Universidade de Aveiro
Ano 2017

Departamento de Eletrónica,
Telecomunicações e Informática

Tomás Marques Rodrigues

End-user quality of service and experience in mobile networks

Qualidade de serviço e experiência em redes móveis na ótica do utilizador final

Universidade de Aveiro
Ano 2017

Departamento de Eletrónica,
Telecomunicações e Informática

Tomás Marques Rodrigues

End-user quality of service and experience in mobile networks

Qualidade de serviço e experiência em redes móveis na ótica do utilizador final

"If your dreams do not scare you, they are not big enough"

– Ellen Johnson Sirleaf

Universidade de Aveiro
Ano 2017

Departamento de Eletrónica,
Telecomunicações e Informática

Tomás Marques Rodrigues

End-user quality of service and experience in mobile networks

Qualidade de serviço e de experiência em redes móveis na ótica do utilizador final

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor João Paulo Silva Barraca, professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor. Joaquim João Estrela Ribeiro Silvestre Madeira

Professor Auxiliar da Universidade de Aveiro

Vogais / examiners committee

Prof. Doutora Marília Pascoal Curado

Professora Auxiliar com Agregação da Universidade de Coimbra

Prof. Doutor João Paulo Silva Barraca

Professor Auxiliar da Universidade de Aveiro

agradecimentos

Ao meu orientador, João Paulo Barraca pela orientação e todo o apoio dado que levou a bom fim a escrita desta dissertação.

A toda a equipa da AlticeLabs associada ao projeto, e em especial ao meu co-orientador, Pedro Fernandes. Obrigado pela forma como me acolheram e me integraram no vosso grupo de trabalho. Foi uma ótima ajuda para que tudo corresse pelo melhor.

Aos meus pais e especialmente à minha irmã, o maior agradecimento. Para vos agradecer tudo o que já fizeram por mim eu precisava de uma tese inteira e não apenas de uma secção nos agradecimentos, vocês foram o meu pilar ao longo destes anos. Obrigado por terem sempre acreditado em mim e por me ajudarem a conseguir fazer sempre mais e melhor. Obrigado por me mostrarem que não interessam as notas ou os graus que eu consiga, que aquilo que interessa é eu seguir o meu sonho e ser feliz. Obrigado por todos os sacrifícios que já fizeram por mim. Obrigado por não me deixarem sentir mal mesmo quando as coisas correm menos bem e por toda a força no sentido de continuar. Obrigado por estarem sempre presentes. Muito, muito obrigado, sem vocês isto nunca teria sido possível.

À minha namorada, Joana e a todos os amigos, por toda a disponibilidade e por me apoiarem nas minhas derrotas e por festejarem comigo as minhas vitórias.

palavras-chave Qualidade de serviço, qualidade de experiência, rede móvel, aplicação móvel, Android SDK

resumo Os operadores de redes móveis recorrem a equipamentos dedicados (sondas) para obtenção de métricas relativas ao desempenho, QoS e QoE das suas redes e serviços. Pretende-se desenvolver uma aplicação Android com funcionalidades de sonda, não só complementando os equipamentos dedicados na recolha de informação relativa ao desempenho, QoS e QoE, como também detetando automaticamente problemas ao nível da rede e dos seus serviços no próprio terminal do cliente final, disponibilizando ferramentas de teste e suporte para uma resolução de problemas mais rápida e eficaz.

keywords Quality of service, quality of experience, mobile networks, mobile app, Android SDK

abstract Mobile network operators use dedicated equipment (probes) to obtain performance, QoS and QoE metrics for their networks and services. This project aims to develop an Android application with probing features, not only complementing the dedicated equipment in the collection of information regarding performance, QoS and QoE, but also detecting problems in the network and its services automatically in mobile network commercial terminal and providing customer test tools and support for faster troubleshooting.

Contents

List of figures

List of tables

List of snippets

Acronyms

2G	Second generation mobile networks
3G	Third generation mobile networks
3GPP	3rd Generation Partnership Project
4G	Fourth generation of wireless mobile telecommunications technology
5G	Fifth generation mobile networks (not yet standardized)
5GNOW	Fifth Generation Non-Orthogonal Waveforms for Asynchronous Signaling
5GPP	5G Infrastructure Public Private Partnership
ADB	Android Debug Bridge
API	Application Programming Interface
APN	Access Point Name
AuC	Authentication Center
AMPS	Advance Mobile Phone Service
AMR-WB	Adaptive Multi-Rate Wideband AMR-WB
BSC	Base Station Controller
BSS	Base Station Subsystem
BTS	Base Transceiver Station
BSSID	Basic Service Set Identifier
CO_2	Carbon Dioxide
C-RAN	Cloud Radio Access Network
CDMA	Code Division Multiple Access
CEPT	European Conference of Postal and Telecommunications Administrations
CSFB	Circuit Switched Fallback
CAPEX	Capital Expenditure
DL	Downlink
DNS	Domain Name System
DVB	Digital Video Broadcasting
DHCP	Dynamic Host Configuration Protocol
DMTF	Dual Tone – Multi-Frequency
eNB	EUTRAN Node B
EIR	Equipment Identity Register
EPC	Evolved Packet Core
EDGE	Enhanced Data rate for GSM Evolution
ETSI	European Telecommunications Standards Institute
EGPRS	Enhanced General Packet Radio Service
ESSID	Extended Service Set Identifier
EUTRA	Evolved Universal Terrestrial Radio Access
EUTRAN	Evolved Universal Terrestrial Radio Access Network
FTP	File Transfer Protocol
FDMA	Frequency Division Multiple Access
GSM	Global System for Mobile Communications
GGSN	Gateway GPRS Support Node
GSMA	Global System for Mobile Communications Association

GPRS	General Packet Radio Service
HD	High Definition
HLR	Home Location Register
HTTP	Hypertext Transfer Protocol
HSPA+	Evolved High-Speed Packet Access
HSCSD	High-Speed Circuit Switched Data
HSDPA	High-Speed Downlink Packet Access
HSUPA	High-Speed Uplink Packet Access
IP	Internet Protocol
IMS	IP Multimedia Service
IoT	Internet of Things
IPX	Internet Exchanges
ISP	Internet Service Provide
ITU	International Telecommunications Union
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IPv4	Internet Protocol version 4
ISDN	Integrated Services Digital Network
ICCID	Integrated Circuit Card Identifier
IMAP4	Internet Message Access Protocol
JPA	Java Persistence API
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
KQI	Key Quality Indicator
L2	Layer 2
L3	Layer 3
LAC	Location Area Code
LAN	Local Area Network
LTE	Long Term Evolution
ME	Mobile Equipment
MS	Mobile Station
M2M	Machine to Machine
MAC	Media Access Control
MCC	Mobile Country Code
MNC	Mobile Network Code
MMS	Multimedia Messaging Service
MOS	Mean Opinion Score
MSC	Mobile Switching Center
Mbps	Megabits per second
MGCP	Media Gateway Control Protocol
MIMO	Multiple Input Multiple Output
MMSC	Multimedia Messaging Service Center
NAT	Network Address Translation
NFC	Near Field Communication

NMT Nordic Mobile Telephone
NSS Network Switching Subsystem

OS Operating System
OEM Original Equipment Manufacturers

OTA Over the Air
OTT Over the Top
OPEX Operational Expenditure

P2P Peer-to-Peer
PCU Packet Controller Unit
PIN Personal Identification Number
PESQ Perceptual Evaluation of Speech Quality
PING Packet Internet Groper
POP3 Post Office Protocol
PLMN Public Land Mobile Network
PSTN Public Switched Telephone Network
POLQA Perceptual Objective Listening Quality Assessment

QoE Quality of Experience
QoS Quality of Service
QME Quality Management Element

RF Radio frequency
RATs Radio Access Technologies
RNC Radio Network Controller
RNS Radio Network Subsystem
ROM Read Only Memory
RTP Real Time Transport Protocol
RTT Round-Trip Time
RUU ROM Update Utility
REST Representational State Transfer
RSCP Received Signal Code Power
RSRQ Reference Signal Received Quality
RSRP Reference Signal Received Power
RTCP RTP Control Protocol

SAE System Architecture Evolution
SCP Secure Copy
SS7 Signaling System No. 7
SIM Subscriber Identity Module
SIP Session Initiation Protocol
SFTP SSH File Transfer Protocol
SGSN Serving GPRS Support Node
SMTP Simple Mail Transfer Protocol
SSID Service Set Identifier
SC-FDMA Single-carrier FDMA

TAC Tracking Area Code
TCP Transmission Control Protocol
TACS Total Access Communication System

TDMA	Time Division Multiple Access
UI	User Interface
UL	Uplink
UDP	User Datagram Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
UMTS	Universal Mobile Telecommunications System
VLR	Visitor Location Register
VoIP	Voice over IP (Internet Protocol)
VoLTE	Voice over LTE
VoWifi	Voice over Wi-Fi
WAN	Wide Area Network
WAP	Wireless Application Protocol
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WPAN	Wireless Personal Area Network
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WPAN	Wireless Personal Area Network
WCDMA	Wideband Code Division Multiple Access

Chapter 1

INTRODUCTION

Human relationships are based on communication and the technology is changing to improve the way we interact with each other. Telecommunications evolved from analog services to the digital, including not only voice but also data services with better throughputs, capacity and lower latency [4]. The requirement for higher data speed on smartphones is increasing rapidly, much due to the usage of social networks and other entertainment data in these small devices [15]. Constant improvement in wireless data rate is already happening and different network technologies are integrated to provide seamless connectivity and transparency to user, making the network appear heterogeneous despite the complexity involved.

User's expectations are always growing with new services appearing constantly and the quality of service needs to be a constant improvement in order to follow this technological evolution. Although the internet was designed to provide services without quality assurances, in the case of operators they are contractually obliged to ensure certain quality in some services provided by them and working with clients that more and more want to be always connected and with mobility. Due to this a lot of work has still to be done to grant good quality, performance and experience to the user.

1.1 Motivation

Technology is, more and more, part of the daily life of the human being and according to Global System Mobile Association (GSMA), the number of mobile connections already surpassed the number of people on earth and it's growing five times faster [1]. Humanity communicates on a global scale thanks to the increasing development of mobile devices technology. Computing and communication had led to a notorious evolution of mobile devices, which have become not only a mean of communication, but also a way of accessing extensive functionalities. The increased broadband speed let us view videos with high resolution or upload photos anywhere, making this small device, that fits in a pocket, the communication and entertainment tool of today's election.

The rapid growth of wireless communications allowed the rising number of smartphones and tablets with battery improvements and connected in a real-time, faster network. Evaluate the network behavior and what is happening is extremely important to the operator to assurance good quality of service to his clients and retain them. Given the importance of this, operators have fixed and mobile probes to try to give the best user experience and guarantee network availability and performance. This dissertation project aims to add a more flexible, transparent and dynamic solution to improve network service quality assessment with the increasing functionalities and technology on these small devices.

1.2 Objectives

The key objective of this project is to develop a solution that retrieves Quality of Service (QoS) metrics from the network, and radio parameters dependent on the access technology being used in the moment, with an Android smartphone to assess the user Quality of Experience (QoE). The resulting application is intended to run tests on the network in order to try troubleshoot possible problems. All the data gathered over time is supposed to be seen in a simpler and attractive user interface, either in a list and map view.

This solution is connected to a backend, sending all the data to a unified platform called ArQoS, a centralized and convergent product that evaluates the customer perceived quality in service usage (Voice, Internet Protocol Television (IPTV), Short Message Service (SMS), Multimedia Message Service (MMS), e-mail and Internet, to name a few), multi-technology and in multi-vendor environments in order to increase customer satisfaction and optimize resources in case of the operator [2].

There are a vast case of scenarios thought for this solution deployment. It can be used in drive tests through the city, continuous network and service monitoring, can be used by operator's technicians to identify problematic locations with poor coverage, real time problems troubleshooting and lastly, be used by a regular user to check internet connectivity or the downlink speed in the network at that moment, for example.

1.3 Contributions

This solution is part of a bigger product named ArQoS, which has mobile and fixed probes monitoring the network, producing diverse Key Performance Indicators (KPIs) and Key Quality Indicators (KQIs).

The ArQoS Pocket solution is another probe that runs on Android devices. This allows end-users to have an interface to report their service experience and perform tests at almost any time and place, which do not happens with the other probes that are completely autonomous. Additionally, because it will run on Android devices, it is expected to collect data in a larger scale, giving to the operator a more accurate performance and notion of what is happening with its network in the moment.

This solution also gains relevance in Voice over Wireless Fidelity (VoWi-Fi) or Voice over Long Term Evolution (VoLTE) tests, as these are functionalities already available in several commercial smartphones but, due to its immaturity, are not yet available in the embedded terminals that typically integrate dedicated probing equipment.

The application already had a dashboard page where the user could see the Wireless Fidelity (Wi-Fi) link speed or the mobile technology if using the mobile network. Furthermore, it is also shown in this page the signal strength level on both mobile and Wi-Fi networks. Some tests like Packet Internet Groper (PING) to the Google website, Hypertext Transfer Protocol (HTTP) Download/Upload and perform a portal login were also available for the user to execute in the application. Reports an encountered anomaly in the operator's network or services was another feature already implemented in the application. Lastly, it was already built a settings page with the option for the user to change the default app's homepage and language.

With the developments made throughout this dissertation, all the results are now locally persisted in a database and sent when possible to a backend system that gathers the information from all the ArQoS probes. New tests such as SMS, Voice, Send DTMF Tones and Scan Wi-Fi networks are now supported, allowing to actively test the network, performing intrusive tests and passively collect KPIs and KQIs from it. All the tests are now allowed to execute on-demand or in scheduled fashion, which are detailed in section 4.2.4, using a redesigned user interface (UI).

The solution has now connectivity with the ArQoS management system, delivering all the results, supporting notifications and requests already defined for other probes. Furthermore, the

application now launches at the device boot, being able to run scheduled tests even if the UI is not in foreground.

Lastly, in the settings page, new configurations are allowed for user to change, accordingly with its preferences and it was given support for automatically register radiologs, which are snapshots of the mobile network state.

1.4 Document Structure

This report is split into 6 chapters of which, chapter 1, Introduction, was already presented. The remaining chapters are:

- **Chapter 2:** presentation of the state of art. The core concepts of quality of service, experience and cellular networks are discussed in this chapter. Distinct mobile operating systems, as well as some new technologies like VoLTE and VoWi-Fi, are also presented in this chapter. Additionally, a detailed analysis of some solutions proposals relevant to the problem are presented;

- **Chapter 3:** identification of main objectives and requirements defined for the solution. Is also given an overview of the ArQoS global product and the ArQoS Pocket solution main goals. Contributions of all stakeholders involved in the solution progression and lastly, the solution architecture.

- **Chapter 4:** description of the implemented solution and internal architecture along with the used frameworks and a detailed explanation of the followed approach during the implementation;

- **Chapter 5:** presentation and evaluation of the obtained results, as well of insights in their regard. A description of the test methodology is described and the objectives and test results are then analyzed.

- **Chapter 6:** final conclusions about the chosen path and obtained solution, also addressing potential improvements for future work.

Chapter 2

State of Art

This chapter provides a review of the main concepts to better understand the solution development direction. It is presented all the research, focused mainly on the mobile networks and the mobile operating systems.

Each presented section aims at providing a deeper theoretical knowledge on a certain subject or to better expose fields or features and how they can be combined in our solution.

2.1 Mobile Network Evolution

Mobile networking is a technology that supports voice and data using radio transmission. In the past, these communications used circuit switching to carry voice over a network, but nowadays both data and voice are transmitted over circuit-switched and packet-switched networks. The use of wireless communications in mobile communications started in the 1970s with the zeroth generation. Today we have reached the fourth generation and several countries are working on the architecture and development of 5G. This section will give an overview of this evolution over all these years. [25]

The pre-cellular system was the first mobile communication technology, it worked using a central antenna mounted per region and strong transmitters and receivers were used to send and receive the data. Preceding cellular mobile telephony technology these systems are also known as **0G** (zero generation) [3].

Based on analog transmission the first mobile systems emerged, later known as **1G**. Using multiple cell sites and having the ability to transfer calls as the user traveled, the first-generation wireless signal established seamless mobile connectivity introducing mobile voice services.

Mobile phones at the time were not as we know them today. They were extremely large, heavy, not comfortable to carry, power inefficient and had high costs for the standards at that time. They used Frequency Division Multiple Access (FDMA) for spectrum sharing, as depicted in figure 2.1, but a large gap of spectrum was required between users to avoid interference. Other major

problems associated to this technology were low traffic density of one call per radio, limited services, calls susceptible to noise, low data rates, inadequate fraud protection and poor security, as anyone with a radio scanner could eavesdrop your unencrypted calls. Perhaps connections to a tower miles away was, at that time, really impressive [4].

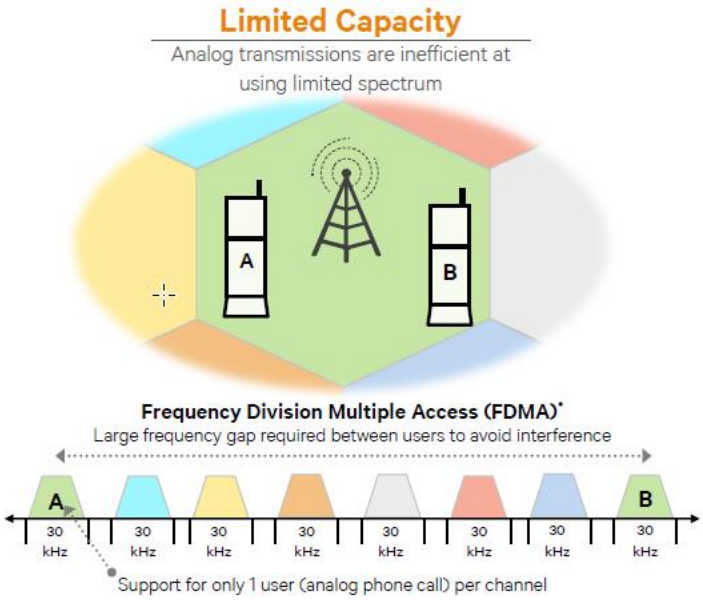


Figure 2.1 - FDMA based on AMPS 1G technology [5]

In the 1990's, the **'second generation' 2G** mobile phone systems appeared, the first digital cellular network, larger and better than 1G. The initial requirements for this network defined by the European Conference of Postal and Telecommunications Administrations and the European Telecommunications Standards Institute (CEPT/ETSI) were: more efficiency on the radio frequency usage, QoS, Numbering ITU-T and signaling protocol in the network.

This generation introduced the Global System for Mobile Communications (GSM) architecture in 1991, enabling more users per channel with the Time Division Multiple Access (TDMA) technique, but still required a large gap of spectrum between users to avoid interference. This architecture can be seen in figure 2.2 and it's divided into three parts: Mobile Station (MS), Base Station Subsystem (BSS) and Network Switching Subsystem (NSS). The first one consists of Mobile Equipment (ME) and a Subscriber Identity Module (SIM) containing the subscriber identity, a password (PIN), subscription information such as last received/dialed numbers, last visited location area and more commonly used for authentication and other security procedures.

The BSS is composed by a base station controller (BSC) and one or more base transceiver station (BTS), placed in the center of a geographic area covered by a base station and maps transceivers.

The cell size is defined based on the BTSs transmitting power. BSC manages radio resources of the BTSs and handles the BTSs and the MS power control, handovers, channel allocation and it also gathers the traffic towards the MSC [6].

At last the NSS is composed by the Mobile Switching Center (MSC) containing four databases: Home Location Register (HLR), Visitor Location Register (VLR), Authentication Center (AuC) and Equipment Identity Register (EiR) and the Gateway MSC. The MSC manages the communication between the mobiles and the fixed network, handles authentication and registration from connections with subsystem databases. The HLR maintains permanent information about the subscribers of a GSM network and tracks the location and state of the mobile terminal within the network, the VLR maintains temporary information about the subscribers registered on a GSM network and keeps up-to-date information about the location of the user within the network, the AuC is responsible for the authentication of the subscribers, maintains the encryption algorithms, the secret key for each subscriber and generates the session keys. Finally, the EiR provides security mechanisms and keeps a list of the authorized/blocked mobile equipments.

