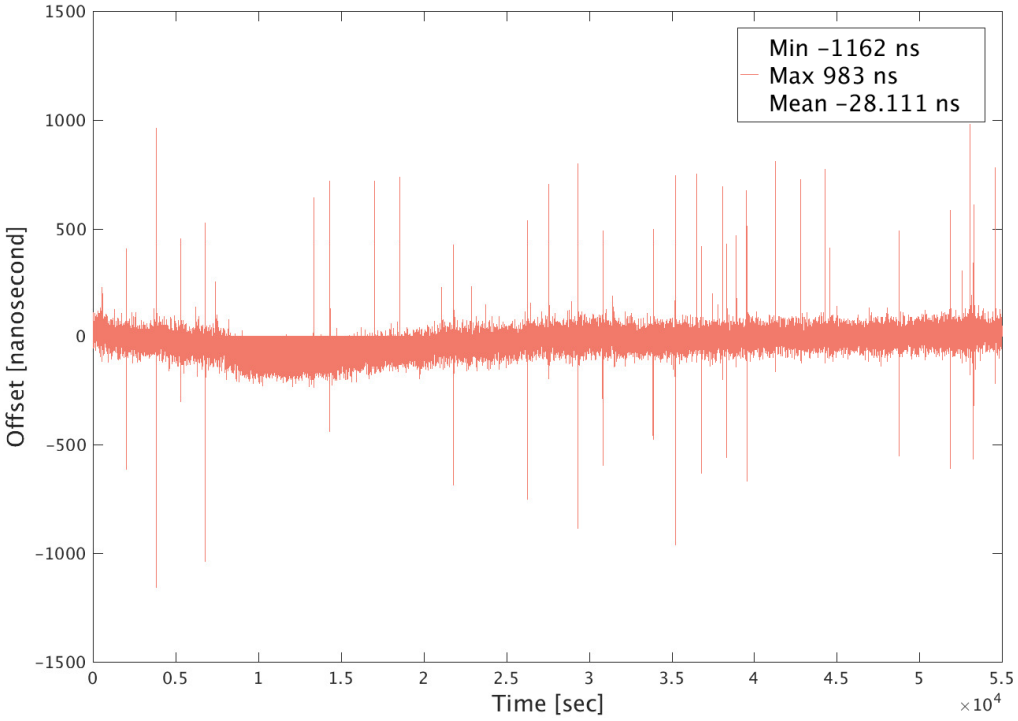


Figure 6.8: PTP Full Synchronization

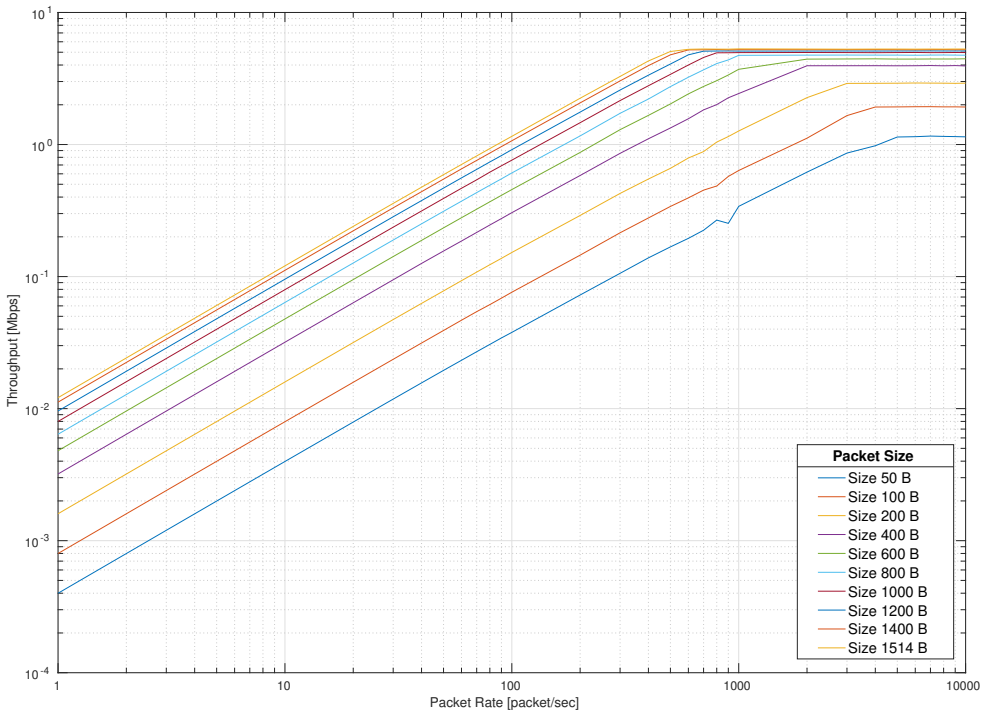
6.4 Experimental Tests and Results

With the test setup presented in figure 6.2, and following the test criteria in section 6.1, it is now the time to present the obtained results. For ITS-G5, one IT2S platform was used as a transmitter at 27 dBm transmission power witha theoretical bitrate equal to 6 Mbps, and the other platform was used as a receiver. Both stations are set to join OCB mode at 5.9 GHz and 10 MHz bandwidth with antennas placed at two meters apart. For LTE, a producer is created on one platform and a consumer on the other with each of them establishing one connection and one channel to the RabbitMQ broker. The producer publishes messages to a *fanout* exchange with no delivery acknowledgments and the consumer creates its own queue and binds it to the exchange. The RSRP of the LTE signal measured at the reception point is -111 dbm. Tests for both RATs were executed simultaneously and the collected data was redirected to spreadsheets for latter analysis and computations.

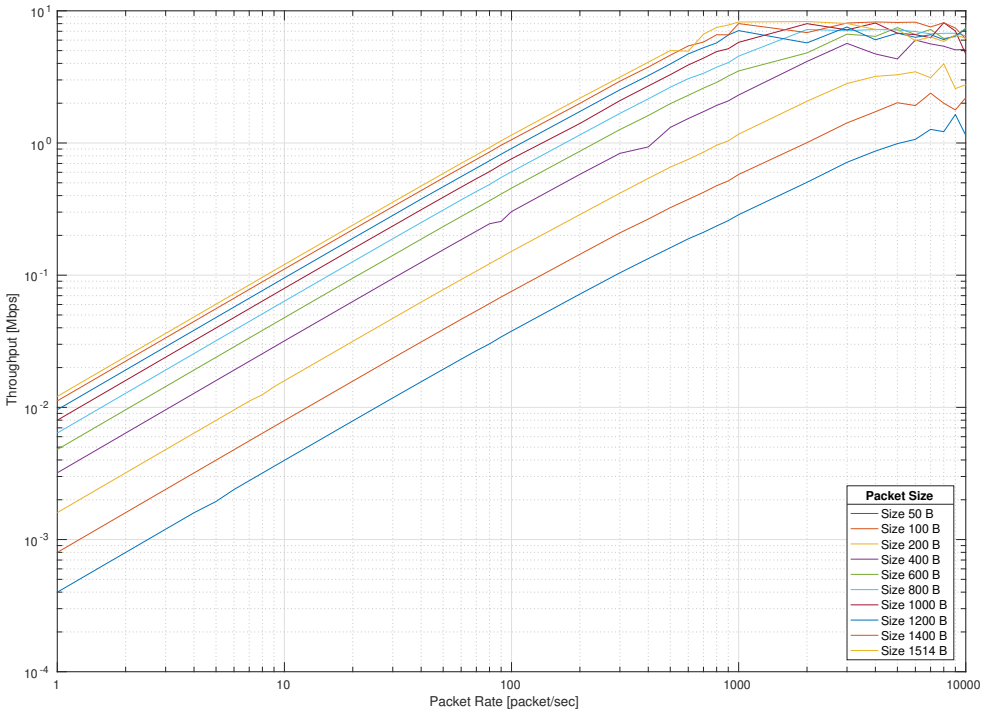
6.4.1 Throughput

For the first set of experimental results, the impact of packet size and packet transmission rate on the communication throughput were studied. It can be seen in figure 6.9 which illustrates the throughput as a function of packet rate for each tested packet size; that both RATs exhibit similar characteristics, that is the proportional increase in network throughput

along with larger number of packet transmissions per second; until a point at which the channel is saturated with traffic and the throughput remains relatively constant, even with the continuous increase in packet rate with larger packets reaching the saturation point at lower packet rates than smaller ones, while larger packets reach higher throughput values. It can also be seen that both technologies present the same throughput values at low packet rates (up to 100 packetsec) and still keep close values for packet rates up to 800 packet per second, and after that they significantly differ in favor of LTE which offers a maximum throughput above 8 Mbps compared to ITS-G5 which only reaches a maximum throughput around 5 Mbps. These observations are confirmed by figure 6.10 which depicts the overall number of delivered messages per second for ITS-G5 6.10a and LTE 6.10b.

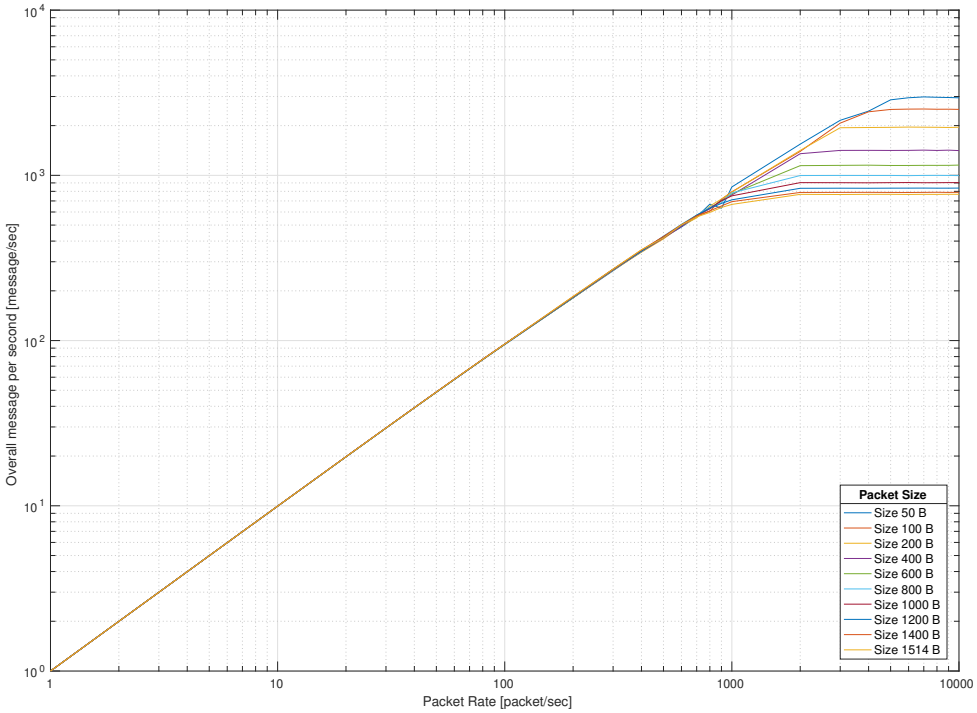


(a) ITS-G5 Throughput

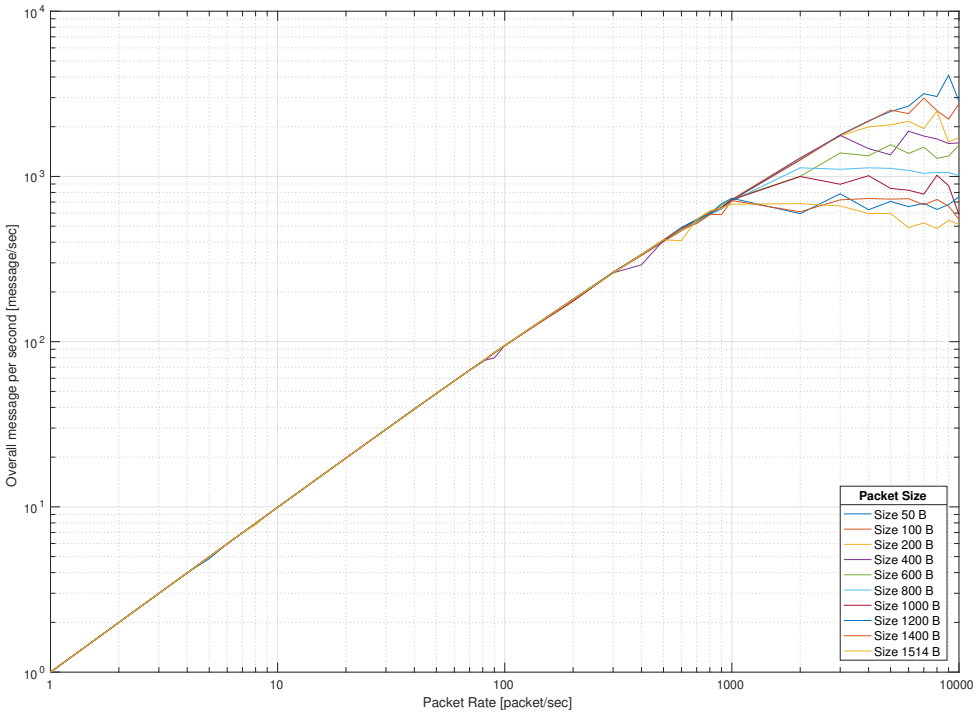


(b) LTE Throughput

Figure 6.9: Communication Throughput



(a) ITS-G5 Overall Message per Second



(b) LTE Overall Message per Second

Figure 6.10: Overall Message per second

6.4.2 Packet Loss & Packet Delivery Ratio

Equally important is the reliability, which is often characterized by the Packet Delivery Ratio (PDR) and packet loss complementary performance metrics, illustrated for ITS-G5 in figures 6.11 and 6.12 respectively. Communication over LTE using RabbitMQ did not present any data loss at any packet size or transmission rate, which might be due to TCP transmission mechanisms. Looking closely at figures 6.11 and 6.12 shows compatible results with what was found while analyzing the throughput, that is, the increase in packet size with larger number of packet transmissions per second gradually saturate the channel with traffic, leading to performance degradation and severe packet drops either in the kernel or at the device driver that reached a top of 43% at packet size 1514 B for transmission rates higher than 2000 packet per second. Another contributing factor towards this packet loss might be the lack of any session, retransmission or acknowledgment mechanism between the connection peers, which leaves the packets prone to different kinds of packet loss factors.

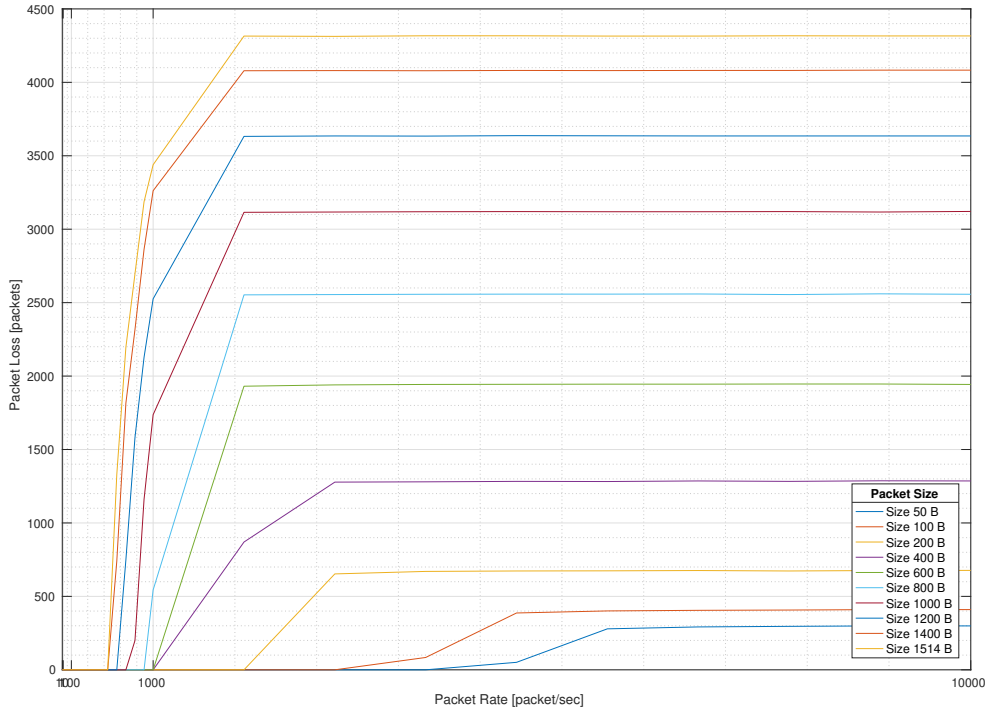


Figure 6.11: ITS-G5 Overall Packet Loss

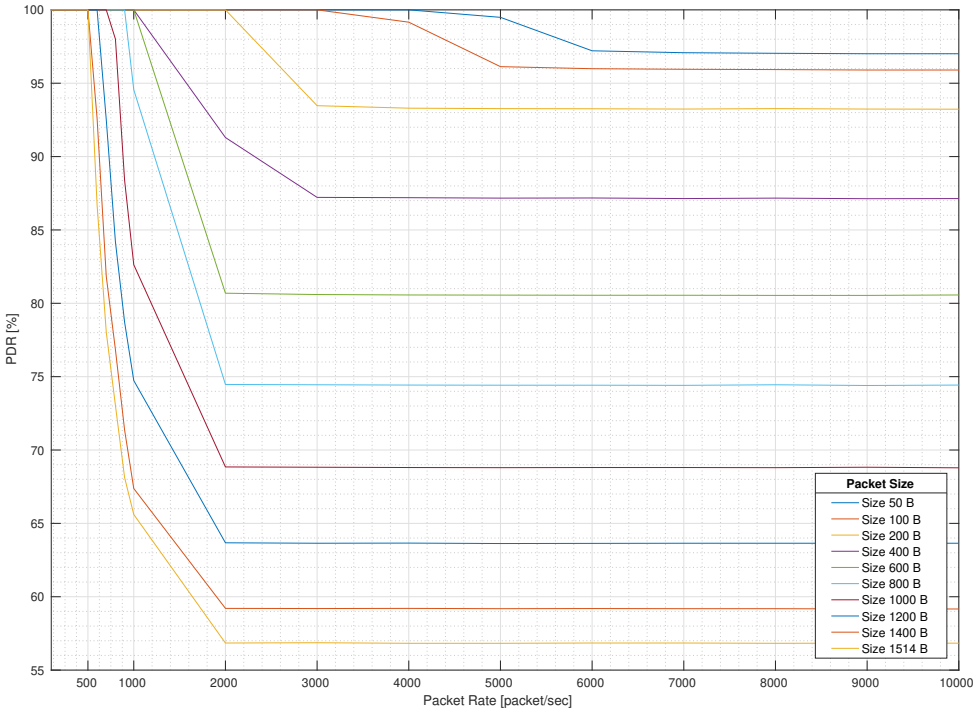
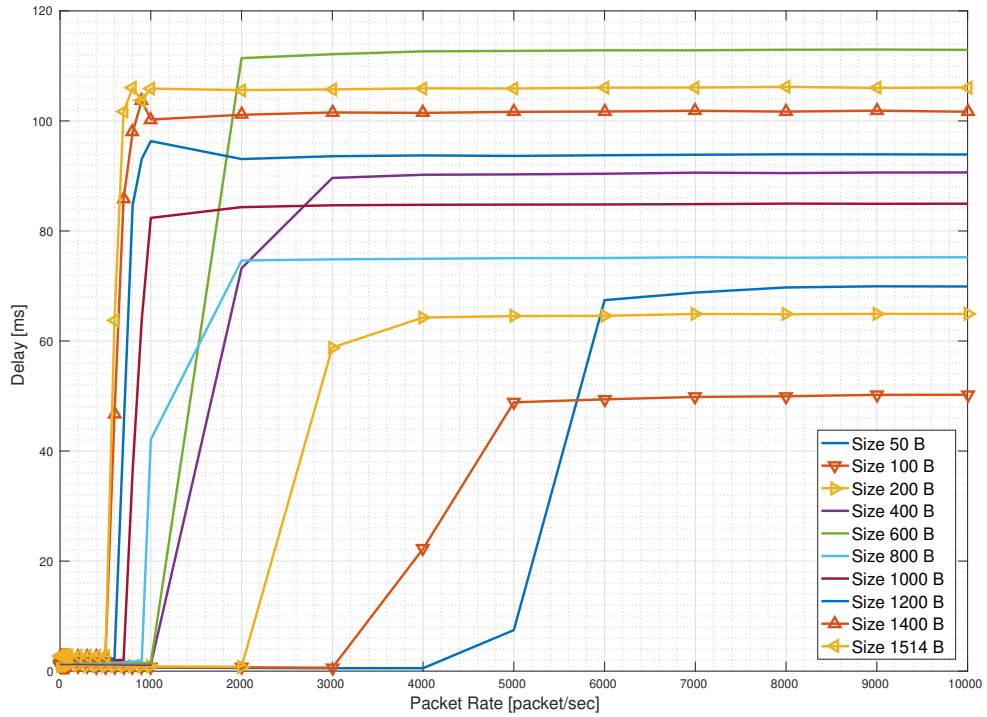