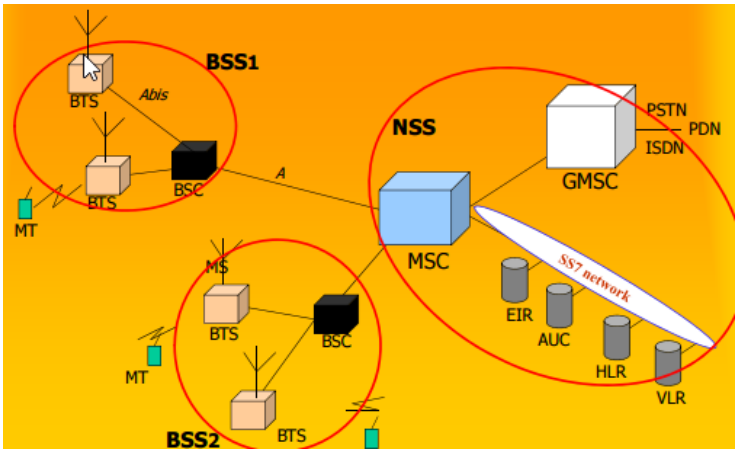


Figure 2.2 - GSM architecture [8]

The second generation introduced the SMS text, a new way to communicate and an alternative over voice calls, nowadays preferred in many situations. In comparison with 1G digital signals, this technology consumes less battery power from mobile phones making mobile batteries last longer. Voice clarity was improved, compressed into smaller "packages", and line noise was reduced. Furthermore, the cost/weight of digital components reduced a lot making possible to have a pocket-sized device with more security and with encryption for the data in voice calls.

Mobile phone usage increased drastically and the first's prepaid mobile phones began to appear [5]. However, although the network offers more capacity, the data was mostly just text messages,

and the delivery was rigid whether the user was or not actively talking. These constraints, as well as the 2G low air interface data rates were improved by High-Speed Circuit Switched Data (HSCSD) and the General Packet Radio Services (GPRS).

HSCSD uses multiple time slots per user (up to 6) and increased data throughput. However, since it is circuit switched, it allocates the time slots even when nothing is being transmitted, thus increasing the blocking probability of the system.

On another front, GPRS was the new technology that made possible to make phone calls and transmit data simultaneously. Its architecture provides both circuits switched and packet switched data for voice and data traffic, respectively. This period is seen as an extra period of mobile networking called **2.5G where new** network applications appeared, as well as new services such as the Wireless Application Protocol (WAP) access, Multimedia Messaging Service (MMS) and e-mails. Bit rates improved from 56 Kbit/s up to 115 Kbit/s and new user-oriented billing mechanisms (e.g by traffic volume), which were an important step towards 3G and for the first evolution of GSM networks [4].

In comparison with the GSM architecture, in GPRS the BSSs evolved due to upgrades on BTSs, BSCs and better network planning. A new element called Packet Controller Unit (PCU) that manages packets toward the radio interface was also added. At the NSS the core network was modified, adding new packet nodes dedicated to GPRS (SGSN, GGSN) and Internet equipment like Domain Name System (DNS) servers and firewalls. This evolution and the upgrades associated are shown below, in figure 2.3.

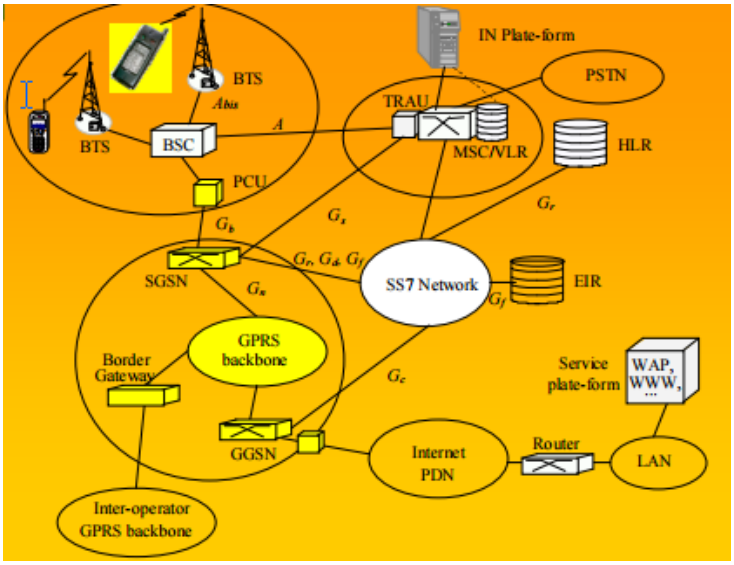


Figure 2.3 - GPRS architecture [8]

With no new technology required, a way was found to double the transfer speed, although was not fast enough to reach the 3G standards defined for the future. The idea was to change the modulation scheme to 8-PSK. This technology was called Enhanced Data for GSM Evolution (EDGE) or Enhanced General Packet Radio Service (EGPRS) an extended version of GSM.

Operators and providers revenues were coming mainly from voice (>70%) and the mindset was to migrate towards more applications, services and new systems [5]. Technological progress and voice/data combination contributed as well to motivate a new system. Taking that in consideration European Telecommunications Standards Institute (ETSI) started the work, carried out later by Third Generation Partnership Project (3GPP) organization for the next generation mobile networks in a system called Universal Mobile Telecommunication System (UMTS).

Combining aspects of the 2G technology the 3GPP model idea, shown in figure 2.4, is transformed from a vertical model to a horizontal one, in which the service and transport layers are independent of the access technology used. This allowed to create a single set of services, generic application servers, a common session control, a consistent user experience and improve operational efficiency to any device anywhere and in any access technology.

The core network of UMTS has base stations called now Node B and Radio Network Controllers (RNC). These components together form the Radio Network Subsystem (RNS) similarly to BSS that was seen before.

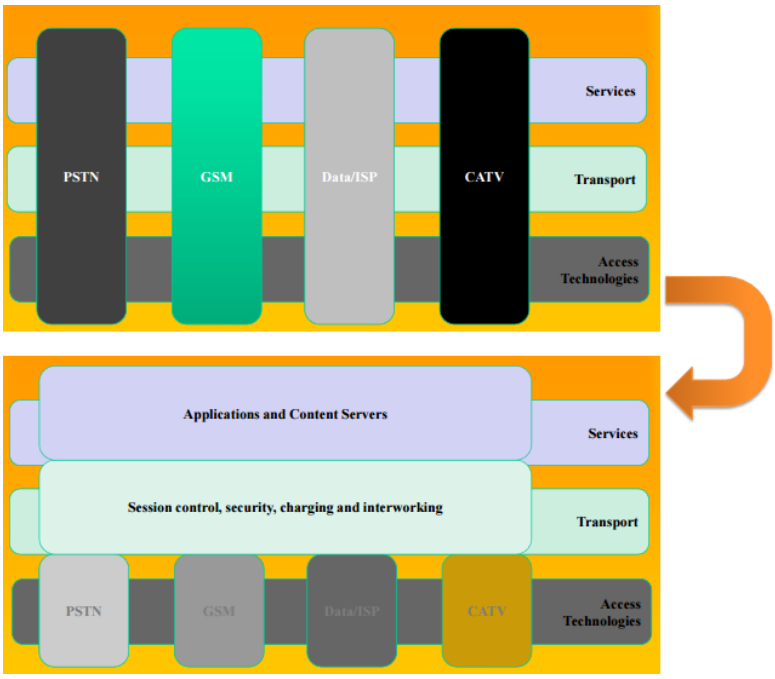


Figure 2.4 - Legacy Telecom Model and 3GPP Telecom Model respectively [8]

In release 5, IMS was introduced with the same core network and was the architectural foundation for next generation networks. It defined a common framework for delivering IP-based multimedia while maintaining QoS, billing and service integration. Release 6 extended to wideband fixed networks (xDSL, WLAN, cable, etc.) and supported services convergence on fixed and mobile networks (circuit-switched voice traffic in IP). For network providers this brings a lot of advantages because a universal network reduces operational expenditures and service providers reach a new and broader market with functions of authentication, charging and billing that could now, be outsourced [21]. The figure 2.5 shows all entities responsible for handling the functions referred and the associated architecture complexity.

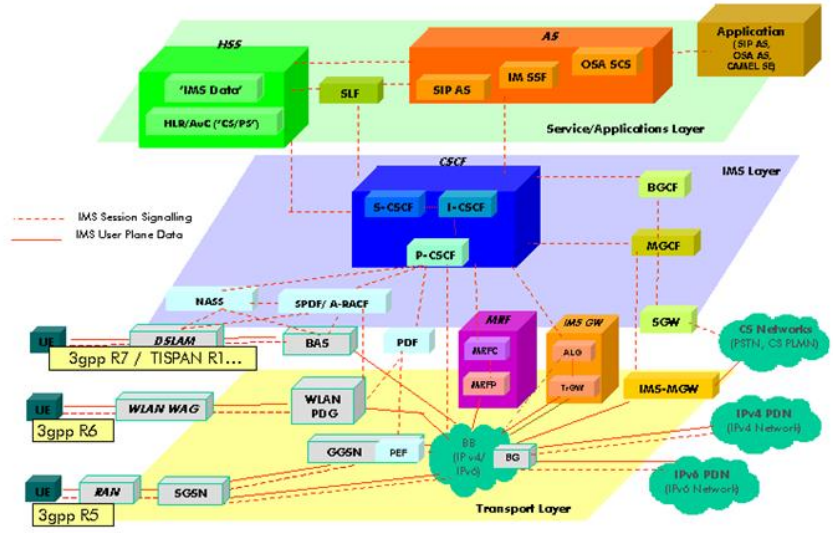


Figure 2.5 - 3GPP IMS architectural overview

Before the fourth generation, some sources consider an extra period, **3.5G** or **3G**+ where mobile telephony protocols like HSDPA and HSUPA provided a smooth evolution of data rates of 3G networks. Complimentary to each other, HSDPA is a packet-based data service in WCDMA downlink, whereas HSUPA is a UMTS / WCDMA uplink evolution technology that will enhance P2P data apps, e-mail apps, games and many others, due to this boost in the link speed.

The successor of 3G UMTS, **4G,** was developed by 3GPP, and is known as Long Term Evolution(LTE). This 4G network must deliver speeds of 100Mbps while moving, 1Gbps while stationary and smooth handover going from 4G to Wi-Fi and back with no data or call interruptions. It utilizes packet switching over IP, Local Area Network (LAN) or wide area network (WAN) networks. The improvements on speed, reliability, capability, security and global mobility are also important upgrades from 3G technology.

Also known for the high-quality audio and video streaming and for provide better latency and throughput, this network merges various radio access interfaces into a single network that can be accessed by every subscriber.

The quality of service was one of many targets, in video chat, mobile TV, Digital Video Broadcasting (DVB) and many other services that make use of bandwidth. Other targets include improved latency, spectrum efficiency, radio frequency (RF) coverage, reduce capital expenditure (CAPEX), operational expenditure (OPEX) and offer compatibility with earlier releases and systems (including handover) providing a steady experience to the user [23].

LTE radio access network is called Evolved Universal Terrestrial Radio Access Network (EUTRAN), it also has an air interface called Evolved Universal Terrestrial Radio Access (EUTRA). The entire core network is known as Evolved Packet Core (EPC) and it provides IP connectivity between the user equipment and an external packet data network using EUTRAN. Base stations are called EUTRAN Node B (eNB) and they take care of access control functions. Also, about the radio interface, it uses Orthogonal Frequency Division Multiple Access (OFDMA), a multiple access technology on the downlink that transmits traffic across hundreds of parallel radio connections known as 'subcarriers'. An adapted version of this is Single-carrier FDMA (SC-FDMA) used on the uplink [7].

Mobile 4G LTE complements 3G providing more data capacity for richer content. Additionally, connections with higher throughput, services and billing systems are also key aspects for the success of the 4G systems. The figure 2.6 illustrates the mobile network evolution from its inception until now.

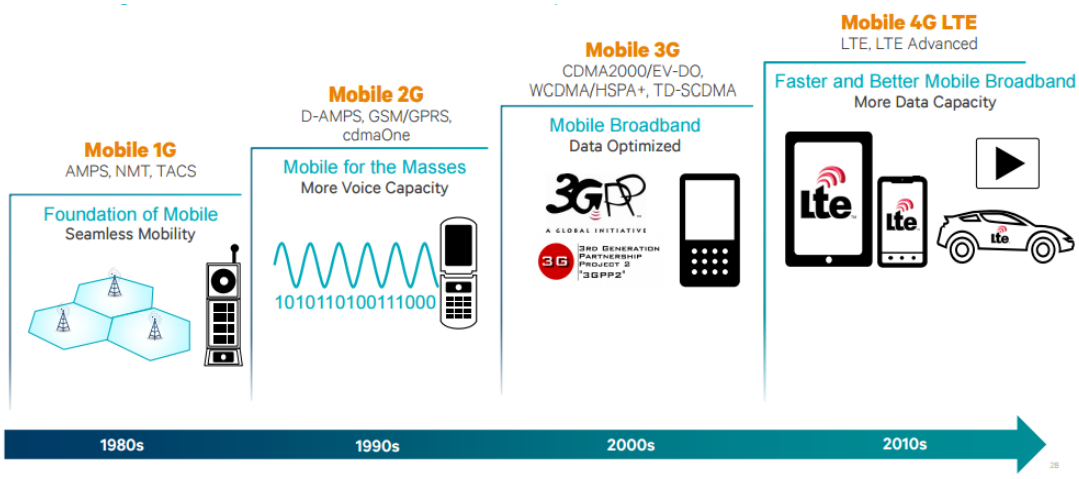


Figure 2.6 - Mobile technology evolution [5]

In 2020 it is expected that **5G** mobile technology will be fully operational in a connected society. Different research groups like Mobile and Wireless Communications Enablers for the Twenty–twenty Information Society (METIS), 5th Generation Non-Orthogonal Waveforms for Asynchronous Signaling (5GNOW), 5G Public-Private Partnership (5GPPP) and many others organizations from America, China, Japan, Korea, Russia and others around the world are actively working on different aspects and areas of 5G. Appendix A has a more detailed view of those projects and activities that are being developed all over the world.

For socio-technical requirements 5G has limits on energy dissipation, more context-related information (e.g. augmented reality), broadband Internet connectivity widely available, increased amount of remote virtual collaboration, more efficient, safer transportation means, personal data stored in the cloud and improvements on IoT (e.g. smart homes and cities) with projects to society, as shown in figure 2.7.

There are currently limitations in 4G networks already like latency, spectral efficiency and support for bursty data traffic. Several mobile applications require sending messages to servers with high data transfer rate for a very short time that still consumes a lot of the mobile equipment's battery [16]. Co-channel interference due to separate channels (DL and UL) in a typical cellular network and base-station utilization efficiency are also points where improvements can be done on future networks [17].

With that said, with 5G networks the numbers of devices like smartphones, TVs, cameras, robots, video surveillance systems, high-resolution TVs, laptops and wearable devices (watches and glasses) connected is expected to continue to grow. The future networks will focus on scalability, high data rates, spectrum usage, low latencies that will supposedly increase the level of QoS and QoE with more satisfaction from users, which will be discussed in sections 2.3 and 2.4. Last but certainly not least, an ubiquitous connectivity, because operating bands are not identical over the globe and it is needed to support a variety of radios and Radio Access Technologies (RATs) [15, 18].

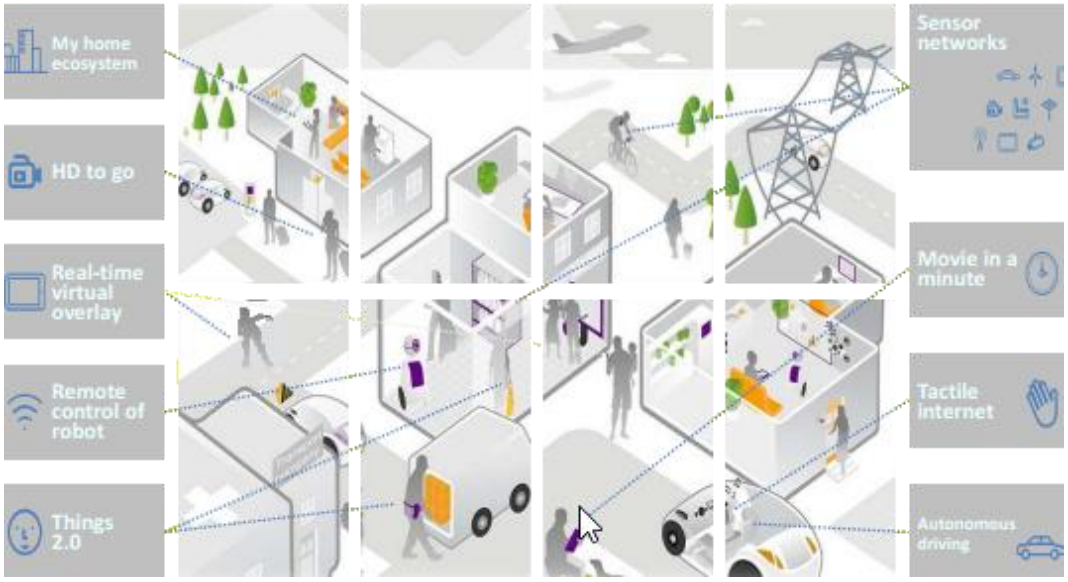


Figure 2.7 - 2020+ experience [8]

In summary 5G networks will be more flexible with a lot of users requesting for different set of services simultaneously, having higher data rates, higher capacity, energy optimization, mobility, availability, reliability, connectivity, cost effectiveness and less latency improving areas like IoT, gaming, personal clouds, and creating opportunities for new applications for many areas of society like tourism, agriculture, e-government and e-health. However, 5G is still facing many challenges such as security, low energy impositions, structure management with self-healing to perform dynamic operations, handoff's in very high speed vehicles and of course, more devices connected brings more and new vulnerabilities. Table 2.1, shown below, compares all generations of the mobile networks previously referred in different aspects and the technologies used throughout its evolution, plus the 5G expectations.

Generations	Access Technology		Data Rate	Frequency Band	Bandwith	Forward Error Correction	Switching	Applications
IG	Advanced Mobile Phone Service (AMPS) (Frequency Division Multiple Access (FDMA0		2.4 kbps	800 MHz	30 KHz	NA	Circuit	Voice
2G	Global Systems tor Mobile communications (GSM) (Time Division Multiple Access (TDM At)		10 kbps	850/900/180 0/1900 MHz	200 KHz	NA	Circuit	Voice + Data
	Code Division Multiple Access (CDMA)		10 kbps		1.25 MHz			
2.50	General Packet Radio Service (GPRS)		50 kbps		200 KHz		Circuit/	
	Enhanced Data Rate for OSM Evolution (EDGE)		200 kbps		200 KHz		Packet	
30	Wideband Code Division Multiple Access (WCDMA) *t* Universal Mobile Telecommunications Systems (UMTS)		3K4 kbps	800/850/900/ 1800/1900/ 2100 MHz	5 MHz	Turbo Codes	Circuit/ Packet	Voice + Data + Video calling
	Code Division Multiple Access (CDMA) 2000		384 kbps		1.25 MHz		Circuit/ Packet	
3.50	High Speed Uplink / Downlink Packet Access (HSUPA / HSDPA)		5-30 Mbps		5 MHz		Packet	
	Evolution-Data Optimized (EVDO)		5-30 Mbps		1.25 MHz		Packet	
3.75G	Long Term Evolution (LTE) (Orthogonal / Single Carrier Frequency Division Multiple Access) (OFDM A / SC-FDMA)		100-200 Mbps	1.8GHz. 2.6GHz	1.4 MHz to 20 MHz	Concatenated codes	Packet	Online gaming + High Definition television
	Worldwide Interoperability for Microwave Access (WIM AX X Scalable Orthogonal Frequency I»vision Multiple Access) SOFDM A »	Fixed WIM AX	100-200 Mbps	3.5GHz. and 5.8GHz initially	3.5MHz and 7MHz in 3.5GHz band; 10 MHz 5.8GHz band			
4G	Long Term Evolution Advanced (LTE-A) (Orthogonal / Single Carrier Frequency Division Multiple Access) (OFDMA / SC- FDMA)		DL 3Gbjw UL l.5Gbps	1.8GHz, 2 6GHz	1.4 MHz to 20 MHz	Turbo codes	Packet	Online gaming + High Definition
	Worldwide Interoperability for Microwave Access (WIM AX X Scalable Orthogonal Frequency-Division Multiple Access) SOFDMA))	Mobile WIM AX	100-200 Mbps	2.3GHz. 2.5GHz. and 3.5GHz initially	3.5MHz. 7MHz. 5MHz. 10MHz. and 8.75MHz initially			Television
5G	Beam Division Multiple Access (BDMA) and Non- and quasi-orthogonal or Filter Bank multi carrier (FBMC) multiple access		10-50 Gbps (expected)	1.8.2.6 GHz and expected 30-300 GHz	60 GHz	Low Density Parity Check Codes tl.DPC)	Packet	Ultra High definition video + Virtual Reality applications

Table 2.1 - Evolution of wireless technologies [25]

2.2 VoIP

When a user establishes an Internet connection, a public or private IP address is given to the device. IP is a protocol that allows sending information, in the form of small packets, from one host to another.