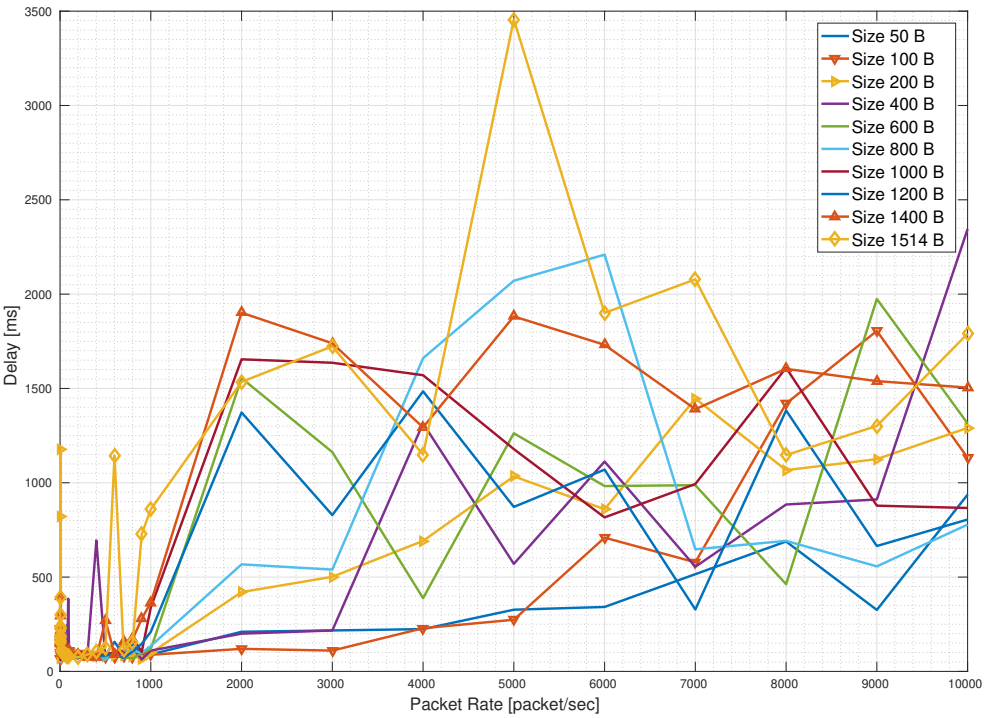
Figure 6.12: ITS-G5 Overall Packet Delivery Ratio

6.4.3 End-to-end Delay

Figure 6.13 shows a clear advantage of the direct ITS-G5 over the indirect LTE in terms of end-to-end delay, more specifically, in Figure 6.13a we can notice that ITS-G5 keeps a very low delay (around 2 ms) for all packet sizes up to transmission rates of 500 packet per second and even until higher rates for smaller packet sizes; before it starts to present significant increments and reaches a different maximum delay at a different packet rate for each packet size and stabilizes after regardless of the continuous increment of packet rate. On the other hand, figure 6.13b shows no traceable pattern for delay in LTE communication. However, the delay seems to be very inconsistent with remarkably high delays that could reach in some cases 30 times the maximum value of ITS-G5. This inconsistency in delay values can be caused by random network delays caused by different routes that each packet can take while travelling through the Internet from the producer to the broker and from the broker to the consumer, which in turn leads to high jitter values as can be clearly seen in figure 6.14 which depicts the delay values of 10000 packets of 200 Byte each, being sent at a transmission rate equals to 10 packets per second; for ITS-G5 6.14a and 6.14b.

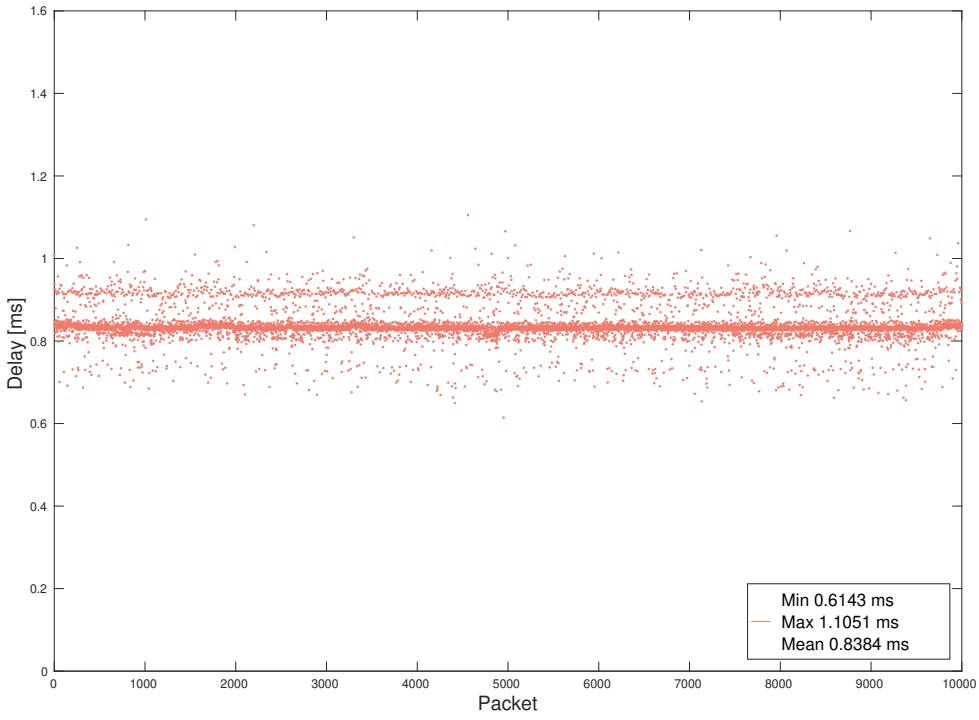


(a) ITS-G5 Average Delay

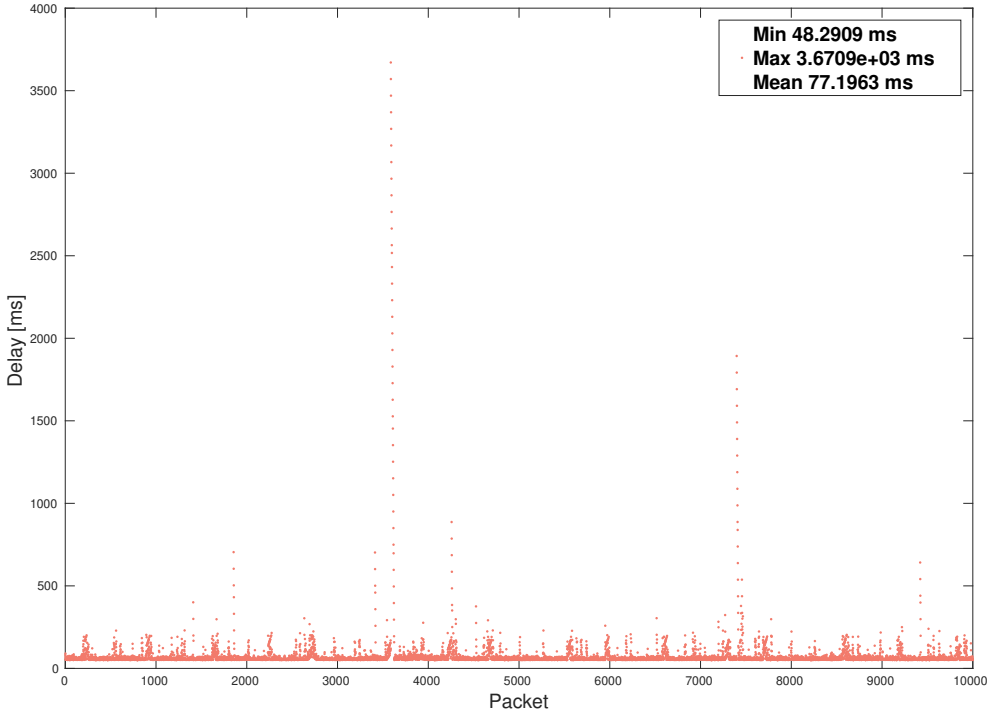


(b) LTE Average Delay

Figure 6.13: End-to-End Delay



(a) ITS-G5



(b) LTE

Figure 6.14: End-to-End Delay values of 1000 packets being sent at 10 packets per second

6.5 Results Discussion

The research and results included in this dissertation aim at offering a first look at the performance and behavior of LTE when used in an ITS scenario, and to conclude whether the presented hybrid vehicular communications solution is viable and for which type of applications. This chapter draws some general conclusions about the use of LTE and ITS-G5 for communications in ITS networks, based on the results that were presented in the previous sections of this chapter.

By studying the obtained results, it can be concluded that the suggested use case of LTE scales better and can be considered as more data reliable than ITS-G5, especially at high packet rates that can happen in real life in highly condensed transportation scenarios. However, it is not always capable of meeting the requirements of ITS safety applications due to its high delay and jitter that might be affected by the connection quality, available network load, the number of cellular network users, average distance between the UEs and the eNB, or even the RabbitMQ broker availability. ITS-G5 on the other hand, exhibits lower latency due to its direct way of communication; and similar PDR and throughput as the implemented LTE approach, in the case that the network is not operating close to its capacity, that is at low data rates or when vehicles are sparsely distributed within the same area, but it looks to lose its advantage due to high packet loss it suffers at high transmission rates.

The latencies and capacity offered by LTE under normal network conditions, promote it to be a good candidate for use in Intelligent Transportation Systems and at the same time it can accommodate for the background traffic data calls (especially in OBUs), without compromising the offered QoS beyond certain acceptable limits.

The main advantages offered by this deployment of LTE in ITS, originate from the facts that LTE is an infrastructure based standard, which allows it to mitigate many of the problems that ITS-G5 faces, such as communication range, collisions, packet losses and interference between users; in addition the TCP based connection used by RabbitMQ, which might explain how LTE managed to maintain 100% PDR even after saturating the connection at very high network loads.

The test environment along with the analyzed metrics could influence the practical relevancy of this performance evaluation. However, it gives a relevant insight on identifying the strengths and weaknesses of each technology and understanding which technology is more suitable for the given networking scenario.

In the light of the above results, it can be inferred that this presented approach of hybrid vehicular communications, that is based on a combination of ITS-G5 and LTE can cover to a large extent the ITS requirements, since ITS-G5 is perfectly suited to serve the Cooperative road safety applications due to its extremely low latencies, while LTE is more capable of serving the traffic efficiency and infotainment applications taking their load off the ITS-G5 which consequently will have no scalability or capacity issues , and will be able to offer the low latencies that are required by safety applications.

CHAPTER 7

Conclusions and Future Work

The hybrid vehicular communication model presented in this dissertation was successfully implemented and shown to be working according to the required expectations. This approach has used the available resources to create a multi-radio communication system and a local analysis based interface selection algorithm, leveraging the flexibility of the existing IT2S platform to support different categories of ITS applications, using two of the most viable communication standards for vehicular communication, namely LTE and ITS-G5. This approach eliminates the single point of system failure, increases the messages delivery probability by allowing simultaneous transmission of packets over all available interfaces. In addition, enabling the cellular link controller module can contribute into reducing the financial costs of the system; enhancing efficiency of spectrum usage, and consequently improving the overall system performance.

Improvement to the current work can be made on several fronts. Due to the flexibility of the proposed design scheme, using novel technologies, such as the IEEE 802.11bd and the 5G NR V2X, can be another aspect of improvement, allowing the design of 5G-Hybrid-V2X solutions, capable of using short range communication technologies for localized direct communication, in addition to using 5G for long range indirect communication. The other line of improvement is the the potential this work has to become the basis for further studies on the subject, by using the obtained results and conclusions to take more advantage of the coexistence of multiple radio access technologies, by allowing more intelligent cooperation models to improve the connectivity and further optimize the Quality of Service. For instance, deploying a more advanced cloud assisted protocol selection algorithm, for dynamically selecting the network interface that is likely to achieve the highest performance, not only from a single user's perspective, but rather from the entire system performance standpoint; which in turn can also be further enhanced by implementing horizontal and vertical handover mechanisms

to handle data transmission between different RATs and reinforce the interworking between the cellular and localized communication.

Referências

[1] The Directorate-General for Mobility and Transport (DG MOVE) of the European Commission, "Transport in the European Union - Current Trends and Issues", Mar. 2019. [Online]. Available: https://ec.europa.eu/transport/sites/transport/files/2019-transport-in-the-eu-current-trends-and-issues.pdf.

[2] Kantar Public Brussels on behalf of TNS opinion & social, "Standard Eurobarometer", Survey 89, Mar. 2018. [Online]. Available: http://data.europa.eu/euodp/en/data/dataset/S2180_89_1_STD89_ENG.

[3] European Commission, "EU transport in figures", Statistical pocketbook, 2017. [Online]. Available: https://ec.europa.eu/transport/facts-fundings/statistics/pocketbook-2017_en.

[4] European Environment Agency, "Passenger and freight transport demand", Indicator Assessment | Data and maps, 2018. [Online]. Available: https://www.eea.europa.eu/data-and-maps/indicators/passenger-and-freight-transport-demand/assessment.

[5] Directorate-General for Research and Innovation, "Global Europe 2050", Research and Innovation, 2012. [Online]. Available: https://ec.europa.eu/research/social-sciences/pdf/policy_reviews/global-europe-2050-report_en.pdf.

[6] European Commission, "Road safety in the European Union", Statistics, Apr. 2018. [Online]. Available: https://ec.europa.eu/transport/road_safety/sites/roadsafety/files/vademecum_2018.pdf.

[7] WHO, "Global status report on road safety", 2018. [Online]. Available: https://apps.who.int/iris/bitstream/handle/10665/276462/9789241565684-eng.pdf.

[8] ——, "Projections of mortality and causes of death, 2016 to 2060", 2018. [Online]. Available: https://www.who.int/healthinfo/global_burden_disease/projections/en/.

[9] Volvo, "Volvo trucks safety report", Statistics, 2017. [Online]. Available: https://www.volvotrucks.com/en-en/news/volvo-trucks-magazine/2018/sep/safety-report-2017.html.

[10] P. Thomas, A. Morris, R. Talbot, and H. Fagerlind, "Identifying the causes of road crashes in europe", *Annals of advances in automotive medicine / Annual Scientific Conference ... Association for the Advancement of Automotive Medicine. Association for the Advancement of Automotive Medicine. Scientific Conference*, vol. 57, pp. 13–22, Sep. 2013. [Online]. Available: https://www.researchgate.net/publication/259651507_Identifying_the_causes_of_road_crashes_in_Europe.

[11] B. A. Hamilton, "History of Intelligent Transportation Systems", U.S. Department of Transportation, Apr. 2016. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/30826/dot_30826_DS1.pdf?.

[12] ERTICO - ITS Europe, *ERTICO Timeline*. [Online]. Available: https://ertico.com/history/ (visited on 07/18/2019).

[13] PASMO, *Plataforma Aberta para o desenvolvimento e experimentação de Soluções para a MObilidade*. [Online]. Available: https://dev.es.av.it.pt/project/pasmo (visited on 04/12/2020).

[14] European Commision, *Action plan for the deployment of Intelligent Transport Systems in Europe*. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:52008DC0886 (visited on 02/16/2020).

[15] ETSI, "EN 302 665: Intelligent Transport Systems (ITS), Communications Architecture", v1.1.1, 2010.

[16] M. Alam, J. Ferreira, and J. Fonseca, *Intelligent transportation systems: Dependable vehicular communications for improved road safety*. Springer, 2016, vol. 52.

[17] EU Commission, "M/453 STANDARDISATION MANDATE ADDRESSED TO CEN, CENELEC AND ETSI IN THE FIELD OF INFORMATION AND COMMUNICATION TECHNOLOGIES TO SUPPORT THE INTEROPERABILITY OF CO-OPERATIVE SYSTEMS FOR INTELLIGENT TRANSPORT IN THE EUROPEAN COMMUNITY", standardisation 453, Jun. 2009. [Online]. Available: `https://ec.europa.eu/growth/tools-databases/mandates/index.cfm?fuseaction=search.detail&id=434`.

[18] ETSI, "EN 302 636-3: Vehicular Communications; GeoNetworking; Part 3: Network Architecture", v1.2.1, 2014.

[19] ——, "TR 102 638: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions", v1.1.1, 2009.

[20] ——, "TS 102 863: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM); Rationale for and guidance on standardization", v1.1.1, 2011.

[21] ——, "TS 302 637-2: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service", v1.4.1, 2019.

[22] I. Standardization, "Iso/iec 7498-1: 1994 information technology–open systems interconnection–basic reference model: The basic model", *International Standard ISOIEC*, vol. 74981, p. 59, 1996.

[23] ISO, "Intelligent transport systems — Station and communication architecture", ISO 21217:2020, 2020.

[24] ETSI, "TS 102 637-1: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 1: Functional Requirements", v1.1.1, 2010.

[25] ——, "TR 102 962: Intelligent Transport Systems (ITS); Framework for Public Mobile Networks in Cooperative ITS (C-ITS)", v1.1.1, 2012.

[26] P. K. Singh, S. K. Nandi, and S. Nandi, "A tutorial survey on vehicular communication state of the art, and future research directions", *Vehicular Communications*, vol. 18, p. 100 164, 2019, ISSN: 2214-2096. DOI: `https://doi.org/10.1016/j.vehcom.2019.100164`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S2214209618300901`.

[27] A.-L. Beylot and H. Labiod, *Vehicular Networks: Models and Algorithms*. Jun. 2013, pp. 1–284. DOI: `10.1002/9781118648759`.

[28] F. Li and Y. Wang, "Routing in vehicular ad hoc networks: A survey", *IEEE Vehicular technology magazine*, vol. 2, no. 2, pp. 12–22, 2007.

[29] Office of Engineering and Technology, "FCC allocates spectrum in 5.9 GHz range for Intelligent Transportation uses", Federal Communications Commission (FCC), News Release ET 99-5, Oct. 1999. [Online]. Available: `https://www.fcc.gov/document/fcc-allocates-spectrum-59-ghz-range-intelligent-transportation`.

[30] European Commission, *Commission Decision of 11 July 2005 on the harmonised use of radio spectrum in the 5 GHz frequency band for the implementation of wireless access systems including radio local area networks (WAS/RLANs)*. [Online]. Available: `https://eur-lex.europa.eu/eli/dec/2005/513/oj` (visited on 02/16/2020).

[31] ——, *Commission Decision of 5 August 2008 on the harmonised use of radio spectrum in the 5875 - 5905 MHz frequency band for safety-related applications of Intelligent Transport Systems (ITS) (notified under document number C(2008) 4145)e*. [Online]. Available: `https://eur-lex.europa.eu/eli/dec/2008/671/oj` (visited on 02/16/2020).