Voice over IP (VoIP) allows the user to establish telephone calls over a data network, such as the Internet, converting an analog voice signal into a set of digital signals in the form of packets with IP addressing. Since it can be used on any network that uses IP, the usage of this communication protocol has increased in the recent years, enabling customers to use their Internet connection to make phone calls [10].

Some popular VoIP protocols are H.323, Session Initiation Protocol (SIP) and Media Gateway Control Protocols (MGCP). H.323 is an International Telecommunications Union (ITU-T's) standard that combines a family of protocols such as, H.225 for call registration, H.245 for establish and control the media sessions, H.26x for defining the video codec, among others [57]. The audio, video and registration packets are usually transported over User Datagram Protocol (UDP) while the data and control application packets use Transmission Control Protocol (TCP). Furthermore, H.323 uses Real Time Transport Protocol (RTP) for media transport and RTP Control Protocol (RTCP) for control the RTP sessions [57, 58].

SIP is another protocol used for signaling and controlling multimedia communication sessions. A SIP messages contain a header and a body, and can be transported either by UDP or TCP. This protocol is modular, which means that it can be changed accordingly to the type of data that is needed to transmit. Another major strength of SIP is the possibility to initiating interactive user session with video, voice, instant messaging among others. For the reasons described, SIP offers a simpler alternative for communication than the H.323 family of protocols [57].

MGCP is a complementary protocol to the previously mentioned SIP and H.323. This protocol is used to control telephony gateways called Media Gateway Control (MGC) or "call agent". This element acts as master, supporting services providers and managing the calls. The Media Gateways (MGs) act as slave, receiving and executing commands from the MGC [59].

The quality of service perceived by the user depends not only, but primarily on three aspects: network topology, interconnection congestion and used codecs. Packets are not delivered instantaneously and this delay, can affect the quality dependent of the number of nodes between

the two points in communication. Interconnection congestion due to other active calls, already established on the network, can affect the quality of the service. Lastly, the used codecs, since voice signal compression can deteriorate or delay the communication. On the other hand, some codecs that usually require more computational processing have a corrective effect on transmission problems, minimizing the impact of packet loss. Other metrics and parameters, which influence the QoS, not only in VoIP, but also in other protocols will be detailed in section 2.3, Quality of Service.

One of the major objectives of VoIP is to transport quality voice traffic over IP networks. As already mentioned, these voice signals are digitized and sent over the network as packets. As bandwidth is not an infinite resource, there have been made significant efforts towards the minimization of the amount of bandwidth needed by VoIP services.

Compressing the voice signals and maintaining an acceptable level of quality by users it's still a challenge. Voice codecs are algorithms that enable this process, so choosing a suitable codec is extremely important because, codecs that allow higher audio quality, usually also requires a lot of bandwidth, making the network less efficient. Although of the demand for high-quality voice, operators tend to support at least one codec with a lower bitrate to have flexibility between quality and bandwidth [60]. Some considerations to be made are the codec compression rate, the delay that it adds to the connection and the compression type. The table 2.2 shows a comparison of these characteristics for some well-known voice codecs.

Codec	Type	Bitrate (kb/s)	Frame size (ms)	Bits per frame	Codec delay (ms)	Compression type
G.711	Narrowband	64	0,125	8	0,25	PCM
G.723.1	Narrowband	6,3/5,3	30	189/159	67,5	MP-MLQ/ACELP
G.729	Narrowband	6.4	10	64	25	CS-ACELP
GSM-FR	Narrowband	13	20	260	40	RPE-LTP
AMR-WB	Broadband	6,6-23,85	20	132-477	45	ACELP
G.722.1	Broadband	24-32	20	480/640	60	MLT

Table 2.2 - Well-known voice codecs characteristics [60]

2.2.1 VoLTE

LTE network architecture was already explained in section 2.1 and VoLTE is based on the IP Multimedia Subsystem (IMS), referred in the same section, with its own profiles for media and

control planes well defined by GSMA. Using the IMS central network means that the LTE architecture is being used with no dependency on legacy circuit-switched network to carry calls. Being an IP-based data connection allows VoLTE to have three times more voice and data capacity than 3G UMTS [40].

However, being a new technology VoLTE has a couple of challenges due to the multiplicity of protocols, technologies and implementation scenarios. In the situation of an international call with roaming, the SIP call signaling and voice path are not essentially the same because signaling could go through one or more Internet Exchanges (IPX). Also “the standards for voice facilities over LTE on 3GPP IMS are still growing" [42] and carriers need to re-engineer their voice call network. Lastly, there is a user quality of experience challenge, in the sense that VoLTE needs to deal with the fact that a user can leave the LTE area coverage during a call and then make the call go via legacy network using the Circuit Switched Fallback (CSFB) technique [40, 42].

It is needed to point out that VoLTE implementations are not yet interoperable, as the technology is new and still needs development to reach an ideal level of quality. However, when the challenges, previously mentioned, are handled there will be calls with fast connections, enabling High Definition (HD) Voice since the audio compression by VoLTE can reach the 13kbps and use the 700MHz band, contributing to more clean audio calls over long distances. Other benefits that can come are the possibility of operators adding to their portfolio extra service packages, containing, for example, video calls, file transfers, attachments or automatic translations. In addition, if we compare VoLTE with OTT apps like Skype the battery consumption level of the device will be lower on VoLTE service [40].

To finish this topic, there is a mobile network operator in India called Reliance Jio Infocomm Limited that doesn’t have legacy network support for 2G/3G services and provides only wireless 4G LTE service network, offering VoLTE services to all those who have an Android 4.0 or higher compatible VoLTE device with their own SIM card, the Jio SIM Card [41].

2.2.2 VoWiFi

To increase network capacity and extend their voice services, carriers are likely to use VoWiFi to extend coverage particularly indoors. VoWiFi is a solution where mobile services providers can deliver “the same set of mobile voice and messaging services that they currently offer over their macro cellular network and Wi-Fi networks” [43].

Such as VoLTE, this technology is defined by mobile industry standards organizations (3GPP and GSMA) and will allow a transparent transaction between the cellular network to any home, office or public Wi-Fi network. VoWiFi can really shine in helping with indoor mobile calls coverage. Providing good coverage in some scenarios is more complex and expensive than one way think, especially in lower floors or in internal rooms.

Apart from extending reach, VoWifi can potential save millions of dollars to some operators as calls placed on a smartphone would be carried over the consumer's broadband network. This enables traffic to be off-loaded to another network and freeing up some cellular capacity [44].

The control of subscribers is still completely retained by operator when a VoWiFi-enabled subscriber connects trough Wi-Fi. As shown in figure 2.8, they attempt to connect to operator's core network and register to receive the service. Once authorized, traffic is routed over the Wi-Fi/Internet connection instead of the macro cellular network, but the operator continues handling all billing, charging and routing.

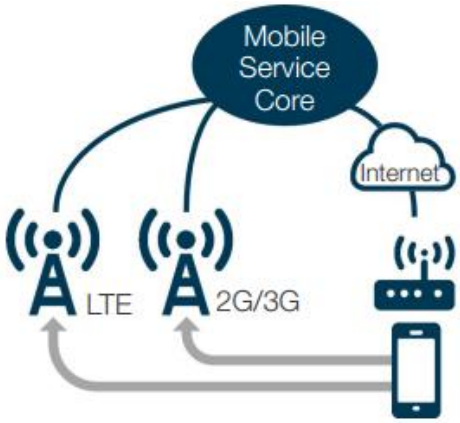


Figure 2.8 - Calls over Wi-Fi and cellular network [43]

VoWiFi service quality is expected to be as good as voice quality over the cellular network since the connections are over the internet, and typically the Wi-Fi networks at home and office have a good throughput and coverage overall. However, in public Wi-Fi networks those same factors are variable and can affect the VoWiFi quality. Another issue to be considered is that carriers should also bear the cost implications for incorporating emergency service support to ensure that these calls go through reaching the destination [43].

Still, there is a cost associated with the deployment of an IP multimedia subsystem, but if operators already have VoLTE, this has already been paid for, as the same IMS core network

supports both services, and there is no doubt that VoWiFi will make OTT apps like Skype, Viber or WhatsUp be less relevant for user as long as they have Wi-Fi indoor coverage which is a big win for operators. All this advantages for the operator explained through this section can be shown in figure 2.9.

Figure 2.9 – VoWiFi deployment reasons [43]

2.3 Quality of Service

The QoS for networks is a set of mechanisms, standards and protocols for ensuring high-quality performance and granting end-to-end consistent treatment of data flow as it travels through the network. Taking this into account, the network can then be managed using the existing resources efficiently to ensure a QoS level and improve the final user's global experience using the service.

Treating all the network traffic equally leads to no guarantees for reliability, delay and other performance parameters, which will be detailed later in this section. For some services or applications, QoS is an important requirement and it can be even critical if some traffic needs to have a preferential treatment.

For the operator measuring QoS is extremely important in order to grant levels of quality in services that they offer. Therefore, operator have all the interests on measure quality parameters in scenarios experienced by users to improve the service availability, reliability and performance.

The QoS performance on VoIP services is briefly described by metrics, such as latency (delay in data transmission from source to destination), jitter (delay variation), packet loss rate and throughput [58]. Additionally, the resource reservation and service prioritization can help improve these parameters.

Considering VoIP voice services only, delay can be described as the time between the instant one starts to say a word and the instant the remote party start to hear ispeaker. It can be divided in source delay, receiver delay and network delay. ITU-T recommends that this time don't exceed the 150ms [61]. Jitter is the delay variation of packet's delivery time. Jitter variation acceptable times are lower than 100ms on an IP network. Larger values can cause loss of voice packets [61]. Packet loss is the loss of packets that may be lost or arrive too late be played back. Finally, throughput is the maximum amount of information that can be received or processed, during a certain interval of time. In VoIP, it depends on the codecs used and on the number of users.

QoS provision in 4G networks is a challenge because the network "supports varying bit rates from multiple users, a variety of applications, hostile channel characteristics, bandwidth allocation, fault-tolerance levels and a frequent handoff among heterogeneous wireless networks" [19]. Different applications can depend primarily on different parameters, for example, an application that is delay sensitive requires the QoS to have delay guarantees, but other application may require that packets are delivered reliably from the source to destination, needing reliability. Other metrics that describe the QoS performance in cellular networks are jitter, service/network availability, bandwidth (the rate at which traffic is carried by the network), throughput and packet loss [62].

QoS support can exist at the transport, application, network, user and switching levels, and it can be applied per flow (individual, unidirectional streams) or per aggregate (two or more flows having something in common) [19].

A lot of research for QoS provision on cellular networks is being made due to the challenges mentioned above. On 4G network these issues are even more challenging, because QoS and signaling protocols may handle voice over IP in 4G networks, but with the exponential increase in data traffic it is needed, more and more, traffic engineering on IP [63].

In section 2.1, it was already seen that QoS improvement in 5G networks is a must. Based on the projections, the number of machine to machine (M2M) connections on the networks of mobile operators will surpass the 20.000.000.000 connections [39]. Two times more than the present number and in 2022 mobile operators will have more than 26.000.000.000 machine to machine connections.

Some general trends related to 5G can be explained in terms of machine to machine traffic and number of machine to machine connections in mobile. These huge numbers can be seen in figure

2.10 at left and right, respectively. Based on this continuous increase of devices connected on the network the data rate, delay bound, cell spectral efficiency and latency in 5G needs to be improved and that is supposed to be attained using non-orthogonal access methods in radio access networks [38, 67].

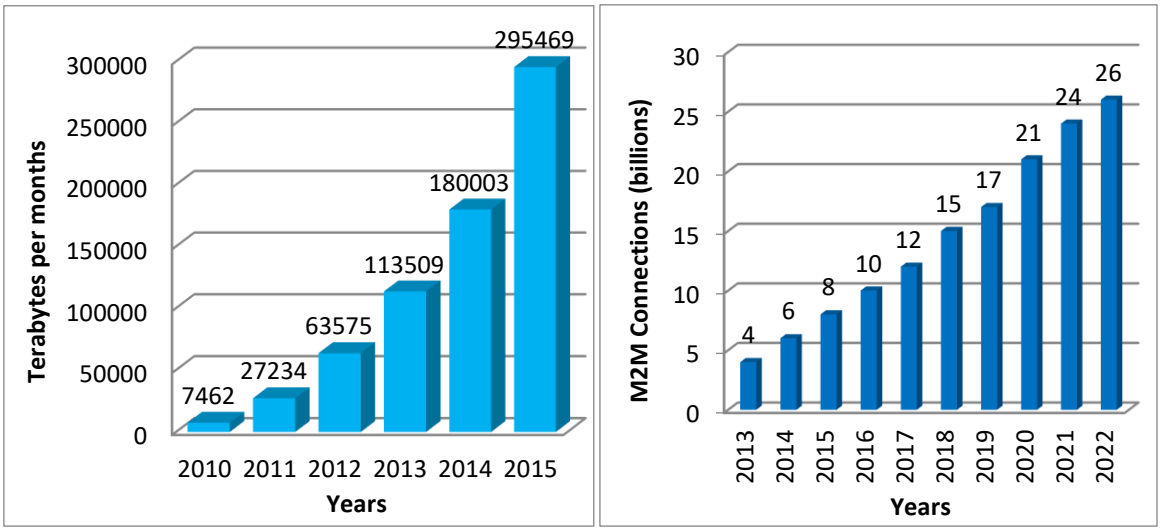


Figure 2.10 - Machine to machine traffic and connections number respectively [67]

A big issue about this topic is the tactile Internet as it requires the best QoS, especially latency at the order of 1ms for senses, such as touching, seeing, and hearing objects far away, as precise as human perception. However, the currently proposed architectures do not support efficient tactile Internet services. In future, it would be a promising area as to encode senses, exchange data satisfying the zero latency, and enable the user to receive the sensation [37]. Lastly, for cloud, it is proposed the deployment of a quality management element (QME) in the cloud for monitoring inter-UEs and inter-layer (the control and the data layers in C-RANs) QoS.

2.4 Quality of Experience

User's satisfaction is the primary goal of a service provider. QoE describes the user's perception of a service and how well the service fulfills the user's expectations. According to [49] it is "a user-centric metric that captures the overall acceptability of the service and includes the end-to-end factors".

Contrary to previously mentioned QoS, QoE is a more subjective concept, as it does not have specific metrics like latency or throughput and is dimensioned or evaluated many times with

terms like "Good", "Fair", "Poor". Nevertheless, this perception is composed by subjective factors and overall experience. QoS differ from QoE, because the first one does not really capture the actual experience of the user and that can dictate the use of an entire system or service [50].

Some sources consider QoE as an extension of QoS, as it tries to capture a service performance with metrics that could be directly communicated by user, for example, the playback start time of a task, or in video streaming the number of interruptions while watching a video, the duration of those interruptions and how that affects the user or the user engagement with that specific video, can contribute to a good or not quality of experience.

One way to try evaluating QoE in general with those more subjective factors is the use of Mean Opinion Score (MOS) classification. Users use the system/service and then rate it on a five-point discrete scale: 1-bad, 2-poor, 3-fair, 4-good, and 5-excellent. It's not easy to automate this process since many times the parameters to have in consideration are too many from user's age, genre to device screen size, device capabilities or internet connection [51]. Also, there are recent studies [52] that indicate that user's psychological state, for example, being on pressured situations rather than relaxed situations, can affect the QoE perception of a service.

Create a correlation model in order to map QoE/QoS is a hard task to do in an efficient way. Some models collect subjective measures as qualitative data by surveying user opinion, on the other hand, other models measure both subjective and objective variables by combining MOS with objective metrics [64]. Taking into consideration the literatures, it is a good approach to collect subjective measures as quantitative data, enabling statistical descriptions and combining the analysis of objective with subjective variables [51, 64].

2.4.1 Audio Quality

Phone calls quality is an essential feature to satisfy service subscribers. Acording to [45], regardless of the more sophisticated and powerful networks and smartphones "none of the 100 plus cell phones in Consumer Reports got an excellent or even a very good rating, for voice quality".

This can be explained by many factors: the smartphone size, cameras, microprocessors and other hardware, which allow those tiny devices do all the great stuff they do. Perhaps, sometimes the "speaker is wedged between the bezel and the front-facing camera" [47] covered in plastic

and microphone is placed on the back of the phone which is not optimal. Additionally, the reasons why voice calls aren't perfect are not only due to devices design architecture, the closest base station can be loaded at the time that a call is being done and is dependent of the distance that the packets must travel till they reach its destination because signal has to jump from cell to cell and every building, mountain or even the weather it's an obstacle making possible to have more than one path to reach the destination [47].

Deciphering this signal's multipath that reach smartphones at different times is not easy, as the signal sometimes has to be retransmitted, which happens in fractions of a second. Moreover, although a call can be done almost everywhere if we have coverage, some places are often very noisy with background sounds like traffic or nature. On those situations, neither the noise-canceling technology that some phones have can do much [48].

Due to all those experiences had with voice calls, alternatives like texting, e-mail and instant messaging are constantly growing at the moment, but there are promises of improvements with HD voice transmitting at a range of frequencies from 50Hz to 7kHz that are more close to human voice frequencies (75Hz-14kHz) than the oldest telephone network that had this ranges between 300 and 3,400 Hz [65].

Most of the new phones already support HD Voice, and is noticed voice quality changes due to the extended frequency ranges of audio signals. VoLTE technology is the bet for boost even more the cellular quality voice as it already supports Adaptative Multi-Rate Wideband (AMR-WB), thus it is not very difficult for carriers to enable HD Voice once calls have moved over to its LTE network. Backbone networks, local broadband links, IMS infrastructure and other issues like ensure priority for different traffic over the LTE network are being studied and developed and it will not take much time to "let VoLTE packets flow seamlessly between mobile handsets and other IP phones" [46, 48].

2.4.2 Video Quality

Comparing video quality is not so simply, because we cannot only compare video resolution, in fact, that could be really misleading because a 1080p video at 700MB size might look worse than the same video at 720p and 700MB, this happens because, for the former the bitrate is too high and it affects the compression. The same principle can be applied for comparing bitrates at similar frame sizes, because the encoders can deliver better quality at less or higher bitrate.

To evaluate video quality, there are a variety of quality metrics, but they can be divided into two big groups, no-reference metrics and full-reference metrics. The no-referenced ones give an output quality score based on a video and they can be used without access to the original video. When there is the original video to comparison the full-reference metrics can be used to estimate the quality loss.

Trying to define a MOS composed of individual quality factors is a hard challenge, where there are several parameters to be considered, at least video length, the viewing audience, original frame size, "quality" before the encoding and the type of video (cartoons, movies, news, etc.), not to mention that some people watch a movie sometimes for its content and they probably don't care about the quality so much, as long as the movie is funny or entertaining.

In the case of Android, there are video encoding recommendations that the Android media framework supports for playback in the H.264 Baseline Profile and VP8 media codec [69], shown in the tables 2.3 and 2.4, respectively.

	SD (Low quality)	SD (High quality)	HD 720p (N/A on all devices)
Video resolution	176 x 144 px	480 x 360 px	1280 x 720 px
Video frame rate	12 fps	30 fps	30 fps
Video bitrate	56 Kbps	500 Kbps	2 Mbps
Audio codec	AAC-LC	AAC-LC	AAC-LC
Audio channels	1 (mono)	2 (stereo)	2 (stereo)
Audio bitrate	24 Kbps	128 Kbps	192 Kbps

Table 2.3 - Examples of supported video encoding parameters for the H.264 codec *[69]*

	SD (Low quality)	SD (High quality)	HD 720p (N/A on all devices)	HD 1080p (N/A on all devices)
Video resolution	320 x 180 px	640 x 360 px	1280 x 720 px	1920 x 1080 px
Video frame rate	30 fps	30 fps	30 fps	30 fps
Video bitrate	800 Kbps	2 Mbps	4 Mbps	10 Mbps

Tabel 2.4 - Examples of supported video encoding parameters for the VP8 codec [69]

2.5 Mobile Operating Systems

A mobile operating system or mobile OS is the software that manages the resources and memory that other programs use. Its development allows developers to create and build specific applications with advanced functions and capabilities easily. Furthermore, these applications also have a great number of devices compatibility independent of its hardware. The most popular mobile OS, at the moment, are Google Android, Apple iOS and Windows Phone OS [70].

Different operating systems bring different navigation controls, user interfaces, features, security and privacy and could even have different support on peripheral support, for example. That said, when buying a new mobile device, it is obviously very important to look at these distinct aspects and see what matters most.

Nowadays, others OSs besides those already mentioned have less than 1% of the market share worldwide [27]. In terms of affordability it is well known that Apple doesn't make budget devices, and with hardware manufacturers like Samsung, ZTE, LG, Lenovo and Huawei abandoning the Windows phones, Android wins this battle, being the best choice for budget-conscious devices, which have similar specifications.

In terms of interface, every OS have its advantages and disadvantages. They are all unique with different animations and transitions. Therefore, having experience with one of them could lead to some learning time, when moving to another OS. The figure 2.11 shows an example of the Windows Mobile, Android and iOS interfaces, from top to bottom.

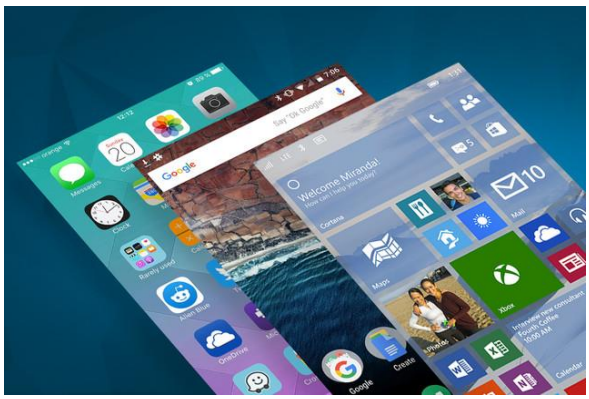


Figure 2.11 - User interface on different mobile OSs [28]

When talking about apps, Windows mobile is not even close of the iOS or Android market having less than one million apps on the Windows Store. Android's market share continues to

grow and has a higher percentage of free apps, however, iOS has been the most lucrative platform for developers, surpassing Android with more sales offer on apps and games [28].

Lastly and probably the main reasons this project focused on Android devices are the jailbreaking, bootloaders and the market share. This topic will be discussed in detail in the next subsection, but in comparison to Microsoft and mostly Apple, many Android OEMs offer a way to unlock their bootloaders. This type of control makes possible to do things that otherwise wouldn't be possible to do, upon which will be referred in the next chapters. Furthermore, Android dominates the market share, as shown in table 2.5, hitting a record of 88% in the fourth quarter of 2016 [54].

Period	Android	iOS	Windows Phone	Others
2015Q4	79,6%	18,7%	1,2%	0,5%
2016Q1	83,5%	15,4%	0,8%	0,4%
2016Q2	87,6%	11,7%	0,4%	0,3%
2016Q3	86,8%	12,5%	0,3%	0,4%

Table 2.5 - Worldwide smartphone OS market share [27]

2.5.1 Android

Android is a Linux-based Operating System for mobile devices, it was developed by Open Handset Alliance [80], but is owned by Google [81] nowadays [26]. This system is designed for touchscreen devices so the primarily market target are smartphones, tablets, Android TV and some gadgets that run Android, for example, watches (Android Wear).

Everything started with Android Inc. a company founded in October 2003 to develop "Smarter mobile devices that are more aware of its owner's locations and preferences" [26]. The first intention was to develop a system for digital cameras, but when they realized that the market was not big enough the attention and effort turned to produce a mobile operating system to rival Symbian and Windows Mobile at the time.

Google bought Android Inc. in 17th August 2005 developing a mobile operating system and making partnerships with mobile operators, software and hardware companies reaffirming that it would be open to mutual cooperation. Said that, the first commercially available smartphone running Android was the HTC Dream, released on October 22, 2008 [26].

At each major update, the Android system codename is changed, in alphabetical order, being always candy names. With Wi-Fi and Bluetooth [82] support Android usually comes with a bunch of Google apps like Maps, Calendar, Search, to name a few, a YouTube video player, a voice dialer, instant messaging and the option for user customize almost everything on its UI. Regular updates and upgrades over the years made this OS become more powerful, redesigned and with an increased overall performance.

The most recent version of Android at the time of this project, launched in 2016 during Google I/O, **Android 7.0 Nougat**, that now it's in version 7.1.2 (April 4, 2017) has the capacity to divide the screen in multiple sub screens, which improves multitasking user experience. It also brings native encryption, a new interface, virtual reality support, the ability of send messages through the notification bar and more [29]. The distribution of previous Android versions can be seen in the figure 2.12.

Version	Codename	API	Distribution
2.2	Froyo	8	0,1%
2.3.3 - 2.3.7	Gingerbread	10	1,7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1,6%
4.1.x	Jelly Bean	16	6,0%
4.2.x		17	8,3%
4.3		18	2,4%
4.4	KitKat	19	29,2%
5.0	Lollipop	21	14,1%
5.1		22	21,4%
6.0	Marshmallow	23	15,2%

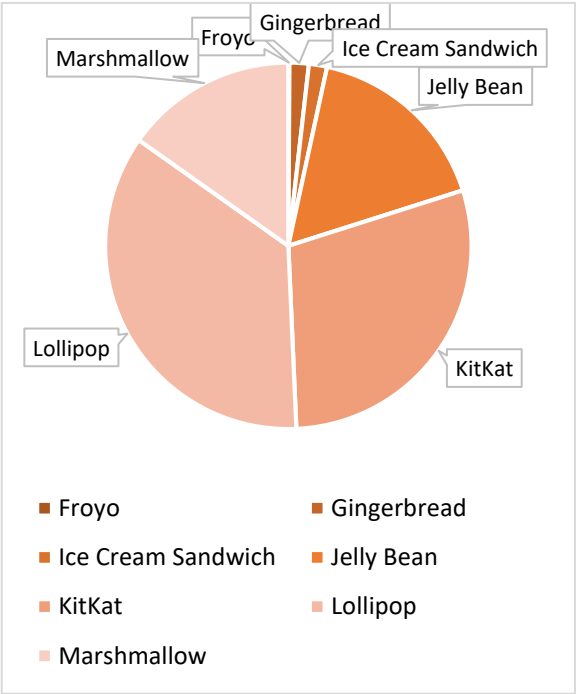


Figure 2.12 - Number of devices per Android version on 1 August 2016 [31]

2.5.2 Root/Jailbreak

Android phones is based on Linux permissions and file system, as was mentioned on the previously section. Therefore, different processes and users can have different privilege levels. These privileges determine what the process is allowed or prohibited from doing. All new apps that are installed have a type of user ID, and certain permissions to do certain things.

Root is nothing more than a user, but he has permissions “to do anything to any file in any place in the system” [32], consequently root brings an incredible power to user since he can overcome limitations that carriers and hardware manufacturers put on the device like uninstall pre-installed applications, speed-up the phone with *overclocking* or modify any file as he wants. Perhaps, "with great power comes great responsibility" because changes that he or apps that have root access do in the system file could lead to security risks if there is no certainty of what it’s being done [34].

The process of allowing Android devices to have privileged control (root access) is called rooting, and it gives similar access to administrative permissions to *superuser* on *Linux*. For ArQoS Pocket Solution, root may be needed if the communication with the management system requires an Access Point Name (APN) change, as it already happens with the other probes part of the ArQoS System Portfolio. For more information on the Management System and its relation with the probes, the architecture diagram of the solution can be encountered in chapter 3, ArQoS Pocket Platform. Furthermore, administrative permissions are required for investigation tasks like read the Android *logfile* or in development to run commands through the Android Debug Bridge (ADB).

This process is legal if you are the owner of the device. However, if someone use it to install spyware on other people's phones, that’s a crime in certain countries around the world [66]. Other than this, many vendors such as HTC, Sony, Asus or Google provide the ability to unlock their devices or even replace whole operating system, but by doing this, the user is “changing everything about the inherent security from Google and the people who built it” [32] and the warranty of the phone will almost certainly be voided. If the warranty is needed for repair some hardware issue, for example, it is possible to unroot a phone. There can be often find instructions for unroot a phone, usually it involves flash a *system binary file* (SBF) or an ROM Upgrade Utilities (RUU). However, some phones have a digital "switch" that flips when a device is unlocked and that is more difficult to revert, so it is advisable to do some research before unlocking a phone if it warranty is needed [33].

The steps for rooting an Android device are in general, unlocking or bypassing any bootloader protection, to allow the system partition to be written, and installing the relevant binaries to acquire root. Both steps vary from phone to phone, install a custom recovery, temporarily boot from an image over Universal Serial Bus (USB), or flash a new bootloader, are common procedures that can help advance the process. Other devices can be unlocked by accessing *fastboot* with a push-button combination. Many manufacturers now offer an official route to unlocking the bootloader [32, 33].

A rooted phone without a custom (Read-Only Memory) ROM flashed will likely still get (Over-The-Air) OTA updates from its carrier and perhaps, it will likely break the root. Fortunately, there are some free apps that can help keep the root access. Otherwise, flashing a custom ROM will make the phone not get OTA updates from its carrier anymore [33].

2.5.3 System application

Searching for an application on Google Play Store and clicking “install”, or loading an APK to the device, always installs an app as a data or user application. However, there are some scenarios where there is a need to install an application as a system app. This makes the app to be able to do certain actions such as programmatically change or add an APN to the device, among other features that Android only permits to system apps.

Certain permissions are protected under the *"signatureOrSystem”* protection level. Such permissions are not available to every application because they will grant risky privileges such as control over other applications, background installation and un-installation, among others that can be utilized for malicious purposes. Therefore, Android will only grant them to system applications or ordinary applications signed by the platform’s signature [36].

A System application is usually placed under */system/app* folder or */system/priv-app* on some Android devices. An application can only be installed in that folder with access to the OSs ROM *(system.img).* After that, a device which loads the custom ROM will have the new system application added. Another particularity of a system application is that, they cannot be removed from the device (cannot be uninstalled by a non-root user), because */system/app* is a read-only folder [35].

On the other hand, a non-system application is an ordinary application which will be installed under */data/app* folder, and which is read-write. It is possible to check if an application is a system

application or not, using *"ApplicationInfo.FLAG_SYSTEM".* If the constant returns true, then the application is a system application.

Programmatically change an APN requires a permission called *"write_apn_settings"* only given to system apps. Therefore, to do this change it is needed, not only root in the device, but also that the app runs under the *system/app* or *system/priv-ap,p* in some devices, with the referred permission in the Android *"Manifest.xml*".

2.6 Other Solutions

There are hundreds of applications in the market to test the mobile networks, Wi-Fi networks or retrieve multiple information about the device, but only a couple of solutions do all that in the same application and still, have a strong test components for Voice or Over the Top (OTT) apps, for example. This project addressed those solutions that have this network information component, but also, a component to test the network as well, making the solution more complete from the QoS assessment point of view.

Many apps that do some of these tasks have a free lite version available in the market for users to test, retrieving parameters from both Wi-Fi and mobile networks, but usually they have a more detailed and paid version as well, exploring other features and with a more complex set of network tests, as is the case, of the PRTG Network Monitor app [71], for example.

From the QoE standpoint, *Xperience* app from Streambow [72] is highlighted, as it provides functionalities for the user to submit bad coverage zones, drop calls, bad quality audio, slow data, among others in order to give feedback to help the operator understand faster where the points that need some improvement are. There are also solutions focused at driving tests and vehicle installation, passively testing and taking snapshots of the network. Furthermore, the schedule of tests to a specific date and hour were also possible in some concurrent solutions.

Following a similar approach, were found solutions like Xcal-Ranger from Accuver [73] with a setup where the devices were in a suitcase, ready to be placed at critical points on the network or to be taken through the streets analyzing the network in drive tests as was mentioned before. These types of solutions communicated with a backed sending all the information and results to the last and could be controlled remotely in a backend interface solution, for example, executing scheduled tests at a certain time.

The "*TEMS"* solution from InfoVista [74], the "*TrueSite Solution*" from VIAVI [75] and the "*XGMA SP"* solution from Focus infocom [76] were the most complete in providing information about the mobile network, retrieving parameters like the cell signal strength, cell id, operator's name, technology, if the roaming is active or not and many others parameters that can vary with the mobile network technology that is being used at that moment. Was also found a solution focused only on the Wi-Fi called Sigma-ML from the Meritech company [77], which provided lots of parameters like the Basic Service Set Identifier (BSSID), Service Set Identifier (SSID), IP, Media Access Control (MAC) address, primary and secondary DNSs addresses, among many others. Information about the device and its specifications such as location, IMEI, battery level or free memory available in the device will be relevant for some task that will be referred in chapter 3 and that's possible to retrieve, as well from Android internal classes.

Commercially, the available network testing tools vary from simple PINGs, HTTP Download/Upload, send an SMS/Multimedia Messaging Service (MMS) or access an e-mail server to, in some more expensive solutions encountered, voice tests using Perceptual Evaluation of Speech Quality (PESQ) and Perceptual Objective Listening Quality Assessment (POLQA), OTT applications like Skype or YouTube, video streaming quality and tests to other applications in vogue like Facebook, YouTube and Dropbox in the various technologies (EDGE, GSM, HSPA+, LTE, …). Still on these more complex solutions, the support for the VoLTE technology is available and for interested clients, the deployment, training and support is also offered by a couple of years.

Figure 2.13 - Existing concurrent solutions

The table 2.6 shows a comparison between the different encountered solutions. Reinforcing the idea and the mobile operating system's market share, presented in subsection 2.5, only four of the sixteen solutions analyzed in this work had an iOS version and the ones who had, also had an

Android version as well. The remaining solutions were developed exclusively for the Android OS. The POLQA column its related with the voice tests. Its goal, as already referred, is to predict speech quality by performing a digital speech signals analysis.

Not all the solutions were tested because most of them are paid. A special highlight given to the QualityProc android Swissqual [78] from Rohde-Schwarz, which seemed to be one of the most complete solutions, with a case for multiple devices that could be remotely controlled and supporting multiple tests for voice, video, data and messaging, beside the many also performed by other solutions. Additionally, it supports video streaming quality analysis and data information on OTT services like Dropbox or Facebook. Lastly, it generates more than 200 KPIs, sent continuously to a backend system and decodes protocol messages of various layers on the supported technologies: 3GPP, L2, L3, TCP/IP, IMS, and SIP.

Product's name	Android	IoS	POLQA	Device Information	VoLTE	Wifi Metrics	Video/ Apps	Test's Scheduler	Error Recuperation	Base Tests
PRTG	x	x		x		x			x	x
Xperience (Streambow)	x			x		x		x	x	x
Xcal-Ranger (accuver)	x		x		x		x	x	x	x
Nemo Handy (annite)			x							x
TEMS (Ascom)	x		x	x				x		x
Amanzitel – Customer IQ	x	x	x	x		x				
XGMA SP (FocusInfocom)	x		x	x	x		x			x
Keynote (SIGOS)	x	x	x		x		x			x
The TrueSite Solution (VIAVI)	x				x	x		x	x	x
Sigma -ML (Meritech)	x		x	x				x	x	x
FalconProbe	x			x		x	x		x	x
QualityProc android Swissqual	x		x	x	x	x	x	x	x	x
SQlive (Telchemy)	x					x	x			x
Gemalto	x	x				x	x	x	x	x
GL communications	x		x	x				x	x	x
Spirent Epitiro	x			x		x	x		x	x

* Base Tests = HTTP, FTP, SMS, PING.

Table 2.6 - Concurrent solutions: comparative table

Chapter 3

ArQoS Pocket platform

This chapter presents a solution to complement the ArQoS system referred in section 1.2. It is an Android app solution that works as a mobile probe collecting multiple data and indicators of the network.

Supporting multiple technologies on mobile networks (GSM, GPRS, UMTS, HSDPA, HSUPA, HSPA+, etc) and Wi-Fi this solution allows continuous tests to check the connectivity and availability of the network, as well as help on troubleshooting and monitor the quality of service with intrusive tests.

3.1 ArQoS

ArQoS is a performance monitoring system that evaluates QoS and QoE on multi-technology fixed and on mobile networks. The complete ArQoS portfolio has fixed and mobile probes in the network testing Voice, Wi-Fi, Interactive Voice Response (IVR), messaging, video-streaming and producing radiologs with snapshots of network's state. Furthermore, there are the probes called NI probes passively testing VoIP services and PON probes testing the optical fiber [20].

The ArQoS NG probes are the most similar probes to pocket since they are small, have a low power consumption built in battery for powerless scenarios, have an integrated GPS (Global Positioning System) and support multi-technology (wireless and mobile), four slots for SIM cards, up to nine antennas, an UBS console port and a 12V plug for external power source connection.

On the other hand, the fixed probes have support for Public Switched Telephone Network (PSTN), Integrated Services Digital Network (ISDN), Ethernet and wireless networks and test video, data, IP, messaging and voice tests (including VoIP).

ArQoS Pocket is a solution that complements this portfolio running over Android devices which act as probes. The end-user as, in this solution, an UI to see the obtained results, which is not possible with the other ArQoS probes.

All these probes are controlled by ArQoS management system, a multi-user system that configures and manages the probes remotely. With flexible test configuration and using an intuitive drag and drop interface it allows to powerful test schedules and troubleshootings operations. This system receives information from all the probes described earlier and provides detailed results and reports based on that information analysis.

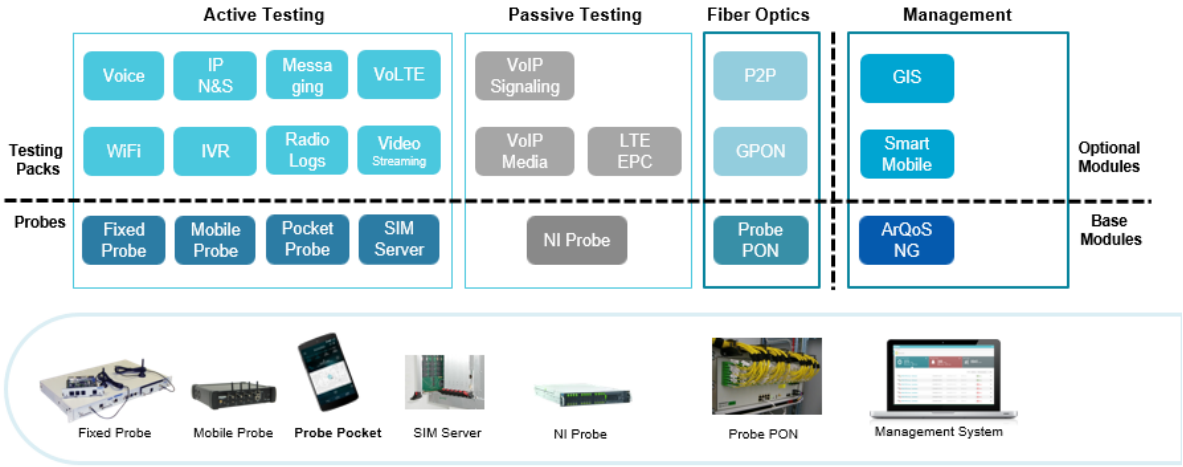


Figure 3.1 - ArQoS Portfolio

3.2 Objectives and Requirements

The ArQoS Pocket is an ArQoS integrated solution that runs on Android terminals, monitoring actively and passively Wi-Fi and mobile networks. The objectives of this solution are testing the connectivity and the availability of the operator's networks by performing tests in order to troubleshoot possible problems and monitor the network. Another main objectives of this software are: collect KPIs and KQIs on a large scale to grant network and service performance and reliability. Those indicators are then sent to a backend where statistical graphics, colored and filtered maps can be generated in order to give an overall state of the operator's network.