[32] ETSI, "EN 302 663: Intelligent Transport Systems (ITS); ITS-G5 Access Layer Specification for Intelligent Transport Systems Operating in the 5 GHz Frequency Band", v1.2.0, 2012.

[33] S. Peng, G. Lee, R. Klette, and C. Hsu, *Internet of Vehicles. Technologies and Services for Smart Cities: 4th International Conference, IOV 2017, Kanazawa, Japan, November 22-25, 2017, Proceedings*,

ser. Lecture Notes in Computer Science. Springer International Publishing, 2017, ISBN: 9783319723297. [Online]. Available: https://books.google.pt/books?id=y7E%5C_DwAAQBAJ.

[34] L. Wang and G.-S. G. S. Kuo, "Mathematical modeling for network selection in heterogeneous wireless networks — a tutorial", *IEEE Communications Surveys & Tutorials*, vol. 15, pp. 271–292, 2013.

[35] EU Commission, "C-ITS Platform final report - Working Group 6-Technical Issues–Hybrid Communications and Spectrum allocation", report, Jan. 2016. [Online]. Available: https://ec.europa.eu/transport/sites/transport/files/themes/its/doc/c-its-platform-final-report-january-2016.pdf.

[36] Office of Engineering and Technology, "Amendment of Parts 2 and 90 of the Commission's Rules to Allocate the 5.850-5.925 GHz Band to the Mobile Service for Dedicated Short Range Communications of Intelligent Transportation Services", Federal Communications Commission (FCC), NOTICE OF PROPOSED RULE MAKING FCC 98-119, Jun. 1998. [Online]. Available: https://transition.fcc.gov/Bureaus/Engineering_Technology/Notices/1998/fcc98119.pdf.

[37] ASTM E2213-02, "Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle Systems — 5 GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications", ASTM International, West Conshohocken, PA, 2002.

[38] ASTM E2213-03, "Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle Systems — 5 GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications", ASTM International, West Conshohocken, PA, 2003.

[39] *IEEE Std 1609.2-2016 - IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages*, 2016.

[40] *IEEE Std 1609.3-2016 - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Networking Services*, 2016.

[41] *IEEE Std 1609.4-2016 - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Multi-Channel Operation*, 2016.

[42] *IEEE Std 1609.11-2010 - IEEE Standard for Wireless Access in Vehicular Environments (WAVE)–Over-the-Air Electronic Payment Data Exchange Protocol for Intelligent Transportation Systems (ITS)*, 2010.

[43] *IEEE Std 1609.12-2019 - IEEE Standard for Wireless Access in Vehicular Environments (WAVE)–Identifiers*, 2019.

[44] ETSI, "EN 302 663: Intelligent Transport Systems (ITS); ITS-G5 Access Layer Specification for Intelligent Transport Systems Operating in the 5 GHz Frequency Band", v1.3.0, 2019.

[45] ISO, "Intelligent transport systems — Localized communications — ITS-M5", ISO 21215:2018, 2018.

[46] ——, "Intelligent transport systems — Communications access for land mobiles (CALM) — Architecture", ISO 21217:2014, 2014.

[47] CEN, "Road transport and traffic telematics - Dedicated short-range communication - Physical layer using microwave at 5,8 GHz", EN 12253, 2004.

[48] ——, "Road transport and traffic telematics - Dedicated Short Range Communication (DSRC) - DSRC data link layer: medium access and logical link control", EN 12795, 2003.

[49] ——, "Road transport and traffic telematics - Dedicated Short Range Communication (DSRC) - DSRC application layer", EN 12834, 2003.

[50] ETSI, "EN 300 674-2-1: Transport and Traffic Telematics (TTT); Dedicated Short Range Communication (DSRC) transmission equipment (500 kbit/s / 250 kbit/s) operating in the 5 795 MHz to 5 815 MHz frequency band; Part 2: Harmonised Standard covering the essential requirements of article 3.2 of the Directive 2014/53/EU; Sub-part 1: Road Side Units (RSU)", v2.1.1, 2016.

[51] ——, “EN 300 674-2-1: Transport and Traffic Telematics (TTT); Dedicated Short Range Communication (DSRC) transmission equipment (500 kbit/s / 250 kbit/s) operating in the 5 795 MHz to 5 815 MHz frequency band; Part 2: Harmonised Standard covering the essential requirements of article 3.2 of the Directive 2014/53/EU; Sub-part 2: Road Side Units (RSU)”, v2.1.1, 2016.

[52] ——, “ ES 200 674-1: Intelligent Transport Systems (ITS); Road Transport and Traffic Telematics (RTTT); Dedicated Short Range Communications (DSRC); Part 1: Technical characteristics and test methods for High Data Rate (HDR) data transmission equipment operating in the 5,8 GHz Industrial, Scientific and Medical (ISM) band”, v2.2.1, 2011.

[53] K. Abboud, H. A. Omar, and W. Zhuang, “Interworking of dsrc and cellular network technologies for v2x communications: A survey”, *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9457–9470, Dec. 2016. DOI: 10.1109/TVT.2016.2591558.

[54] “Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Wireless access in vehicular environments”, *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)*, pp. 1–51, 2010.

[55] “Ieee standard for wireless lan medium access control (mac) and physical layer (phy) specifications”, *IEEE Std 802.11-1997*, pp. 1–445, 1997.

[56] X. Wu, J. Li, R. M. Scopigno, and H. A. Cozzetti, “Insights into possible vanet 2.0 directions”, in *Vehicular ad hoc Networks*, Springer, 2015, pp. 411–455.

[57] K. Liu, J. K. Ng, V. C. Lee, S. H. Son, and I. Stojmenovic, “Cooperative data scheduling in hybrid vehicular ad hoc networks: Vanet as a software defined network”, *IEEE/ACM transactions on networking*, vol. 24, no. 3, pp. 1759–1773, 2015.

[58] A. Fonseca and T. Vazão, “Applicability of position-based routing for vanet in highways and urban environment”, *Journal of Network and Computer Applications*, vol. 36, no. 3, pp. 961–973, 2013.

[59] “Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications”, *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pp. 1–1076, 2007.

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

[61] “Ieee standard for information technology–local and metropolitan area networks–specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications - amendment 8: Medium access control (mac) quality of service enhancements”, *IEEE Std 802.11e-2005 (Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003)*, pp. 1–212, 2005.

[62] A. Fitah, A. Badri, M. Moughit, and A. Sahel, “Performance of dsrc and wifi for intelligent transport systems in vanet”, *Procedia Computer Science*, vol. 127, pp. 360–368, 2018.

[63] P. K. Singh, S. Sharma, S. K. Nandi, and S. Nandi, “Multipath tcp for v2i communication in sdn controlled small cell deployment of smart city”, *Vehicular communications*, vol. 15, pp. 1–15, 2019.

[64] “Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications”, *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1–2793, 2012.

[65] IEEE, *IEEE P802.11-TASK GROUP BD (NGV) MEETING UPDATE*. [Online]. Available: http://www.ieee802.org/11/Reports/tgbd_update.htm (visited on 04/16/2020).

[66] “IEEE Guide for Wireless Access in Vehicular Environments (WAVE) Architecture”, *IEEE Std 1609.0-2019 (Revision of IEEE Std 1609.0-2013)*, 2019.

[67] I. Std ISO, "Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 2: Logical Link Control8802-2: 1998", *IEEE Std*, 1998.

[68] T. ETSI, "Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part", *Intelligent Transport Systems (ITS)*, no. 102 687 V1.2.1, pp. 1–45, 2018.

[69] ETSI, *4th Generation (LTE)*. [Online]. Available: https://www.etsi.org/technologies/mobile/4g (visited on 08/23/2019).

[70] 3GPP, "UTRA-UTRAN Long Term Evolution (LTE) and 3GPP System Architecture Evolution (SAE)", 3GPP, Tech. Rep., 2006.

[71] S. Sesia, I. Toufik, and M. Baker, *LTE-the UMTS long term evolution: from theory to practice*. John Wiley & Sons, 2011.

[72] C. Cox, *An Introduction to LTE: LTE, LTE-Advanced, SAE and 4G Mobile Communications*, 1st. Wiley Publishing, 2012, ISBN: 9781119970385.

[73] A. Kukushkin, *Introduction to Mobile Network Engineering: GSM, 3G-WCDMA, LTE and the Road to 5G*. Wiley, 2018, ISBN: 9781119484172. [Online]. Available: https://books.google.pt/books?id=8OCWtAEACAAJ.

[74] 3GPP, "Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN)", 3rd Generation Partnership Project (3GPP), TR 25.913, Jun. 2005. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/25913.htm.

[75] 3GPP, "Feasibility study for evolved Universal Terrestrial Radio Access (UTRA) and Universal Terrestrial Radio Access Network (UTRAN)", 3rd Generation Partnership Project (3GPP), TR 25.912, Mar. 2005. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/25912.htm.

[76] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-Advanced for Mobile Broadband*, 1st. Orlando, FL, USA: Academic Press, Inc., 2011, ISBN: 9780123854896.

[77] Ericson, "LTE Performance Evaluation - Uplink Summary", 3rd Generation Partnership Project (3GPP), RP 072444, May 2007. [Online]. Available: https://www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_49/Docs/R1-072444.zip.

[78] Nokia, "LTE Performance Evaluation - Uplink Summary", 3rd Generation Partnership Project (3GPP), RP 072261, May 2007. [Online]. Available: https://www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_49/Docs/R1-072261.zip.

[79] 3GPP, *3GPP Portal*. [Online]. Available: https://portal.3gpp.org/ (visited on 08/28/2019).

[80] J. Penttinen, *The LTE-Advanced Deployment Handbook: The Planning Guidelines for the Fourth Generation Networks*. Wiley, 2016, ISBN: 9781118484807. [Online]. Available: https://books.google.pt/books?id=H67QCgAAQBAJ.

[81] Working Party 5D, "Submission and evaluation process and consensus building", ITU-R, ITU Report, Mar. 2008. [Online]. Available: https://www.itu.int/md/R07-IMT.ADV-C-0002/en.