Simulating the user experience with scheduled tests plans, it is possible to provide continuous service performance monitoring, and guarantee a healthy network by detecting behaviors that can jeopardize the performance and violate customer SLAs, maintaining customer expectation and loyalty.

This is important not only to help operators on knowing where the failure points are, but also to help on planning the network to cope with new business needs. Furthermore, this app will also help to improve operator's services that already exist and consequently raise the quality of experience percept by user.

Figure 3.2 shows the requirements and below, the main objectives for the development of this solution. Complementing other probe's data with QoS and QoE indicators from Android devices and simulating the user-experience with scheduled tests, this solution allows the production of indicators, which will be detailed in the next chapter, for service performance, reliability and availability. These indicators contribute either for QoS improvement activities or for a better network planning in a future.

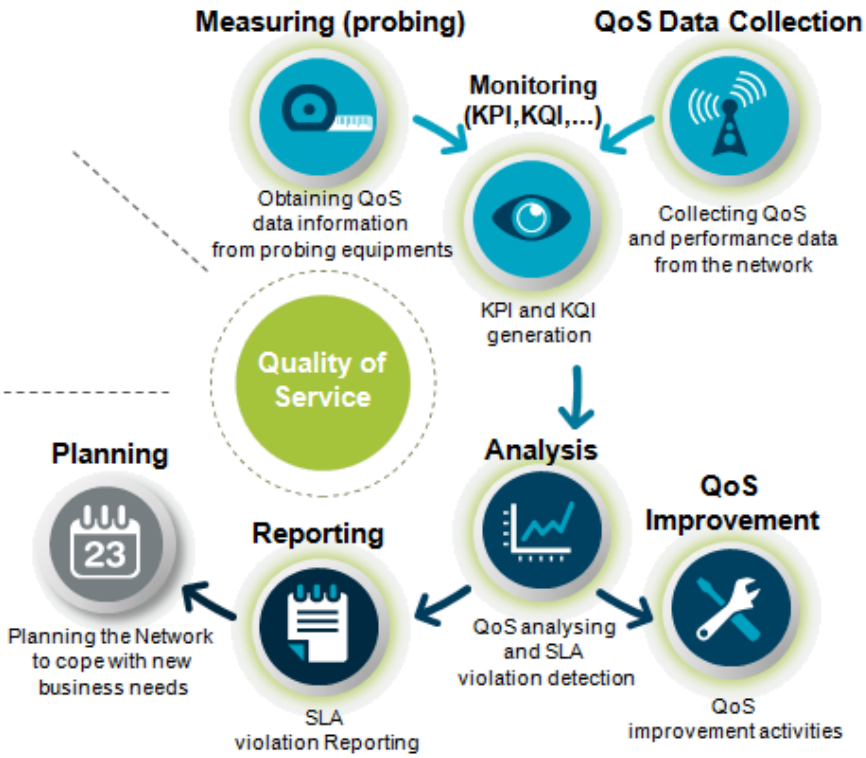


Figure 3.2 - ArQoS Pocket objectives

ArQoS Pocket is a solution that is being developed since 2011. As such, the current version already has some features implemented. Those features are: a dashboard with a summary for the mobile network and Wi-Fi reception in real-time, an anomaly page that allows users to submit failures detected on the network, associating it with geographic information and lastly, a tests page where users can test the network. Tests already implemented comprise PINGs to a specific

website and HTTP tests (download/upload) that allow the app to do, for example, browsing or network speed tests.

With a deeper analysis of the code, it was considered that the project structure should be refactored, not only because the code has been developed for several years by different persons, but also to have a structured Android project architecture, which will allow to save time in the long-term for future developers. This is, therefore, an improvement of some non-functional requirements since this refactor makes the application more scalable with the upgrade from local file storage to a database storage, more flexible and capable of failure handling, with the use of new frameworks that handle asynchronous tasks with separated threads running in parallel.

This project intends to divide the existing extensive classes in multiple shorter ones, that only have/run code of what they are meant to run and let other tasks to the respective entity, as well. Beside this, user-friendly interfaces and fast responsiveness of the app, were also focus points taken into account on the development, trying to give always feedback to user on every action performed, as well as parallelize or execute in background what could be executed with transparency for the user, not leaving him stuck waiting for the end of certain task.

Despite the passing of the years, this project has been stopped for a long time and there are still many functional requirements to be developed. The solutions still need to communicate with a backend and send all the anomalies, tests results and radiologs to the management system, receive configurations and schedule tests, which should run at a specific date, save all the results locally and implement features like radiolog and snapshots and support Short Message Service (SMS) and voice tests.

The identified requirements for this project are: the support for the tasks to send and receive an SMS, support for voice tests (record, reproduce, make, answer and hang up a call), the storage of all the results in a database, the connection to the management system and delivery of the results, the possibility of test scheduling, the probe's discovery and lastly, the production of radiologs, snapshots and scanlogs, which will be detailed in the subsection 4.2.5. These requirements can be seen as well, in the figure 3.3, and will be explained further with detail.

Figure 3.3 - ArQoS Pocket Requirements

One of the main functional requirement is to improve and implement more tests, such as SMS tests, voice tests (establish a call, answer a call, hang up, reproduce and record audio). Those tests must have a snapshot of the mobile and, if possible, Wi-Fi information attached, as well as the GPS location.

Other functional requirement is the production of radiologs in JSON format with device and mobile network information or Wi-Fi information, in the case of scanlogs. A variation of this, is a snapshot, which consists of a radiolog with a user commentary at the end, describing why that snapshot was taken. In addition, there are special radiologs taken when certain events happen. These events can be a call establishment, a call drop, a handover, a roaming state change, a cell reselection, a call setup or a call release. Those radiologs must be activated by the user, or remotely by the management system with a configurable interval, and stored locally according to a configurable space limit.

The mandatory parameters of a radiolog are the probe Integrated Circuit Card Identifier (ICCID), the timestamp in epoch time (time in milliseconds since 1/1/1970), probe's International Mobile Equipment Identity (IMEI), the GPS coordinates, mobile network mode, id of the cell, operator's name, if device's roaming is active, the rxlevel, area code and some specific parameters that vary with the technology, for example, the Received Signal Code Power (RSRP) on 3G and the Reference Signal Received Power (RSRP) or Reference Signal Received Quality (RSRQ) on 4G. Additionally, it can contain the neighbor's cells information, an event or a user feedback comment in the case of a snapshot.

Other requirement, considering the amount of information that now needs to be stored is to pass from the local storage of those data files to a local database. In addition, Wi-Fi information is also stored in a similar fashion to the radiologs in a table called scanlogs. In the next chapter, it will be explained how these requirements were implemented, how the database is structured, what parameters are being saved and what they mean in more detail. It should be possible for the user to choose if it wants to store the results in the phone's internal memory or in the external memory card. The user should also be able to consult those results in a map or a list.

A key feature in the ArQoS Pocket solution is the scheduling of personalized tests. The device can have pre-configured tests available, but is mandatory for the app to receive configurations and scheduled tests from the management system. In addition, anomalies and networks logs must be saved locally and may be seen in an history page with all the information associated.

Other key feature for the ArQoS Pocket solution is naturally the integration with the ArQoS management system. This communication should be asynchronous and bi-directional in order to reduce and distribute the processing load by the probes and by the management system. It should, as well, be based on Representational State Transfer (REST), HTTP with Java Script Object Notation (JSON) and Secure Copy Protocol (SCP)/Secure File Transfer Protocol (SFTP) and follow the same protocol definition as the NG probes, ensuring compatibility and optimized integration effort. Furthermore, the use of web protocols allows a more direct and efficient processing on the management system side, without a significant overhead on the probes side. The connection also handle several issues associated with the probe discovery, the results delivery, the test's programming and schedule through the management system and the support for dynamic configurations.

The connection with the management network depends on the availability of the latter. Usually the management network is an internal operator's network accessed frequently via a specific APN. In this case, there are two ways to initiate this connection: by user request or automatically. In the first case, it is the user's responsibility to turn on Wi-Fi and change the default APN on the Android configurations, enabling communication with the management system. The "automatic" mode has an APN address configurable by the user, and all the connection establishment happens automatically, with transparency for the user. In this last case, the app cleverly connects to the management system, delivering the results when there are no tests to execute (at free times). In case of failure, the application must wait a "*backoff*" time and try the delivery again later.

The last requirement is to have an option in order to automatically start the application when the device boots. If possible boot the device automatically, if it is charging, in ordert to support scenarios where the device is analyzing the network without any user interaction.

3.3 Stakeholders

A project success depends a lot of the agents involved in it, and it's crucial to identify the stakeholder's expectations to satisfy them. ArQoS solution has essentially 3 parties involved, namely: project team members, project managers and general users. The project team members are composed by: the ArQoS pocket developers, the ArQoS NG team, management system team and the usability team. To help on understanding what these teams do, a complete diagram, involving all these stakeholders, its functions and interactions, is shown in the figure 3.4.

Project team members are essentially, the ones that have a direct influence on the project development and execution, consequentially, they have a big responsibility since they are the ones who implement all the features and requirements of the project. In this solution, project team members can be divided into ArQoS Pocket developers, ArQoS NG team, the management system team and the usability team.

As mentioned before ArQoS solutions already exists in production and use fixed and mobile probes, testing and retrieving QoS and QoE metrics from the network. These probes and the tasks they perform are improved all the time. Therefore, there are many features that the ArQoS Pocket solution must support that are already implemented and executed on the fixed/mobile probes spread across the world. The experience and the knowledge of what is expected to happen provided by ArQoS NG team, are a great help for the ArQoS Pocket developers making possible this last team to progress more quickly and optimally. The management system team is composed by developers and system administrators that work on the management and maintenance of the product backend.

The usability team contributes for the product with usability tests, studying the user's interaction with the application User Interface (UI) and detecting possible problems that they may

encounter. Focusing on behaviors and not in opinions, this team main goal is to understand how the people use the product and after an analysis and propose a solution to improve the product design and user interaction [53]. This team has contact with many frameworks and has many years of experience with users interface issues. Additionally, this team has great design skills too, producing all the graphical assets needed for the product, therefore also contributing to a quicker development since programmers can be focused on the code only, instead of implementing both functional features and the design at the same time.

ArQoS pocket developers are the ones that have a direct contact with the code, developing and implementing all the requirements defined by the project. Sometimes the product installation, training and acceptation are made by them, as well. They are led by the project managers, whose function is to guide them, validate all the implemented features, define, order the product roadmap and finally, handle the sales with the final customer.

These teams have one or more project managers, need to communicate between them and are completely focused on the project success, aiming at satisfying the customer's needs and expectations. A project manager leads the project, the project team, and is the one that makes the decisions when needed. He can also deal with the sale of the product, having the first contact and handling the communication with the final customers.

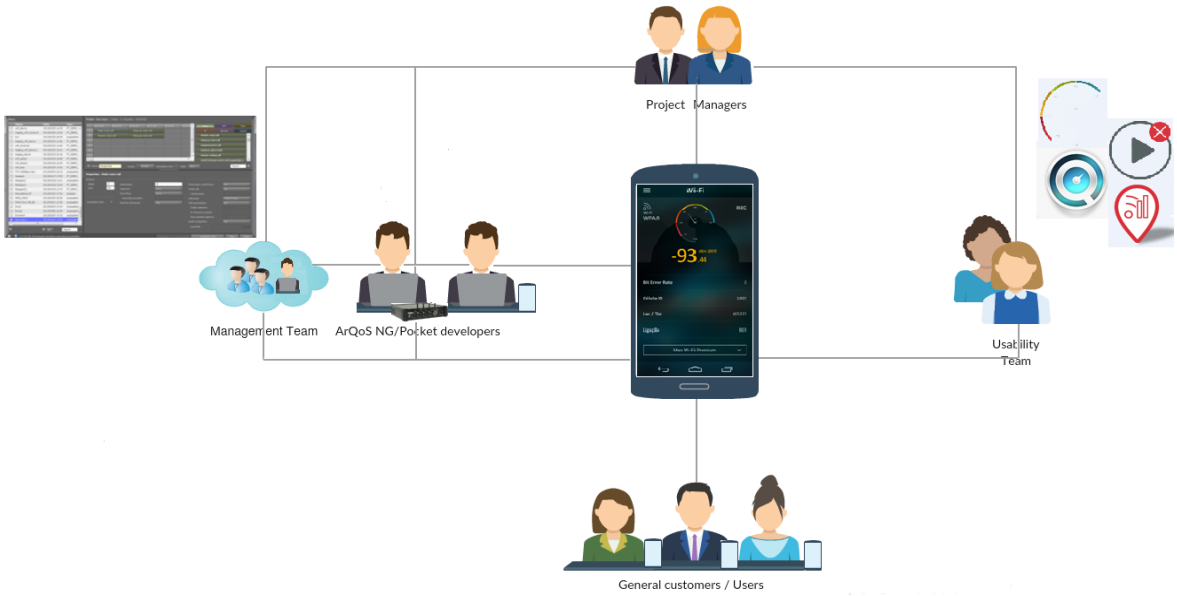


Figure 3.4 - ArQoS Pocket: Stakeholder's interaction

3.4 Use cases

To better understand the features offered by the ArQoS Pocket solution and how it interacts with the users, in this section will be presented the main use-cases examples on using the ArQoS platform.

The presented solution does some tasks in the background such as receive programmed tests from the management system, register network events like call drops or cell reselections and send tests results and radiologs (network information at a specific time and location) to the management system.

Besides tasks that run in the background, without the user perception, there are also on demand tests that user can execute. This can be done by accessing the “Tests” page and by pressing the “play” button on a loaded test. All the results for a specific test can be seen at the test’s history page.

The network state can be seen as well, by taking on demand snapshots. These snapshots are instantaneous radiologs taken by user at “Radiologs/Snapshots” page where the user can attach a feedback message, describing why that snapshot is being taken, for example. The parameters contained in this structure can be observed in the Appendix C.

On the dashboard it is possible to see Wi-Fi and mobile networks details. These details contain the mobile network technology, the signal level, the device identifier(ID)/IMEI, the cell ID, operator code, the operator name, the Subscriber Identity Module (SIM) Card IMSI and if the roaming is active or not at the time, for the mobile network. On the other hand, for the Wi-Fi network it is possible to retrieve the link speed connection, the SSID, channel number, BSSID, if the SSID is hidden, the device IP address, the primary and secondary DNS addresses, the Dynamic Host Configuration Protocol (DHCP) lease duration time and the gateway IP address.

Report an anomaly on the network is another use-case, which can be done by accessing the anomalies page. The first step is to choose between 5 different categories: Voice, Internet, Messaging, Coverage or Other. After that the user must choose the proper anomaly detected for that category. For Voice is possible to choose between a dropped call, an unanswered call, a missed call, poor audio quality or other. On the Internet category, the options are: no data access, intermittent access, slow access, message not sent or other type of anomaly. On messaging: message not received, slow delivery. Finally, on Coverage the options are: no indoor coverage or

no outdoor coverage. After that the user can slide the map to the position, where the anomaly was detected and optionally, attach a feedback message. The described views can be observed in the figure 4.5.

Lastly, the user can change the app settings at the “Setting page”. This page makes it possible for users to choose the time interval for radiologs registers, the app language (English, Portuguese, French) and the application homepage. All these features, previously described, are summarized in the figure 3.5.

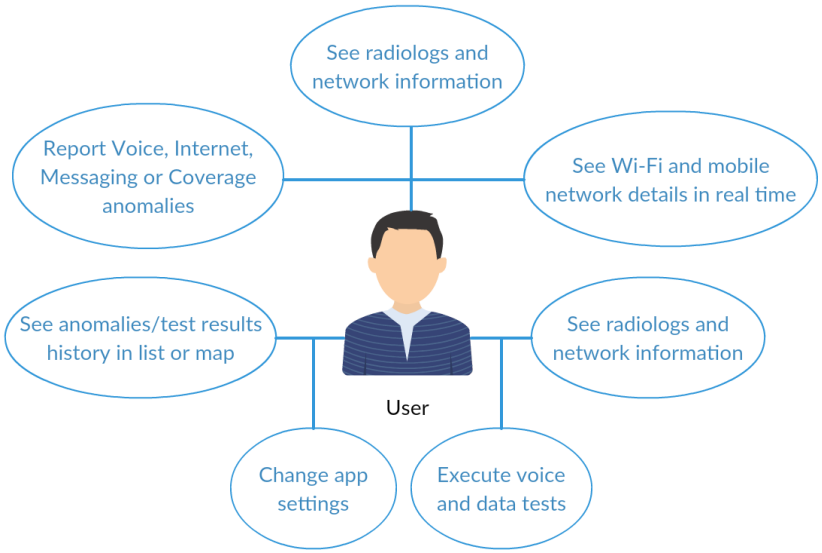


Figure 3.5 - Use cases diagram: User - App interaction

All the tests results, anomalies and radiologs information taken by the application, is gathered in a backend, making possible to the system administrators to have an overall overview of what is happening with its own network. This gives more knowledge to the operator and facilitates important decisions, such as prioritize the action points. Furthermore, with all these data, the operator can now filter all information by technology (2G, 3G, 4G), date or by events to see geographically where special events such as bad coverage zones or more dropped calls happen.

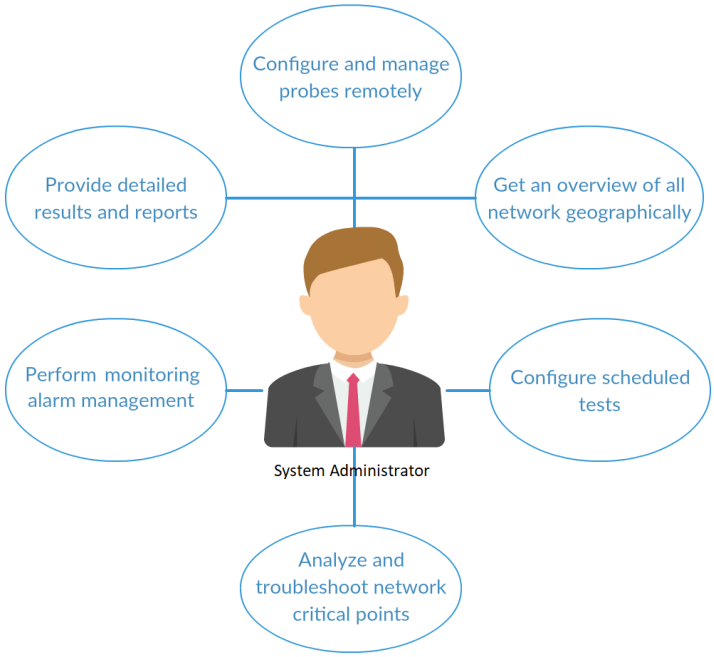


Figure 3.6 - Use cases diagram: System administrator - Management system: interaction

At the management system it is also possible to manage and change the configuration of all ArQoS probes (fixed, mobile and pocket) and program scheduled tests for a certain date to troubleshoot network operations, for example. How the pocket probes are configured, how is a test scheduled and all the protocols involved in this communication will be detailed further in the section 4.3, Management System Connection. Figure 3.6 shows the use-cases referent to the backend/management.

3.5 ArQoS Pocket Solution Architecture

ArQoS global product, whose architecture is depicted in the figure 3.7, is used all over the world measuring QoS data information from probing equipment's (mobile and fixed probes). All data is gathered, parsed and analyzed in the management server, to improve user quality of service and experience with operator's services.

The ArQoS pocket probes support various mobile networks technologies such as, GSM, GPRS, UMTS, HSDPA, HSUPA, HSPA+, LTE and Wi-Fi complementing the ArQoS NG probes with continuous tests to infer the network performance and availability. Contrary to the automated and centrally-managed NG probes, pocket probes have a user-friendly UI that allows the users to

see the results of tests done in order to troubleshoot possible problems or to monitor the services QoS.

The communication with the management system will allow the probes discovery, the delivery of all the probes results, the scheduling of tests and to configure all probes remotely. Furthermore, it should be possible to request the probe's state, hardware information, the probes occupation between two dates, among other requests.

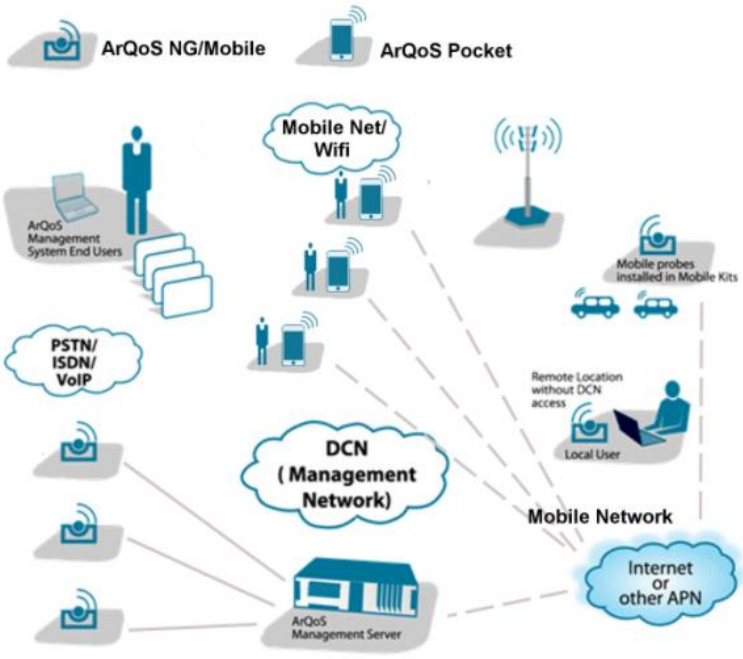


Figure 3.7 - ArQoS architecture diagram

Figure 3.8, shows the architecture particularly from the ArQoS Pocket solution. The Android application is divided into 3 main packages: app, core and persistence. The first one contains all the code referent to Android activities, fragment's logic and manages all the animations and transitions in the user interface.

The core has all the code logic that is needed without affecting the UI directly, for example, retrieving mobile or Wi-Fi network parameters from Android internal classes on the various technologies and finally, the "persistence" package is related to the SQLite database [83] that we can see in figure 4.1. This database contains all the anomalies detected on the network, radiologs (mobile network information), scanlogs (Wi-Fi network information) and test/tasks results that will be after sent to the management system via REST.

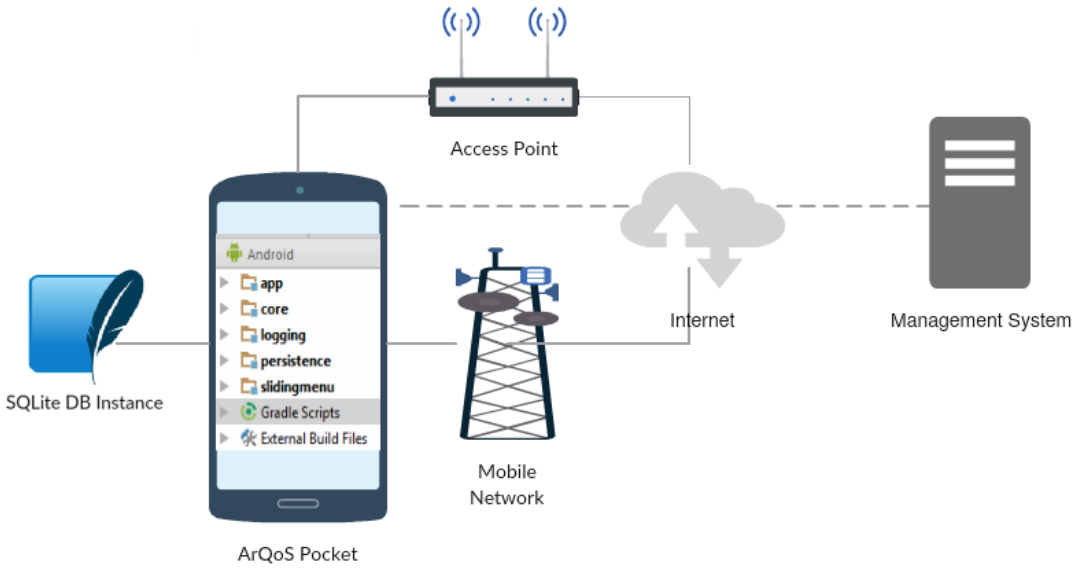


Figure 3.8 - ArQoS Pocket architecture diagram

The database has support for tests, anomalies, radiologs and test results. Tests are distinguished in scheduled tests and on-demand tests. Also, it is possible to cancel or update a test.

Scheduled tests are assigned to a certain date that must be in the future to be valid. Otherwise, it will be ignored by the database and not executed. An example of these tests can be seen in snippet 4.2, along with the many parameters that can be configured and which will be explained in the next chapter. On-demand are a different type of test where the initial and final dates of the test are only used to validate the test, because it is a test to run on user demand, by clicking the "play" button on the test details view.

One test may have multiple tasks associated to it. Tests can have more complexes parameters such as "recursion", which makes it execute periodically. There are common parameters to all tasks and then specific parameters depending on the task being executed. After a task has executed, a result should be obtained and saved into the database. Similarly to tasks, a test has a result too which may or may not be successful, but in the test case, if just one task fails for the test, that test will be considered a failed test.

Independently of the scenario where the application is used the objectives remain the same: gather QoS metrics, performance data, network information and deliver that information to the management system, making possible to improve QoS and QoE to the final user. A prototype of the scenario described can be seen in chapter 5, Evaluation and Results, in the figure 5.3.

Chapter 4

ArQoS Pocket Implementation

This chapter's objective is to describe all the implementation, explaining the steps taken and the reasoning behind each decision, giving a more powerful insight of the work done.

Section 4.1 describes the technologies and frameworks chosen to support whole the development, Section 4.2, shows the current app UI and explains the progress done on it. Section 4.3 details the communication with the management system. Finally, Section 4.4 describes the new implemented tests and features developed on the solution.

4.1 Technical implementation

The used programming languages were Java and C++. Java because is an Android application and C++ to enable the integration of required JNI libraries. From the moment that the app starts opening, several background services are called in order to support the UI information that is seen on the application. The app homepage is usually the "Dashboard", but may be other page/activity. Nevertheless, the splash screen that is seen when the app initiates, for the first time, allows the app to buy time and fill the structures need for update the information needed.

Wi-Fi and mobile networks information are retrieved from numerous Android internal classes by the *WifiManager* and *MobileNetworkManager* services, respectively. Scheduled tests callbacks are done in other service. In other words, every time that it is required to update a scheduled test date, update the Wi-Fi or the mobile network information, these services are the ones that handle the updates and then forward the information again to update the respective UIs pages.

The radiologs information is filled in if on the Settings page the option to register radiologs is enabled. If so, the *RadiologsManager* service is called automatically, filling in the radiologs fields with the already existing Wi-Fi and mobile networks information plus some information about the neighbor cells and, if available network event's information. Subsection 4.2.5 details the possible events automatically acquired by this service.

Similar to radiologs, anomalies are also stored and managed by the respective entities/classes and procedures by the loading procedure of its history pages Then existing adapters transform the raw data in structured data, which can be used in the UI or be delivered to the management system.

There is another type of adapters that handle the multiple fragments from an activity. These fragments represent a behavior or part of the UI and they can be reutilized in multiple activities. In figure 4.5, it is presented a great example of this with multiple fragments all in the same activity and with the same purpose, report an anomaly.

All the tests executed by the app are saved in the database. It stores anomalies, radiologs and test results. The tests can be on-demand or scheduled. There were also developed features to update or cancel a scheduled test, if and only if, that test is not already running.

There are six test types: “Date”, “Boot”, “Iterations”, “Time interval”, “Week” and “User request” defined for ArQoS NG probes. The application supports both “Date” and “User request” tests, at the moment. All the tables containing the tests and its associated tasks, how they interact with each other and some examples of the fields on it are shown in the figure 4.1, below.

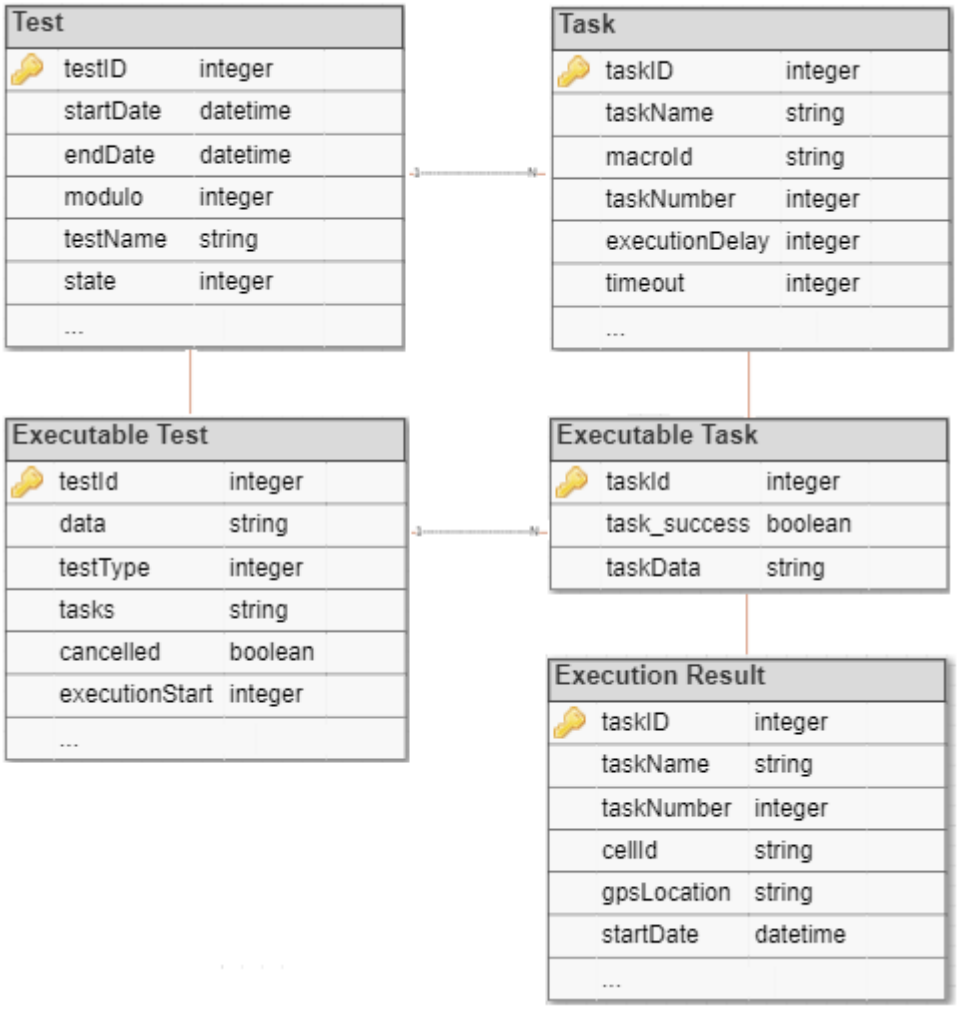


Figure 4.1 - ArQoS Pocket database schema: Tests and tasks results

There are common parameters to all tasks and then specific parameters depending on the task being executed. After a task has executed, a result is obtained and saved into the database. Similarly, to tasks, a test has a result too which may or may not be successful. In the test case, if just one task fails, that test will be considered a failed test.

The application can be adapted to different type of users and deployed in multiple scenarios, working as stand-alone or with interacting with of the already existing ArQoS NG probes on the network. Users have in the settings page a couple of options to explore, allowing the application to adapt itself according to the user's preferences. Regarding the settings page, a user can activate or not the radiologs registration and choose the time interval between each register. It is also possible to choose the application homepage and the app language. All of these parameters/choices are saved locally into the S*haredPreferences,* which is an internal Android

interface for accessing and modifying preference data. That way the same application installed in different devices can run with different configurations saved locally in each device.

A different service, running in the background as well, is a GPS based location service that retrieves the latitude and longitude coordinates, altitude, among other device parameters. This service is used primarily to show pins/markers of the different anomalies, tests or radiologs on history's map view, but also to attach to every task details the GPS information that will after be sent to the backend system.

Lastly and associated to the test scheduling, there is a background service handling the Alarms. An alarm is triggered every time when it's needed to insert, update or cancel a test. The *AlarmManager* Android class [87] is being used, since it already provides the tools to handle the alarm services and therefore, ensures that the application is capable of running a test at a certain time, in the future.

4.1.1 Frameworks

Many tools were used to build a consistent, integrated and operational solution. Those frameworks/technologies used are the Android SDK, JSON, SQLite database, Google Maps API, ReactiveX for Java [88], Requery [89] and image edition tools.

The Android SDK is the software that helps and enables the development or creation of applications for the Android platform. For this project was used the Android studio 2.3.1 and the application was tested with an emulator included in the SDK, but primarily, in real devices whose specifications can be encountered in Appendix D.

Test's data, parameters and configurations are received from the management system through HTTP POSTs with JSON content and handled by an internal database. Still about the database implementation a framework named requery, was used, providing classes for SQLite databases and useful UI adapters. Primarily to facilitate much of the database creation, it also brings other advantages like lifecycle callbacks, Java Persistence API (JPA) annotation support, caching and no dependencies are needed.

ReactiveX is a cross-platform API available in multiple programming languages that manipulates UI events and Application Programming Interface (API) responses. Furthermore, it handles concurrency and asynchronous processes that almost every Android application needs to

implement, because most of the http requests are not allowed to be done in the Android main thread. Reducing the amount of code needed in order to obtain synchronicity between the UI, the database and the backed server, this framework is equipped with proper mechanisms for handling errors and allowing the programmer to abstract low-level threading, synchronization, and concurrency issues. In this project, RxJava is used mainly to handle asynchronous events and database transactions.

The map that is seen in the distinct history pages, the personalized markers, the zoom and all the animations associated are done using the Google Maps API.

4.2 Application UI Overview

It is possible to divide the application into six different main parts, Dashboard, Anomalies, Tests, Radiologs/Snapshots, Settings and the About section, as can be seen in application main menu illustrated in figure 4.2.

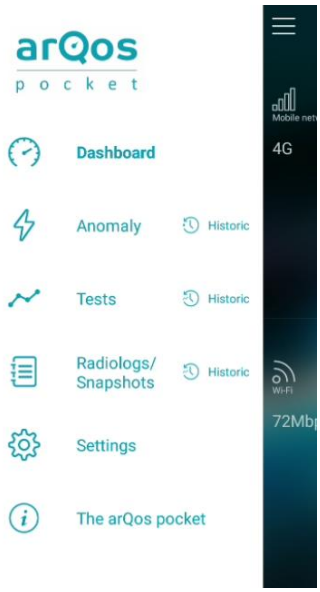


Figure 4.2 - ArQoS Pocket UI: Sliding Menu

4.2.1 Design options and techniques

This subsection, presents some design techniques used to improve the user experience on our solution, as well as patterns and templates used to provide a consistent user interaction.

The app opening starts with a splash screen that can be seen below in figure 4.3, this allows a background service that initiates at the app start to load the app content and thus, fill in all the information correctly before user reaches the application homepage. A more detailed and technical information about this and other services that are running on apps background will be explained in the next section, technical implementation.

Figure 4.3 - ArQoS Pocket UI: Splash screen

There is a lot of Android material design used, but there are a lot of customized buttons, *textviews*, *checkboxes* and *dropdowns*, as well. In the Settings page, which can be seen in figure 4.10 it is possible to observe customized *textviews* called "SuperTextView" in the project, round checkboxes and custom switches composed by an image indicating success/failure and the switch, followed by a message in order to give the user feedback of the operation state.

A framework for the slide animations called "androidslidingmenu" was used. Every time that the user slides something, for example, to see more details about it, it is indicated in the app with a "plus" symbol or similar symbols placed on the right or left borders of the application UI. Examples of this interaction can be seen in the figures 4.5, 4.7, 4.9 and 4.9.

In order to help the user on finding quickly what he intends, in the history pages it is always possible to filter the information by categories, as depicted in figure 4.4. This feature is implemented both in the history list view and map view, filtering or cleaning the list or the map pins, respectively, of others than the selected category.

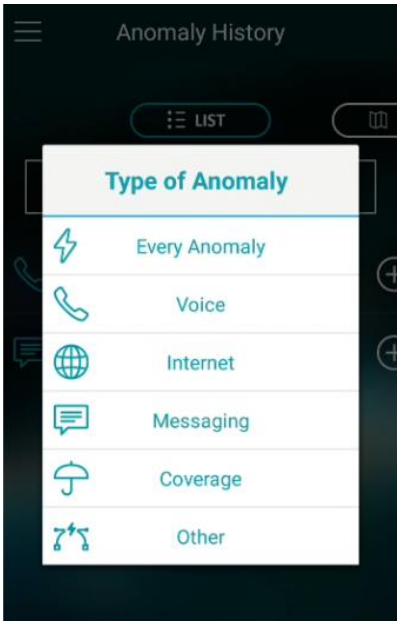


Figure 4.4 - ArQoS Pocket UI: Anomaly's filtering dropdown

4.2.2 Dashboard

The dashboard allows seeing mobile and Wi-Fi network information when available. In the main page, users also get an idea of the signal strength observing both gauges in the center. Below the mobile network icon, the technology of mobile network (2G, 3G, 4G) can be seen, and below the Wi-Fi icon it's the link speed connection in Mb/s can be found.

Sliding in each half of the screen will lead us to a more detailed page where can be observed the Cell ID, Device ID (IMEI), IMSI, Operator name, Mobile Country Code (MCC)/Mobile Network Code (MNC) and if the roaming is active or not. On Wi-Fi side, there is information about the APs BSSID, MAC address, channel number, primary and secondary DNSs, the (Extended SSID) ESSID, Gateway, lease duration time in seconds, network mask and if the SSID is hidden or not. Both views can be seen in figure 4.5 at middle and right, respectively.

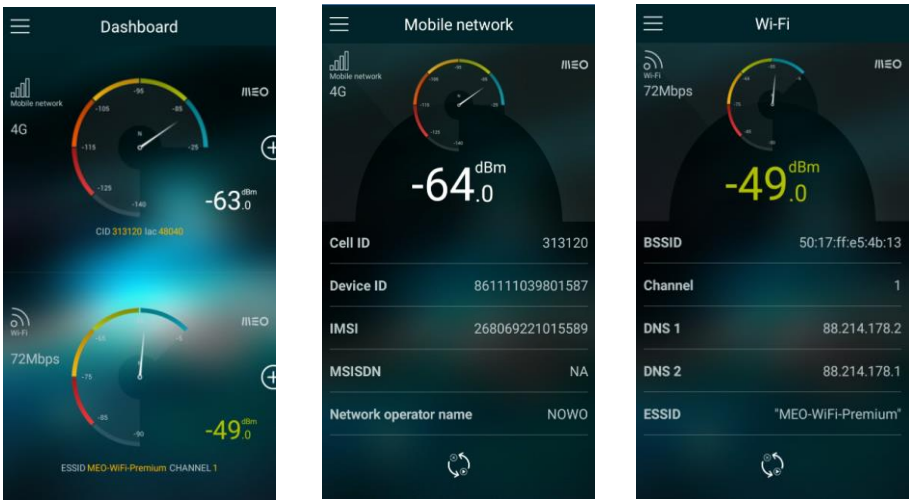


Figure 4.5 - ArQoS Pocket UI: Dashboard

The Wi-Fi information comes from the *WifiInfo* Android class added in API 1 ref. On the other hand, the mobile network information is being retrieved from multiple sources, from the *NetworkInfo* class to know, for example, if the roaming is active to the *CellIdentity* class, parsed for the active mobile technology at the moment, that provides parameters like the cell ID, cell location area code, MCC/MNC, among other parameters. Other Android classes such as *CellInfo* are called to distinguish the cell to which the device is currently attached with from the neighbors cells, the *CellSignalStrength* class to get specific parameters from a certain mobile technology, for example, Received Signal Strength Indication (RSSI) on 2G and RSRP, RSRQ or Channel Quality Indicator (CQI) on LTE [87].