[82] ITU-R, "Invitation for submission of proposals for candidate radio interface technologies for the terrestrial components of the radio interface(s) for IMT-Advanced and invitation to participate in their subsequent evaluation", ITU-R SG5, Circular Letter 5/LCCE/2, Mar. 2008. [Online]. Available: https://www.itu.int/md/R00-SG05-CIR-0002/en.

[83] ITU-R, "Requirements, Evaluation Criteria and Submission Templates for the Development of IMT-Advanced", ITU Report M.2133, 2008. [Online]. Available: https://www.itu.int/pub/R-REP-M.2133-2008.

[84] ITU-R, "Requirements related to technical performance for IMT-Advanced radio interface(s)", ITU Report M.2134, 2008. [Online]. Available: https://www.itu.int/pub/R-REP-M.2134-2008.

[85] ITU-R, "Guidelines for evaluation of radio interface technologies for IMT-Advanced", ITU Report M.2134, 2008. [Online]. Available: https://www.itu.int/pub/R-REP-M.2135-2008.

[86] 3GPP, "Requirements for further advancements for Evolved Universal Terrestrial Radio Access (E-UTRA) (LTE-Advanced)", 3rd Generation Partnership Project (3GPP), TR 36.913, Jun. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/36913.htm.

[87] 3GPP, "Feasibility study for Further Advancements for E-UTRA (LTE-Advanced)", 3rd Generation Partnership Project (3GPP), TR 36.912, Sep. 2009. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/36912.htm.

[88] 3GPP TSG RAN, "Initial 3GPP submission of a candidate IMT-Advanced technology", 3rd Generation Partnership Project (3GPP), RP 080763, Sep. 2008. [Online]. Available: http://www.3gpp.org/ftp/tsg_ran/TSG_RAN/TSGR_41/docs/RP-080763.zip.

[89] 3GPP TSG RAN, "3GPP Submission Package for IMT-Advanced", 3rd Generation Partnership Project (3GPP), RP 090939, Sep. 2009. [Online]. Available: http://www.3gpp.org/ftp/tsg_ran/TSG_RAN/TSGR_45/Docs/RP-090939.zip.

[90] Working Party 5D, "Acknowledgement of candidate submission from 3GPP proponent (3GPP organization partners of ARIB, ATIS, CCSA, ETSI, TTA AND TTC) under Step 3 of the IMT-Advanced process (3GPP technology)", ITU-R, ITU Report, Oct. 2009. [Online]. Available: https://www.itu.int/md/R07-IMT.ADV-C-0008/en.

[91] ITU, *ITU paves way for next-generation 4G mobile technologies - ITU-R IMT-Advanced 4G standards to usher new era of mobile broadband communications*. [Online]. Available: https://www.itu.int/net/pressoffice/press_releases/2010/40.aspx (visited on 08/25/2019).

[92] 5G Americas, *5G & LTE Deployments*, Statistics. [Online]. Available: https://www.5gamericas.org (visited on 09/18/2019).

[93] Ovum, Statistics. [Online]. Available: https://ovum.informa.com/ (visited on 09/18/2019).

[94] 3GPP, *Release 12*, Mar. 2015. [Online]. Available: https://www.3gpp.org/specifications/releases/68-release-12/ (visited on 02/15/2020).

[95] D. Flore, "Initial cellular v2x standard completed", *3GPP*, Sep. 2016. [Online]. Available: https://www.3gpp.org/news-events/3gpp-news/1798-v2x_r14 (visited on 02/15/2020).

[96] M. Wang, M. Winbjork, Z. Zhang, R. Blasco, H. Do, S. Sorrentino, M. Belleschi, and Y. Zang, "Comparison of lte and dsrc-based connectivity for intelligent transportation systems", pp. 1–5, Jun. 2017. DOI: 10.1109/VTCSpring.2017.8108284.

[97] S. Gopinath, L. Wischhof, M. Jaumann, C. Ponikwar, and H.-J. Hof, "On context-aware communication mode selection in hybrid vehicular networks", 2016.

[98] S. Andreev, O. Galinina, A. Pyattaev, K. Johnsson, and Y. Koucheryavy, "Analyzing assisted offloading of cellular user sessions onto d2d links in unlicensed bands", *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 1, pp. 67–80, 2014.

[99] W. Gay, *Linux Socket Programming by Example*, ser. By Example Series. Que, 2000, ISBN: 9780789722416. [Online]. Available: https://books.google.pt/books?id=xrhGOP91ZWkC.

[100] *socket(7) - Linux man page*, Nov. 2019.

[101] N. Yocom, J. Turner, and K. Davis, *The Definitive Guide to Linux Network Programming*, ser. Books for professionals by professionals. Apress, 2004, ISBN: 9781590593226. [Online]. Available: https://books.google.pt/books?id=YM9QAAAAMAAJ.

[102] W. Stevens, B. Fenner, and A. Rudoff, *UNIX Network Programming: The sockets networking API*, ser. Addison-Wesley professional computing series. Addison-Wesley, 2004, ISBN: 9780131411555. [Online]. Available: https://books.google.pt/books?id=cdBQAAAAMAAJ.

[103] M. Kerrisk, *The Linux Programming Interface*, ser. No Starch Press Series. No Starch Press, 2010, ISBN: 9781593272203. [Online]. Available: https://books.google.pt/books?id=5J9wmAEACAAJ.

[104] IBM, *Socket programming*, 7.2. IBM Corporation, 2013.

[105] S. Walton, *Linux Socket Programming*, ser. Sams White Bks. Sams, 2001, ISBN: 9780672319358. [Online]. Available: https://books.google.pt/books?id=adlQAAAAMAAJ.

[106] P. Wang, *Mastering Modern Linux*. Taylor & Francis Group, 2018, ISBN: 9780815381112. [Online]. Available: https://books.google.pt/books?id=uFwptAEACAAJ.

[107] *read(2) - Linux man page*, Nov. 2019.

[108] *write(2) - Linux man page*, Nov. 2019.

[109] "Protocol numbers", IANA, Tech. Rep., 2020. [Online]. Available: https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml#protocol-numbers-1 (visited on 01/11/2020).

[110] J. P. J. Reynolds, "Assigned numbers", IETF, RFC, Tech. Rep. 1700, 1994. [Online]. Available: https://tools.ietf.org/html/rfc1700 (visited on 01/11/2020).

[111] *Advanced Message Queuing Protocol*, Jul. 2019. [Online]. Available: http://www.amqp.org/about/what (visited on 07/22/2019).

[112] A. Consortium *et al.*, "Amqp: Advanced message queuing protocol", Version 0-9-1. Protocol specification, AMQP Consortium, Tech. Rep., 2008.

[113] O. S. OASIS, "Oasis advanced message queuing protocol (amqp) version 1.0", *OASIS, Burlington, MA, USA*, 2012.

[114] *RabbitMQ Documentation*, Aug. 2019. [Online]. Available: http://next.rabbitmq.com/documentation.html (visited on 03/11/2019).

[115] *RabbitMQ Tutorials - AMQP 0-9-1 Model Explained*, Aug. 2019. [Online]. Available: https://www.rabbitmq.com/tutorials/amqp-concepts.html (visited on 03/11/2019).

[116] *RabbitMQ Queues*, Aug. 2019. [Online]. Available: https://www.rabbitmq.com/queues.html (visited on 03/11/2019).

[117] 3GPP, "Study on LTE-based V2X services", 3rd Generation Partnership Project (3GPP), TR 36.885, 2016. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2934.

[118] M. Muhammad and G. A. Safdar, "Survey on existing authentication issues for cellular-assisted v2x communication", *Vehicular Communications*, vol. 12, pp. 50–65, 2018.

[119] M. Mueck and I. Karls, *Networking Vehicles to Everything: Evolving Automotive Solutions*. De Gruyter, Incorporated, 2018, ISBN: 9781501507205. [Online]. Available: https://books.google.pt/books?id=UEBEDwAAQBAJ.

[120] 3GPP, *Release 16*. [Online]. Available: https://www.3gpp.org/release-16 (visited on 03/28/2020).

[121] J. Thota, N. F. Abdullah, A. Doufexi, and S. Armour, "Performance of car to car safety broadcast using cellular v2v and ieee 802.11p", in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, Jun. 2018, pp. 1–5. DOI: 10.1109/VTCSpring.2018.8417885.

[122] ETSI, "TS 122 185: LTE;Service requirements for V2X services (3GPP TS 22.185 version 14.3.0 Release 14)", v14.3.0, 2017.

[123] 3GPP, "Vocabulary for 3GPP Specifications", 3rd Generation Partnership Project (3GPP), TR 21.905, 2019. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=558.

[124] H. A. Omar, W. Zhuang, A. Abdrabou, and L. Li, "Performance evaluation of vemac supporting safety applications in vehicular networks", *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 1, pp. 69–83, 2013.

[125] IEEE High Efficiency (HE) Wireless LAN Task Group, *Status of Project IEEE 802.11ax*. [Online]. Available: http://www.ieee802.org/11/Reports/tgax_update.htm (visited on 04/10/2020).

[126] "Ieee draft standard for information technology – telecommunications and information exchange between systems local and metropolitan area networks – specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment enhancements for high efficiency wlan", *IEEE P802.11ax/D3.0, June 2018*, pp. 1–682, 2018.

[127] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi, "A tutorial on ieee 802.11ax high efficiency wlans", *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 197–216, 2019.

[128] F. Borgonovo, A. Capone, M. Cesana, and L. Fratta, "Adhoc mac: New mac architecture for ad hoc networks providing efficient and reliable point-to-point and broadcast services", *Wireless networks*, vol. 10, no. 4, pp. 359–366, 2004.

[129] H. A. Omar, W. Zhuang, and L. Li, "Vemac: A tdma-based mac protocol for reliable broadcast in vanets", *IEEE Transactions on Mobile Computing*, vol. 12, no. 9, pp. 1724–1736, 2013.

[130] G. V. Rossi and K. K. Leung, "Optimised csma/ca protocol for safety messages in vehicular ad-hoc networks", in *2017 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2017, pp. 689–696.

[131] N. Shahin and Y.-T. Kim, "Scalable tdma cluster-based mac (stcm) for multichannel vehicular networks", in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, 2017, pp. 13–18.

[132] Y. Cao, H. Zhang, X. Zhou, and D. Yuan, "A scalable and cooperative mac protocol for control channel access in vanets", *IEEE Access*, vol. 5, pp. 9682–9690, 2017.

[133] W. Zhu, D. Gao, C. H. Foh, H. Zhang, and H. Chao, "Reliable emergency message dissemination protocol for urban internet of vehicles", *IET Communications*, vol. 11, no. 8, pp. 1275–1281, 2017.

[134] R. Merz, D. Wenger, D. Scanferla, and S. Mauron, "Performance of lte in a high-velocity environment: A measurement study", in *Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges*, 2014, pp. 47–52.

[135] 3GPP, "Multimedia Broadcast/Multicast Service (MBMS); Architecture and functional description", 3rd Generation Partnership Project (3GPP), TS 23.246, 2019. [Online]. Available: `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=829`.

[136] A. Mahmood, B. Butler, Q. Z. Sheng, W. E. Zhang, and B. Jennings, "Need of ambient intelligence for next-generation connected and autonomous vehicles", in *Guide to Ambient Intelligence in the IoT Environment: Principles, Technologies and Applications*, Z. Mahmood, Ed. Cham: Springer International Publishing, 2019, pp. 133–151, ISBN: 978-3-030-04173-1. DOI: `10.1007/978-3-030-04173-1_6`. [Online]. Available: `https://doi.org/10.1007/978-3-030-04173-1_6`.

[137] J. Lianghai, M. Liu, A. Weinand, and H. D. Schotten, "Direct vehicle-to-vehicle communication with infrastructure assistance in 5g network", in *2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, 2017, pp. 1–5.

[138] L. Hu, J. Eichinger, M. Dillinger, M. Botsov, and D. Gozalvez, "Unified device-to-device communications for low-latency and high reliable vehicle-to-x services", in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, IEEE, 2016, pp. 1–7.

[139] D. Kim, I. Yeom, and T.-J. Lee, "Mitigating tail latency in ieee 802.11–based networks", *International Journal of Communication Systems*, vol. 31, no. 1, e3404, 2018.

[140] T. Sukuvaara, K. Mäenpää, and R. Ylitalo, "Vehicular-networking-and road-weather-related research in sodankylä", *Geoscientific Instrumentation, Methods and Data Systems*, vol. 5, no. 2, p. 513, 2016.

[141] K. Tokarz, "A review on the vehicle to vehicle and vehicle to infrastructure communication", in *Man-Machine Interactions 6*, A. Gruca, T. Czachórski, S. Deorowicz, K. Harężlak, and A. Piotrowska, Eds., Cham: Springer International Publishing, 2020, pp. 44–52, ISBN: 978-3-030-31964-9.

[142] M. N. Tahir, K. Maenpaa, and D. T. Sukuvaara, "Evolving wireless vehicular communication system level comparison and analysis of 802,11 p, 4g 5g", in *2019 2nd International Conference on Communication, Computing and Digital systems (C-CODE)*, Mar. 2019, pp. 48–52. DOI: `10.1109/C-CODE.2019.8680977`.

[143] M. Muhammad and G. A. Safdar, "Survey on existing authentication issues for cellular-assisted v2x communication", *Vehicular Communications*, vol. 12, pp. 50–65, 2018.

[144] G. Araniti, C. Campolo, M. Condoluci, A. Iera, and A. Molinaro, "Lte for vehicular networking: A survey", *IEEE Communications Magazine*, vol. 51, no. 5, pp. 148–157, 2013.

[145] K. Yang, S. Ou, H.-H. Chen, and J. He, "A multihop peer-communication protocol with fairness guarantee for ieee 802.16-based vehicular networks", *IEEE Transactions on Vehicular Technology*, vol. 56, no. 6, pp. 3358–3370, 2007.

[146] H. Zhang, F. Bai, and X. Ju, "Heterogeneous vehicular wireless networking: A theoretical perspective", in *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, 2015, pp. 1936–1941.

[147] S. Céspedes and X. Shen, "On achieving seamless ip communications in heterogeneous vehicular networks", *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3223–3237, 2015.

[148] I. Lequerica, P. M. Ruiz, and V. Cabrera, "Improvement of vehicular communications by using 3g capabilities to disseminate control information", *IEEE network*, vol. 24, no. 1, pp. 32–38, 2010.

[149] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, and E. Cerqueira, "Towards software-defined vanet: Architecture and services", in *2014 13th annual Mediterranean ad hoc networking workshop (MED-HOC-NET)*, IEEE, 2014, pp. 103–110.

[150] Y. Wang, J. Jose, J. Li, X. Wu, and S. Subramanian, *Opportunistic use of the dsrc spectrum*, US Patent App. 13/921,706, 2014.

[151] G. El Mouna Zhioua, N. Tabbane, H. Labiod, and S. Tabbane, "A fuzzy multi-metric qos-balancing gateway selection algorithm in a clustered vanet to lte advanced hybrid cellular network", *IEEE Transactions on Vehicular Technology*, vol. 64, no. 2, pp. 804–817, 2014.

[152] S. Bi, C. Chen, R. Du, and X. Guan, "Proper handover between vanet and cellular network improves internet access", in *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, IEEE, 2014, pp. 1–5.

[153] J. Park, W. Lee, and S. Lee, "An intelligent interface selection scheme for vehicular communication system using lte and ieee 802.11p", in *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, Dec. 2016, pp. 868–870. DOI: 10.1109/ICTC.2016.7763318.

[154] B. Liu, D. Jia, J. Wang, K. Lu, and L. Wu, "Cloud-assisted safety message dissemination in vanet-cellular heterogeneous wireless network", *ieee systems journal*, vol. 11, no. 1, pp. 128–139, 2017.

[155] S. Ansari, T. Boutaleb, S. Sinanovic, C. Gamio, and I. Krikidis, "Mhav: Multitier heterogeneous adaptive vehicular network with lte and dsrc", *Ict Express*, vol. 3, no. 4, pp. 199–203, 2017.

[156] T. Koskela, S. Hakola, T. Chen, and J. Lehtomäki, "Clustering concept using device-to-device communication in cellular system", in *2010 IEEE Wireless Communication and Networking Conference*, 2010, pp. 1–6.

[157] A. Benslimane, T. Taleb, and R. Sivaraj, "Dynamic clustering-based adaptive mobile gateway management in integrated vanet—3g heterogeneous wireless networks", *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 3, pp. 559–570, 2011.

[158] S. Ucar, S. C. Ergen, and O. Ozkasap, "Multihop-cluster-based ieee 802.11 p and lte hybrid architecture for vanet safety message dissemination", *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 2621–2636, 2015.

[159] E. Yaacoub, F. Filali, and A. Abu-Dayya, "Qoe enhancement of svc video streaming over vehicular networks using cooperative lte/802.11 p communications", *IEEE journal of selected topics in signal processing*, vol. 9, no. 1, pp. 37–49, 2014.

[160] K. Yang, S. Ou, H.-H. Chen, and J. He, "A multihop peer-communication protocol with fairness guarantee for ieee 802.16-based vehicular networks", *IEEE Transactions on Vehicular Technology*, vol. 56, no. 6, pp. 3358–3370, 2007.

[161] K. Abboud and W. Zhuang, "Stochastic modeling of single-hop cluster stability in vehicular ad hoc networks", *IEEE Transactions on Vehicular Technology*, vol. 65, no. 1, pp. 226–240, 2016.

[162] ——, *Mobility Modeling for Vehicular Communication Networks*. Springer, 2015.

[163] F. Bai and B. Krishnamachari, "Spatio-temporal variations of vehicle traffic in vanets: Facts and implications", in *Proceedings of the sixth ACM international workshop on VehiculAr InterNETworking*, 2009, pp. 43–52.

[164] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges", *IEEE internet of things journal*, vol. 1, no. 4, pp. 289–299, 2014.

[165] W. Zhang, Y. Chen, Y. Yang, X. Wang, Y. Zhang, X. Hong, and G. Mao, "Multi-hop connectivity probability in infrastructure-based vehicular networks", *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 4, pp. 740–747, 2012.

[166] T. Mangel, T. Kosch, and H. Hartenstein, "A comparison of umts and lte for vehicular safety communication at intersections", in *2010 IEEE Vehicular Networking Conference*, IEEE, 2010, pp. 293–300.

[167] C. Sommer, S. Joerer, M. Segata, O. K. Tonguz, R. L. Cigno, and F. Dressler, "How shadowing hurts vehicular communications and how dynamic beaconing can help", *IEEE Transactions on Mobile Computing*, vol. 14, no. 7, pp. 1411–1421, 2014.

[168] K. Xu, K.-C. Wang, R. Amin, J. Martin, and R. Izard, "A fast cloud-based network selection scheme using coalition formation games in vehicular networks", *IEEE Transactions on Vehicular Technology*, vol. 64, no. 11, pp. 5327–5339, 2014.

[169] J. Manner, M. Kojo, T. Suihko, P. Eardley, and D. Wisely, "Mobility related terminology", IETF, RFC, Tech. Rep. 3753, 2004. [Online]. Available: `https://tools.ietf.org/html/rfc3753` (visited on 01/11/2020).

[170] S. Fernandes and A. Karmouch, "Vertical mobility management architectures in wireless networks: A comprehensive survey and future directions", *IEEE Communications Surveys & Tutorials*, vol. 14, no. 1, pp. 45–63, 2012.

[171] K. Zhu, D. Niyato, P. Wang, E. Hossain, and D. In Kim, "Mobility and handoff management in vehicular networks: A survey", *Wireless communications and mobile computing*, vol. 11, no. 4, pp. 459–476, 2011.

[172] I. L. d. MEDEIROS *et al.*, "Qoe and qos-aware handover for video transmission in heterogeneous vehicular networks", 2018.

[173] R. I. Meneguette, L. F. Bittencourt, and E. R. Madeira, "User-centric mobility management architecture for vehicular networks", in *International Conference on Mobile Networks and Management*, Springer, 2012, pp. 42–56.