4.2.3 Anomalies

In this section, the user can report an anomaly detected on the network. As can be seen on the left of figure 4.6, anomalies can be divided by 5 types: voice, internet, coverage and other. Those categories have sub categories already described in section 3.4. On the right side of figure 4.6, it's depicted a map that lets the user choose the location where the anomaly was detected by dragging the map. This is useful if a user detects an anomaly, but just reports it after a certain time. This map will open by default centered on the current location of the user.

The last step is optional and consists in attaching a feedback message explaining why the anomaly is being reported. The time and the geographical identification of an anomaly are automatically constructed and attached to it through the Android *Calendar* class [87] and by a

Google Maps API, which does a reverse geocoding, returning an address based on the geographical coordinates as input, respectively.

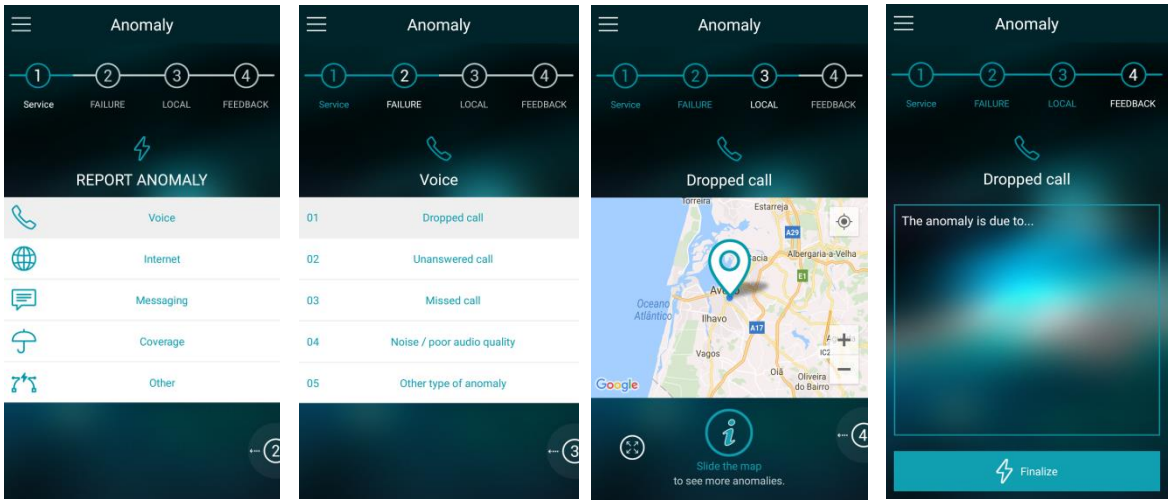


Figure 4.6 - ArQoS Pocket UI: Report an anomaly

For anomalies, tests and radiologs pages there is also a respective history page where the user can see all the inputs previously submitted. This history can be observed in list or map format on the left and middle of figure 4.7. It is also possible to filter information with a click on the dropdown, also illustrated on the figure. This allows the user to quickly find, for example, an Internet related anomaly. With a tap on a specific anomaly, also depicted at the right side of the figure below, the app shows a detailed information about it with the feedback reported previously and information about the local with a map below containing only that specific anomaly on it.

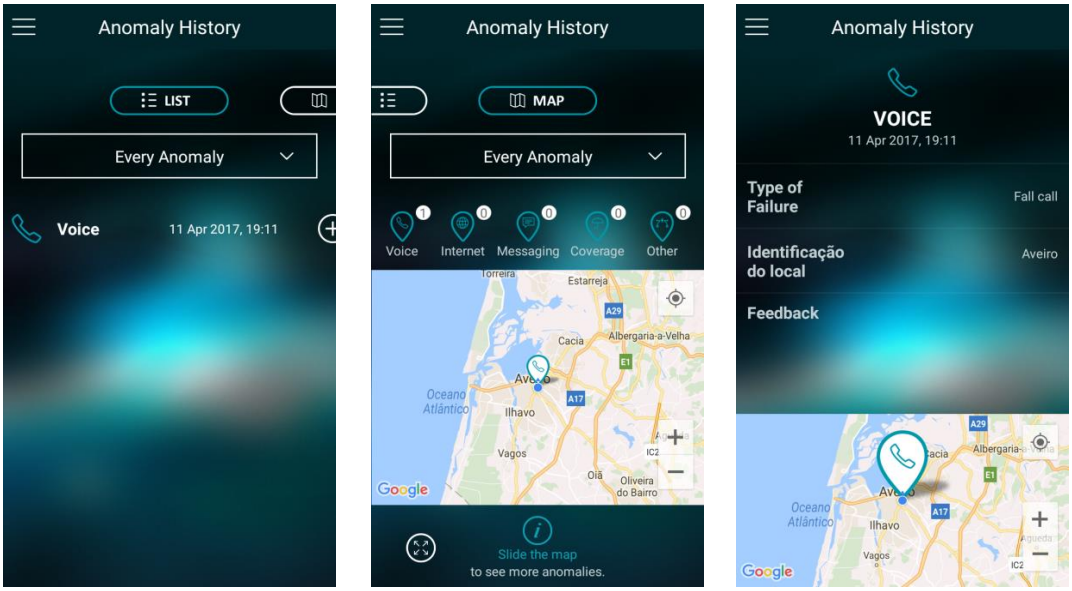


Figure 4.7 - ArQoS Pocket UI: Anomaly history

4.2.4 Tests

Several new service tests were implemented, not only to check quality of service metrics, but also to help troubleshoot possible problems in the network. These tests can be divided into two categories: intrusive and passive tests. The intrusive ones interact with the network in order to evaluate its real-time availability and performance. Passive tests are tests that run in the background, not noticed by the user, and do not need to directly interact with the network. Both tests retrieve important information about the surrounding networks and will be detailed in the subsection 4.4.

The ArQoS Pocket app tests voice and data. Regarding voice, answering voice call can be simulated, the call can be hung up and the in-call audio can be recorded to be analyzed after the results delivery to the management system, with PESQ [84] or POLQA [85] algorithms. Regarding, the internet access service, Internet Control Message Protocol (ICMP), Round-Trip Time (RTT), portal logins and many more useful tests are implemented.

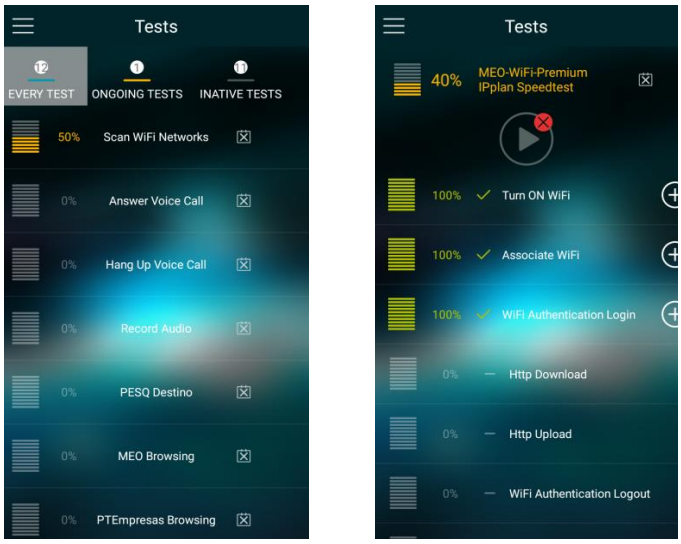


Figure 4.8 - ArQoS Pocket UI: List of tests and test details page

Regarding the test user interface, there are three lists: one with every test, a second one with the ongoing tests (tests that are currently running) and a third one with inactive tests. If a user clicks on a specific test a details page open. A test can have multiple associated tasks, as can be seen in figure 4.8. Besides that, in this test's detailed page the user has total feedback of what is happening with the application in the background, being able to check which tasks were already executed with success and the current progression of the test. The orange color on the test title

indicates that the test is currently running and when that is the case the “play” at the top right corner turns gray and not clickable.

Similarly to the anomalies history, it is possible to see the test’s results in a list or a map. In this page, the available filters on the *dropdown*, depicted on the left in figure 4.9, are the mobile network, Wi-Fi or “Every test”. Tapping a specific test will open a “details page”, as depicted at the middle of figure 4.9, showing the multiple tasks associated to it. Those tasks will appear with a success (green) or failure (red) symbol, and after tapping a specific task, a pop-up message appears, as depicted on the right side of figure 4.9, showing all the metrics and parameters retrieved from that specific task, as well as a brief snapshot from the mobile and Wi-Fi network state in the moment that the test was initiated.

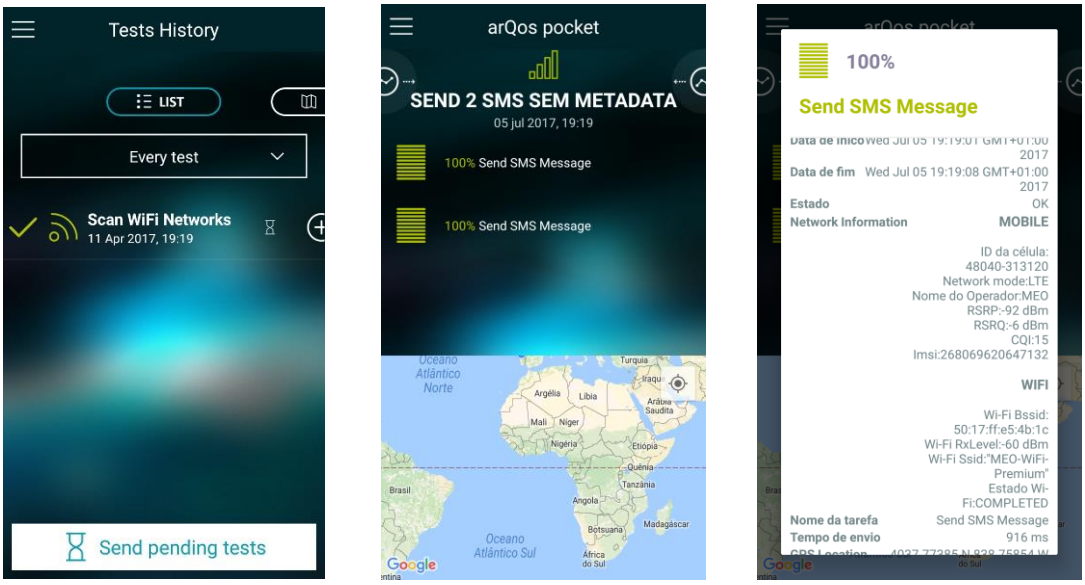


Figure 4.9 - ArQoS Pocket UI: Test history and task details pages

4.2.5 Radiologs

The Wi-Fi and mobile network signal strength and many other important parameters vary a lot either by internal or external conditions of the network. As explained in section 3.1, radiologs are snapshots of the mobile network state, periodically taken. On the other hand “snapshots” are instant radiologs taken at user’s request. Therefore, while radiologs are generated periodically without user interaction in a background task, a snapshot is taken once at a certain time. The interval time between each radiolog and the option for enabling or not radiolog generation is controlled by the user in the “Settings” page, which will be mentioned right away.

Some radiologs contain “Events”. This happens when an event is detected by the app. The events that are currently being picked up by the app are roaming state changes, Public Land Mobile Network (PLMN) changes, call establishment, call end, call setup and handovers or cell reselections dependent if a call is currently active or not, respectively.

The radiologs history page is depicted in figure 4.10 with, both the list and map view. On the right screen, it’s presented the full content of a radiolog with the geographical identification, cell id, network availability status, signal strength, the technology (HSPA in the example), the operator name, if roaming is active, MCC, MNC, Location Area Code (LAC) or Tracking Area Code (TAC) and neighbor’s cell information. The presentation of TAC or LAC depends if the device is using the LTE technology or not, respectively. Signal strength and some other parameters may vary with network technology too. In 2G the RSSI is retrieved and in 3G the RSCP. The RSRP, the RSRQ and CQI only make sence in 4G. All these parameters can be used to infer the QoS on the respective mobile network technology.

Wi-Fi network information is retrieved in the background too, but does not have yet a structure implemented and ready to be delivered to the management system. Nevertheless, both mobile and Wi-Fi information are being attached to all tasks in test’s history.

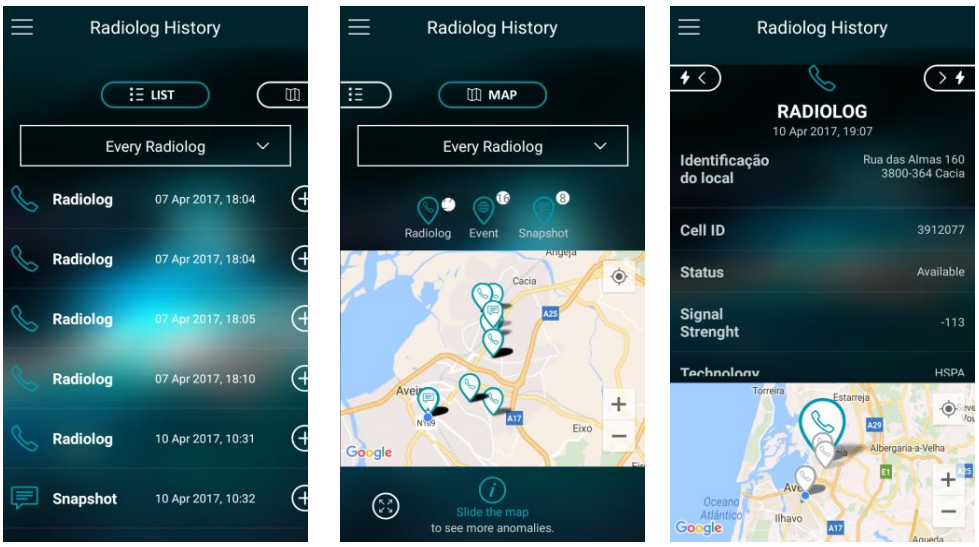


Figure 4.10 - ArQoS Pocket UI: Radiologs history and entry details page

4.2.6 Settings

The settings page, which can be viewed in figure 4.11, has relevant features that may change the application behavior. As was mentioned before, a user can record the radiologs and choose the time interval between those records, has the possibility to set a homepage and last, but not least, the possibility of changing the app language. There are three supported languages: English, French and Portuguese. The “Tests execution settings” are currently disable, since the app currently only has the possibility of automatically run tests triggered by date.

The app also has an “About” page explaining the purpose of the application and reserving all the copyrights associated to the last.

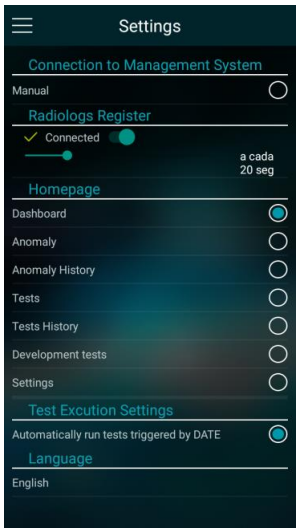


Figure 4.11 - ArQoS Pocket UI: Settings page

4.3 Management System Connection

All the tests results registered in every device running the application are sent to a management system that gathers all the data and analyzes it, producing views, helping with troubleshooting and facilitating the production of performance reports. These tests, as already said, may have multiple tasks, which have attached information about the WiFi and mobile networks, taken at the moment that the test started. Furthermore, it is also possible to manage all ArQoS probes and configure scheduled tests through this backend system.

The connection between the application and the management system is supposed to be automatic, but without a rooted phone, it is not possible to programmatically change the default associated APN. Therefore, for this initial phase, a slider in the "Connection to the Management System" group of *Settings* page was created. The user must drag this slider to initiate the connection with a management.

In the ArQoS pocket solution, the probe discovery starts with an *HTTP POST* internally called *"autodiscovery-process"* to the endpoint *"http://<addr>/southboundInterface/probe-announce?action=autodiscovery-process",* wherein the *<addr>* is the management system address, with the equipment type, serial number, IMEI, IP address and timestamp. This way the management system knows to who it is talking to. After validating the IMEI and IP address it sends a response to the probe's endpoint based on its IP address. Then periodic *keep-alive* messages are exchanged between the probe and the management system.

When the probe IP address change, for example, when the 4G connection drop to 3G, because of coverage issues, for example, there's also an "IP update" message similar to the *auto discovery* message, but with the old and new probe's IP address, on the message's body. It is important that both the management system endpoint and the pocket are reachable. For example, if the pocket connects to a Wi-Fi network, using Network Address Translation (NAT) the communication from the pocket to the management system and vice-versa directly will not be possible due to the IP address translation mechanisms. On pocket, the packets are being sent and received through the port 8080, therefore, this port must be open in the network firewall.

The management system answers to all these messages with HTTP responses: 200 OK when the *POST* is delivered with success or appropriate 400's or 500's error messages, when something that is not expected happens. A *Backoff* mode is configured on the application on these error cases. When a message fails to be delivered, the probe waits a time interval between 5s and 160s (maximum) and the backoff time interval is doubled with each failed attempt to avoid possible overloads on the management system side. Additionally, a random value between 0 and 10 seconds is added in order to minimize collisions between probes. An example of the previously described messages is depicted on a sequence diagram, in the figure 4.12.

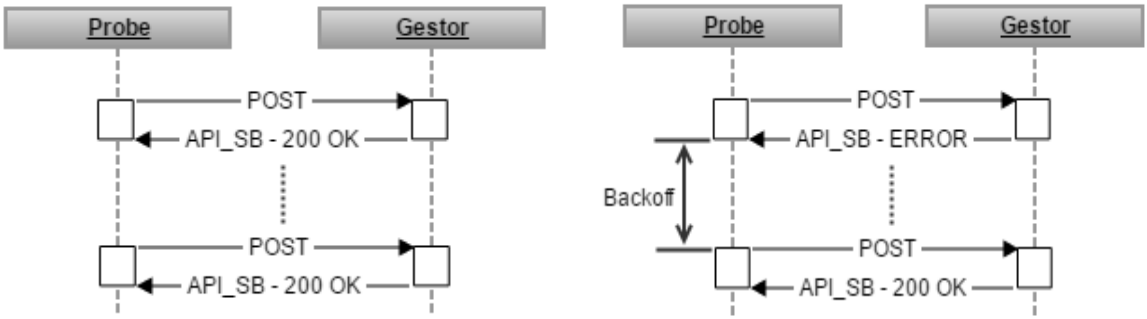


Figure 4.12 - Sequence Diagram: Management system confirmation messages

Using the management system, it is possible to remotely configure the probes, get their "hardware" information, get their occupation between two dates, check the configured tests and schedule tests. All of this is being done with *HTTP POST* methods, which goes a bit against the standard of the HTTP protocol, due to the *HTTP GET* and *DELETE* methods that it has. However, in this project only the application side was developed and those messages were already implemented and in use by the fixed and mobile probes. Therefore, to keep the consistency between all ArQoS probes the same approach was followed.

Test's results delivery was implemented as well, following the initial requirements. They are sent with a snapshot of the mobile and/or Wi-Fi network information and GPS location attached. After the results are delivered to the management system, they are deleted from the phone after 3 days. This time can be changed in the settings page or configurable through the management system.