[174] R. M. Abdullah and Z. A. Zukarnain, "Enhanced handover decision algorithm in heterogeneous wireless network", *Sensors*, vol. 17, no. 7, p. 1626, 2017.

[175] G. Oladosu, C. Tu, and P. A. Owolawi, "Optimized handover algorithm for vehicular ad hoc network", in *Proceedings of the 2019 7th International Conference on Information Technology: IoT and Smart City*, 2019, pp. 317–321.

[176] T. Higuchi and O. Altintas, "Interface selection in hybrid v2v communications: A hierarchical approach", in *2017 IEEE Vehicular Networking Conference (VNC)*, IEEE, 2017, pp. 9–16.

[177] M. Syfullah and J. M.-Y. Lim, "Data broadcasting on cloud-vanet for ieee 802.11 p and lte hybrid vanet architectures", in *2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT)*, IEEE, 2017, pp. 1–6.

[178] R. I. Meneguette, L. F. Bittencourt, and E. R. M. Madeira, "A seamless flow mobility management architecture for vehicular communication networks", *Journal of Communications and Networks*, vol. 15, no. 2, pp. 207–216, 2013.

[179] S. Gopinath, L. Wischhof, C. Ponikwar, and H. Hof, "Hybrid solutions for data dissemination in vehicular networks", in *2016 Wireless Days (WD)*, 2016, pp. 1–4. DOI: `10.1109/WD.2016.7461519`.

[180] M. C. Roman, "Heterogeneous parallel multi-radio transmission system in wireless vehicular communication", PhD thesis, Oxford Brookes University, Sep. 2016.

[181] B. Abidi, F. M. Moreno, M. E. Haziti, A. Hussein, A. A. Kaff, and D. M. Gomez, "Hybrid v2x communication approach using wifi and 4g connections", pp. 1–5, Dec. 2018. DOI: 10.1109/ICVES.2018.8519489.

[182] ETSI, "TR 103 576-2: Intelligent Transport Systems (ITS); Pre-standardization study on ITS architecture; Part 2: Interoperability among heterogeneous ITS systems and backward compatibility", v1.1.1, 2020.

[183] *MQTT For Sensor Networks (MQTT-SN) - Protocol Specification*, Nov. 2013. [Online]. Available: http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf (visited on 07/22/2019).

[184] *MQTT Version 3.1.1 - OASIS Standard*, Oct. 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf (visited on 07/22/2019).

[185] *MQTT Version 5.0 - OASIS Standard*, Mar. 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf (visited on 07/22/2019).

[186] PC Engines GmbH, *PC Engines - Apu3 series system board*, English, PC Engines, Mar. 2017, 10 pp.

[187] Compex Systems Pte Ltd, *Compex WLE200NX Dual Band 2x2 MIMO 802.11n Mini PCIe Module - Datasheet*, English, Compex, Jul. 2019, 4 pp. [Online]. Available: https://compex.com.sg/shop/wifi-module/wle200nx/.

[188] Atheros Communications, Inc, *Atheros AR9280 Single-chip 2.4/5 GHz 802.11n WLAN solution for PCI-Express - Datasheet*, English, Atheros, Jun. 2010, 2 pp.

[189] Huawei Technologies Co., Ltd., *HUAWEI ME909s Series LTE Mini PCIe Module - Hardware Guide*, English, version V100R001_07, Huawei, Dec. 2018, 87 pp.

[190] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception", 3rd Generation Partnership Project (3GPP), TS 36.101, release 8, Sep. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/36101.htm.

[191] *wireless-regdb*, Jun. 2016. [Online]. Available: https://wireless.wiki.kernel.org/en/developers/regulatory/wireless-regdb (visited on 02/25/2020).

[192] *Central Regulatory Domain Agent*, Mar. 2015. [Online]. Available: https://wireless.wiki.kernel.org/en/developers/regulatory/crda (visited on 02/25/2020).

[193] *ip(8) - Linux man page*. [Online]. Available: https://linux.die.net/man/8/ip (visited on 01/28/2020).

[194] I. USB Implementers Forum, "Universal Serial Bus Communications Class Subclass Specification for Mobile Broadband Interface Model", standard Revision 1.0 - Errata-1, May 2013. [Online]. Available: https://www.usb.org/document-library/mobile-broadband-interface-model-v10-errata-1-and-adopters-agreement.

[195] The Free Software Foundation - Aleksander Morgado, *Mobile Interface Broadband Model (MBIM)*. [Online]. Available: https://www.freedesktop.org/wiki/Software/libmbim/ (visited on 10/28/2019).

[196] *mbimcli (1) - Linux Man Pages*, Nov. 2019.

[197] *mbim-network (1) - Linux Man Pages*, Nov. 2019.

[198] "Ieee 802 numbers", IANA, Tech. Rep., 2020. [Online]. Available: https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml (visited on 01/09/2020).

[199] C. Benvenuti, *Understanding Linux Network Internals: Guided Tour to Networking on Linux*. O'Reilly Media, 2005, ISBN: 9780596552060. [Online]. Available: https://books.google.pt/books?id=yy7tihZLgGYC.

[200] M. Barbeau and E. Kranakis, *Principles of Ad-hoc Networking*. Wiley, 2007, ISBN: 9780470512487. [Online]. Available: https://books.google.pt/books?id=lk%5C_CJJhlMucC.

[201] *htons(3) - Linux man page*. [Online]. Available: https://linux.die.net/man/3/htons (visited on 01/28/2020).

[202] *packet(7) - Linux man page*, Nov. 2019. [Online]. Available: https://man7.org/linux/man-pages/man7/packet.7.html (visited on 10/28/2019).

[203] *setsockopt(2) - Linux man page*. [Online]. Available: https://man7.org/linux/man-pages/man2/setsockopt.2.html (visited on 01/28/2020).

[204] *getsockopt(2) - Linux man page*. [Online]. Available: https://man7.org/linux/man-pages/man2/getsockopt.2.html (visited on 01/28/2020).

[205] *netdevice(7) - Linux man page*. [Online]. Available: https://man7.org/linux/man-pages/man7/netdevice.7.html (visited on 01/28/2020).

[206] *ioctl(2) - Linux man page*. [Online]. Available: https://man7.org/linux/man-pages/man2/ioctl.2.html (visited on 01/28/2020).

[207] *bind(2) - Linux man page*. [Online]. Available: https://man7.org/linux/man-pages/man2/bind.2.html (visited on 01/28/2020).

[208] *sendto(2) - Linux man page*. [Online]. Available: https://man7.org/linux/man-pages/man2/sendto.2.html (visited on 01/28/2020).

[209] *recvfrom(2) - Linux man page*. [Online]. Available: https://man7.org/linux/man-pages/man2/recvfrom.2.html (visited on 01/28/2020).

[210] G. Roy, *RabbitMQ in Depth*. Manning Publications, 2017, ISBN: 9781617291005. [Online]. Available: https://books.google.pt/books?id=u1zgsgEACAAJ.

[211] A. Videla and J. Williams, *RabbitMQ in Action: Distributed Messaging for Everyone*. Manning Publications, 2012, ISBN: 9781935182979. [Online]. Available: https://books.google.pt/books?id=g3cyygAACAAJ.

[212] *What can RabbitMQ do for you?*, Jan. 2020. [Online]. Available: https://www.rabbitmq.com/features.html (visited on 01/22/2020).

[213] *RabbitMQ - Clients Libraries and Developer Tools*, Jan. 2020. [Online]. Available: https://www.rabbitmq.com/devtools.html (visited on 01/22/2020).

[214] L. Polyak, *Arch linux - rabbitmq 3.8.4-1*, https://www.archlinux.org/packages/community/any/rabbitmq/, 2020.

[215] Alan Antonuk, *Rabbitmq c amqp client library*, https://github.com/alanxz/rabbitmq-c, 2013.

[216] D. A. BUTTLAR, B. Nichols, D. Buttlar, J. Farrell, and J. Farrell, *Pthreads programming: A POSIX standard for better multiprocessing*. O'Reilly Media, Inc., 1996.

[217] S. Kleiman, D. Shah, and B. Smaalders, *Programming with threads*. Sun Soft Press Mountain View, 1996.

[218] *1003.1c-1995 - Standard for Information Technology–Portable Operating System Interface (POSIX(R)) - System Application Program Interface (API) Amendment 2: Threads Extension (C Language)*, 1995.

[219] *1003.1-2008 - IEEE Standard for Information Technology - Portable Operating System Interface (POSIX(R))*, 2008.

[220] *P1003.1 - Standard for Information Technology–Portable Operating System Interface (POSIX(R)) Base Specifications, Issue 8*, 1995.

[221] D. R. Butenhof, *Programming with POSIX threads*. Addison-Wesley Professional, 1997.

[222] B. Lewis and D. J. Berg, *Threads primer: a guide to multithreaded programming*. Prentice Hall Press, 1995.

[223] *clock_gettime(3) - Linux Man Pages*, Nov. 2019.

[224] *TIME(1) - Linux Man Pages*, Nov. 2019.

[225] *openvpn(8) - System Manager's Manual*, Nov. 2019.

[226] "Ieee standard for a precision clock synchronization protocol for networked measurement and control systems", *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–300, 2008.

[227] R. Cochran and C. Marinescu, "Design and implementation of a ptp clock infrastructure for the linux kernel", in *2010 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, IEEE, 2010, pp. 116–121.

[228] R. Cochran, C. Marinescu, and C. Riesch, "Synchronizing the linux system time to a ptp hardware clock", in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, IEEE, 2011, pp. 87–92.

[229] R. McMahon *et al.*, *Precision time protocol daemon*, 2011. [Online]. Available: `https://sourceforge.net/projects/linuxptp/` (visited on 03/17/2020).

[230] R. Cochran *et al.*, *The linux ptp project*, 2011. [Online]. Available: `https://sourceforge.net/projects/linuxptp/` (visited on 03/17/2020).

[231] P. Ohly, D. N. Lombard, and K. B. Stanton, "Hardware assisted precision time protocol. design and case study", in *LCI Intl. Conf. on High-Perf. Clustered Comp*, Citeseer, 2008.