The messages requests already supported and implemented are: i) the "Load Test Request", that loads a scheduled test into the app; ii) the "Cancel Test Request", which cancels a configured test; iii) a request to change the probe configuration; iv) a request to get the probe's hardware information, which is depicted in snippet 4.1; v) a request to get the real time probe's state and vi) a request to check the loaded tests on the probe.

```
{
  "msg_type": 2,
  "macaddress": "357810081051139",
  "modulo": 0
}
```

```
{
"probenotification": {
 "associatedResponse": {
  "bat_level": 15,
  "bat_time": -1,
  "call": "call0=0 - call1=3 tipo=1
in_out=1",
  "cell": "48040-50976",
  "country_code": "pt",
  "df": 84,
  "g_call_status": 0,
  "gps": "4037.8123 N 838.79114 W
1.930453 336.349030",
  "gsmroam": 0,
  "iccid": "89351060000611821057",
  "mcc": 268,
  "mnc": 6,
  "modem": 1,
  "pwr_status": 2,
  "rede": "rede=gsm=0 gprs=0 umts=0
lte=1 operadora=MEO",
  "rsrp": -91,
  "rsrq": -7,
  "sinal": -91,
  "sim_type": 0,
  "sync": "NETWORK",
  "tarefa": 0,
  "temperatura": 30,
  "teste": 0,
  "wifi_addr": "172.21.77.38",
  "wifi_bssid": "cc:d5:39:d1:d1:4c",
  "wifi_connection_state": 3,
  "wifi_dns1": "88.214.178.2",
  "wifi_dns2": "88.214.178.1",
  "wifi_gw": "172.21.76.1",
  "wifi_lease": 600,
  "wifi_mask": "255.255.254.0",
  "wifi_noise_level": -79,
  "wifi_quality": 2,
  "wifi_ssid": "MEO-WiFi-Premium"
 },
 "equipmenttype": "13",
 "ipaddress": "172.21.77.38",
 "macaddress": "357810082121048",
 "msg_type": 254,
 "serialnumber": "357810082121048",
 "timestamp": "20170710112831"
 }
}
```


Snippet 4.1 - Get probe's state request and response, respectively

The probe's response to this request contains the follow parameters: battery level in percentage, estimated remaining battery time in minutes (not implemented because it's not possible to get an estimative from the Android *BatteryManager* class [87] and, therefore, there is no estimate); call parameter indicating if there is a voice and/or a data call on the device; cell information (id and location); country code; free space on the device's internal memory in percentage; the internal state of a voice call (Idle, In Progress, Connecting, Active, On hold, Waitting, Calling, Busy); geographical coordinates; if roaming is active; ICCID, MCC/MNC; modem

state (Offline, Online); if probe is charging; mobile network technology and the associated signal values; from where is the probe synchronized; if the probe is running a task; device's temperature and lastly, all the Wi-Fi parameters described on the subsection 4.2.2.

All the requests have the same structure, using the respective "message_type", which is an integer that identifies the request, and the probe's MAC address or IMEI, in the pocket case. Optionally the module number can be an input as well, but in the case of the pocket probes, this field is considered always 0, even for dual SIM card devices.

The pocket response to these requests vary according to the received request. For example, for the previously mentioned "Loaded Test Status" request, the pocket response contains the configured testname, test ID, initial data, final date, test type, if the test as a recursion object, the recursion configurarion, the executed iterations, how many iterations were broken or skipped, the number of macros per iteration and the internal state of the test (if it's running at the moment, if it's scheduled, if was cancelled, if it was misconfigured, if it executed with success or not, if it was updated, among other internal states, they are fifteen at total.) An example of the described response can be observed in Appendix F.

The implemented notifications, both in the context of this project and of previous developments on ArQoS Pocket are *keep-alives*, the *auto-discovery* process, the *IP-update* process and the *results* process used when sending test's results information. To schedule a test on a pocket probe through the management system, the *POST* body message is similar to what is depicted in the snippet 4.2, preceded by two fields: "msg_type" and the IMEI of the pocket probe.

4.4 Network and Data tests

Tests like File Transfer Protocol (FTP) download and upload can measure parameters from the network like throughput, access time, the duration of the operation for a certain number of bytes transferred, among others. *MMS* and e-mail tests can test the network connectivity and infer parameters like the time spent delivering the message to the destination. Some of these tests were already investigated in order to know what would be possible to obtain from the Android classes, however they are not yet functional. As example, all the tests output parameters from a specific test, collected by the ArQoS NG probes are shown in the Appendix B.

On the other hand, SMS and voice tests were implemented according to the established requirements and matching what the mobile probes already do. Tests can be more intrusive introducing packets in the network or non-intrusive, retrieving information from the Wi-Fi or Mobile network passively.

Tests can run on-demand in the app, or be scheduled for a certain date by the management system. Tests that are already implemented with all the requirements and ready to be used with the management system are SMS tests (send and receive), HTTP tests (download and upload) such as MEO GO browsing and MEO Wi-Fi Premium login and a speedtest. Voice tests are implemented as well, but the test results are still not ready to be delivered.

Every test has its specific configuration. In snippet 4.2 it is given an example of the complexity involved in an SMS test. All the tests have an initial and final date, a test has a unique id, a test type, (for now we will focus only on on-demand tests (7) and scheduled tests (0)), state, priority and can have more optional parameters, such as a recursion object with an event and an interval in seconds for the repetition of the test execution, in the case of the snipper 4.2, it would serve to send an SMS every “x” seconds.

In this specific test, there is a more complex parameter called “data”. It has more parameters, specified in the ArQoS internal documentation, such as the task timeout, the destination number, message text, among others.

```
{
    "testeid": "T17",
    "dataini": "20170510105200",
    "datafim": "20200717133554",
    "modulo": 0,
    "testname": "Receive SMS DATE COM METADADOS",
    "macronr": 1,
    "testtype": 0,
    "state": 1,
    "priority": 1,
    "data": [
        "1|1||0|30|36|0|1234||"
    ]
}


{
    "testeid": "T16",
    "dataini": "20170531164600",
    "datafim": "20200717133554",
    "modulo": 0,
    "testname": "Send SMS DATE COM METADATA",
    "macronr": 1,
    "testtype": 0,
    "state": 0,
    "priority": 1,
    "data": [
        "1|1||0|30|12|0|967200007|ArQoS Pocket SMS
Test1234|1234|0|0|||"
    ]
},
```


Snippet 4.2 - Send/Receive SMS with metadata test configuration

All these tests can fail for diverse reasons, such as network conditions, malformed tests, exceeded timeouts, invalid parameters, etc. These failures are all identified and documented, as well.

• **Start a call with a custom APN:** In 3GPP data access networks, an APN is used to identify the packet data network that the user wants to communicate with or to define the type of service (e.g MMS, WAP). It also contains a mandatory network identifier that defines the external network to which the GGSN is connected and an optional operator identifier that defines the operator's packet domain (e.g. Internet-v4.mnc111.mcc222.gprs, mmsc.tmn.pt).

A list with 1370 APNs and its configurations was obtained from Android. In further investigations it was confirmed that Google has a *.xml* file on Android devices keeping this information, normally

under the */etc* folder. The choice of the correct APN is defined by the SIM card inserted in the device.

Changing the preferred APN requires the application to be a system application. As explained in section 2.5.3, these applications are located under the *system/app* folder and have some permissions that user apps don't have, allowing, for example, the change or insert of new APNs on the device's configurations.

• **Send DTMF tones:** In the past, dialing was achieved through a "disk" that generated a sequence of pulses. Dual-tone multi-frequency signaling is a substitute for that on keypad phones. When a user presses a key, a tone is generated, composed by a high frequency and low frequency components, based on the key pressed and is then sent to the central that analyses the signal, which detects the key that was pressed. The table 4.1 shows the correspondent combined frequencies for the various keys of the keypad.

Hz	1209	1336	1477
697	1	2	3
770	4	5	6
852	7	8	9
941	*	0	#

Table 4.1 - DTMF table

In the app it's possible to send a sequence of DTMF tones, for example, in order to interact with the voicemail service automatically (e.g. "200,3,,4,,1"). Unfortunately, on Android, it is not possible to inject these tones on the voice's uplink during an active call, but it's possible to do it when initiating the call. There are a few contributions, at the moment, to also allow this injection, during an active call on the *CallManager* Android class waiting for Google developers to review and accept it on "Android open source project".

Different tests are done passively in the app without user interaction, in other words, they are done only by listening the network itself gathering information or by analyzing other sources, as in the case of the Logcat reading.

• **Scan Wi-Fi Networks:** There are networks around us almost every time and, in big cities more than a few. Knowing which network has the best signal level in a certain location, the network's name, or which network has or not a security password allowing a free connectivity is important to users. The Android OS does this scan periodically as well, but this test was implemented, because there are valuable parameters that Android doesn't show us in his main interface and that are being retrieved at the execution of this test.

This test is done passively and periodically in the app, but if the user wants he can run this test on-demand, as well. It is obviously required that the device's Wi-Fi is turned on, so it is turned on at the beginning of the test if it isn't already. After that, the scan is performed and the parameters retrieved from this test are: SSID, BSSID, signal level, security capabilities, frequency, timestamp, channel bandwidth, center freq, etc.

• **Write LogCat to a file:** Android has several files where he dumps information about what is happening with the system at that moment. The location of this files are not standardized (i.e. some can be ROM-specific):

- */data/anr:* Some trace files are here (*Dalvik* writes stack traces on *ANR*, i.e. "Application Not Responding")
- */data/dontpanic* seems to be a standard location and contains some crash logs including traces.
- */data/kernelpanics* is another location but not having any "kernel panic" on the Android device yet means no content there yet too.
- */data/panic/panic_daemon.config* may point to other locations configured like */sdcard/panic_data/ or /data/panicreports* directory.
- */data/tombstones* may hold several *tombstone_nn* files (with "nn" being a serial number, increased with every new file). As tombstones are placed for the dead, they are created here for "processes died by accident" (i.e. crashed) and it is what is referred to as "core dumps" on Linux/Unix systems. However, not all apps create tombstones. This must be explicitly enabled by the developer.

Most of the logging is done on *tmpfs*, but with a reboot, this data is lost. Most developers usually use these logs to help troubleshoot problems or crashes in applications, but there is a lot more information there divided into 5 levels – verbose, debug, error, info and warning. Furthermore, there have been concerns about user's privacy, because despite we can write messages to logfile to use them for debug purposes, it is also possible that SMS/MMS content,

contacts information or e-mails be written, as well [22]. In snippet 4.3, it's seen an example of private information that can be used by hackers, in this case, user's current location:

```
I/ActivityManager(526): START u0 {act=android.intent.action.VIEW
dat=google.navigation:///?ll=49.872600,8.636720&q=Alexanderplatz,Berlin&
token=FdjAIdMMmDACmJjSWej3C9RzGzM32OzZezHw&entry=w&opt=cmp=com.google
.android.apps.maps/com.google.android.maps.driveabout.app.NavigationActivity} from
pid 22056
```

Snippet 4.3 - Logfile - information about user location [22]

Accessing a Wi-Fi network exposes the SSID or the associated MAC address. Other examples can be given, for example, it was found the Bluetooth address written in clear text and after the opening of a specific app, the log file contains the package name of the selected app. Although all these examples were found in clear text, there is also information that is sanitized. In fact, there are some ways that explicitly prevent some information to be written in the logfile. If applications call the toSafeString() method URIs are sanitized before they are written into the "log file", because of that URIs started by "tel:", "smsto:" or "mailto:" are written as "xxxxxxx". The Snippet 4.4 shows an example of sanitized information written into the logfile.

```
I/ActivityManager(522): START u0 {act=com.android.phone.SIP_SELECT_PHONE
dat=tel:xxxxxxxxx flg=0x10000000
cmp=com.android.phone/.SipCallOptionHandler (has extras)} from pid 767
```

Snippet 4.4 - Logfile information: sanitized information

This can still be a problem nowadays in rooted phones, however, since Android version 4.1 Google doesn't allow to read log entries from other applications anymore. Building an application and dumping the log file in an unrooted device with a version higher than 4.1, will only log messages from that specific application.

All of this information has been tested and to read logs on rooted devices, permissions need to be granted, not only in the *"Manifest.xml*", but in real time via shell commands too. After that, the app just needs to write the log file in a background process to a *.txt* file. In the next chapter, Evaluations and Results, a section for the Android logfile analysis is detailed, with useful information encountered, that can be parsed and used in the future on the ArQoS Pocket solution.

• **SMS Tests:** The SMSs are sent through Android Intents, which is “an abstract description of an operation to be performed” and set *with help of the* *SMSManager* internal Android class [87]. The input parameters of the “Send SMS” test are: the destination number, text message, trailer text, enable/disable metadata, SMS Centre number and the SMS encoding.

With this test is possible to retrieve the time spent delivering the message to the SMSC, although the total end-to-end SMS time is only possible to calculate if the “Receive SMS” task is configured on a test on another probe. This task waits for an SMS which is validated by text or by trailer, as explained before, retrieving as output parameters the SMS text, the sender number, the SMSC number, SMS encoding, the end to end message delivery time and the time spent waiting for the received message since the beginning of the task. A sequence diagram is depicted in figure 4.13, containing the workflow of the messages.

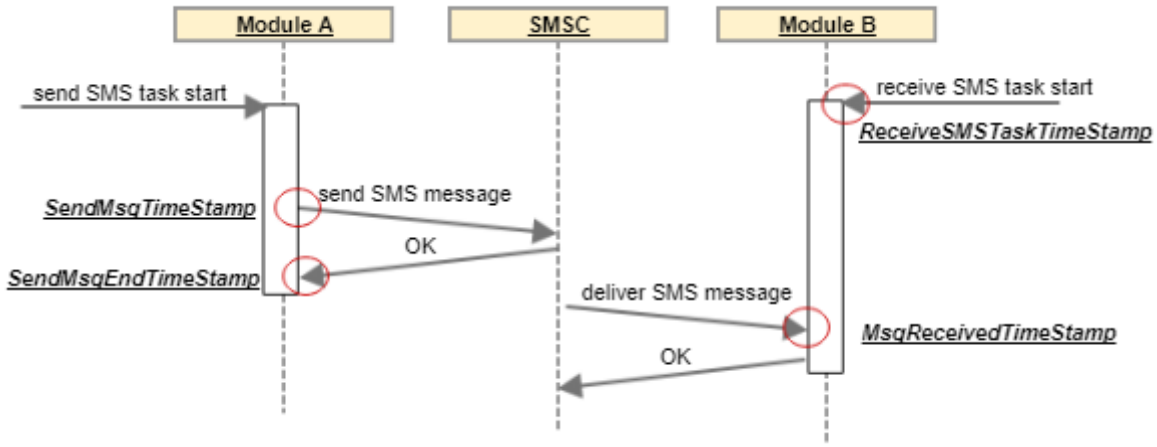


Figure 4.14 - Sequence diagram: SMS test KPIs

• **Voice Tests:** The voice tests required for the application are: make a voice call, answer or reject a voice call, hang up a voice call and play audio in an active call. The “Play Audio” test was discarded, because it is not possible, due to security issues, to reproduce an audio file in an active call on Android [60].

Make, answer and hang up a voice call, were already implemented on the previous version of the application. However, only the “Make a voice call” was functional when tested. This happened, because the “answer” and “hang up” task’s implementation used a breach/gap on the Android security. Both tests were implemented simulating a key press in headset connected through Bluetooth.

This breach only worked in a few older devices, nevertheless it was fixed in devices with Android versions greater than 6.0. As these voice tests are a very important feature for the final solution, a new approach was implemented, overcoming the described problems: the simulation of the user's touch on the screen through ADB commands, in order to answer/hang up an incoming voice call.

It was verified that if an app is running in the foreground, the dialer app is not called instantaneously, instead a notification appears on the top of the screen. For this reason, it is checked if the phone is locked. If it isn't, a tap on the notification is simulated, on the other hand, if the phone is locked, a slide to the right or left, is simulated on the dialer app in order to answer or reject the incoming call, respectively.

A rooted phone is required for these tasks to work properly, since calling the commands to simulate a tap or slide in the screen at runtime, via ADB require administrative permissions on the device. Furthermore, as this solution is based on screen coordinates, it is only functional on the Samsung Galaxy S7, notwithstanding and due to the importance of these voice tests, this was the adopted approach in order to get these tasks to work again.

Chapter 5

Evaluation and Results

In this chapter, it is presented all the obtained results from the tests made in order to validate the proposed solution. All results are discussed and compared with the already existing results from fixed and mobile probes, already deployed all over the world. Considering the requirements presented in chapter 3, the effectiveness, metrics, performance and behavior of the solution is analyzed in different conditions.

5.1 Probe's deployment scenario

The ArQoS Pocket solution is intended to be used as a stand-alone application. However, it was considered the existence of other deployment scenario for this solution, which aims on integrate the pocket probe with the already existing fixed and mobile probes.

These two ArQoS probes communicate using the Android ADB through a USB connection, and the time synchronization needed is guaranteed by an NTP server running on the mobile probe by which the pocket probe synchronizes.

A prototype design of these probes integration is depicted in figure 5.1, with a case containing two smartphones and a mobile probe. For this specific scenario, the smartphones that will be used are the Samsung Galaxy S7 with the Android version 7.0.1, which is the primary target phone of the solution.

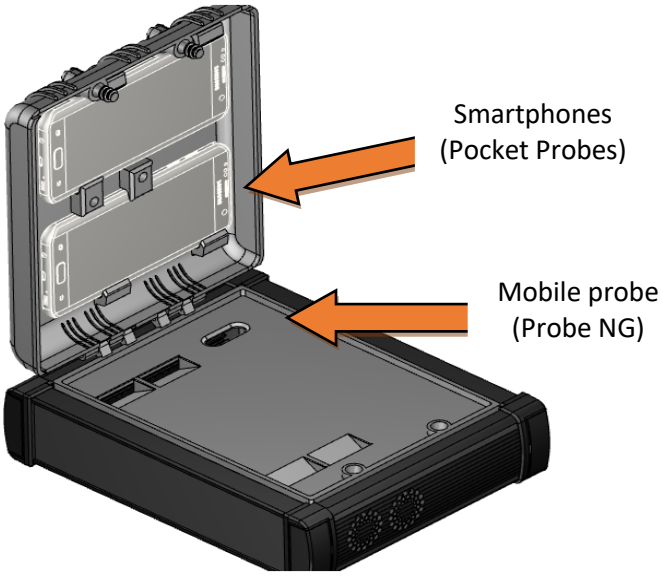


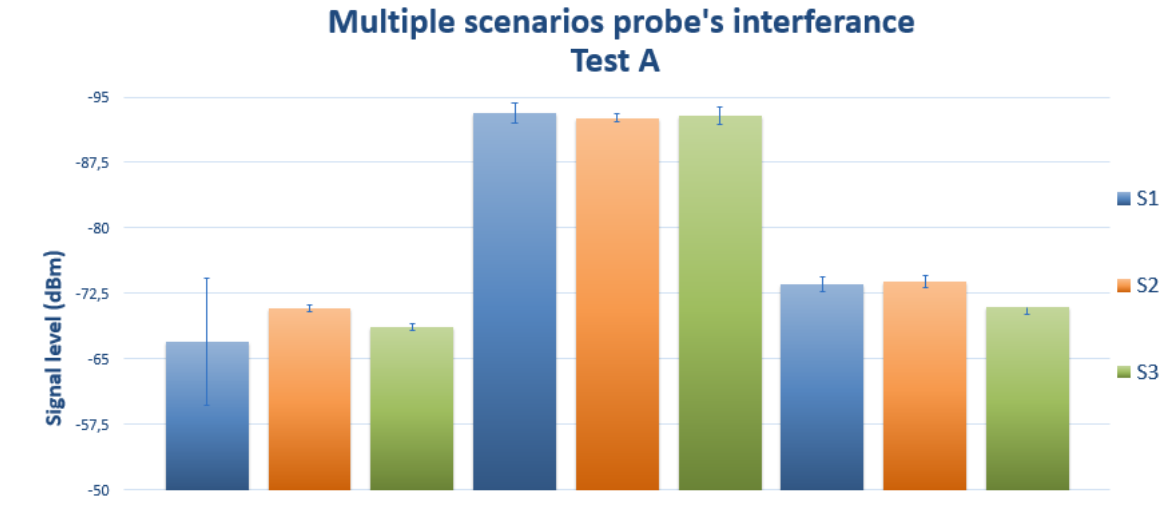
Figure 5.1 - 3D prototype of open probe with the Pocket solution integration without wiring

It is important to study the pocket/mobile probes integration and analyze the possible interferences generated by the mobile probe, on the signal strength signal captured on the phone from cell towers, in the different technologies in order to conclude which deployment scenario is the best to be adopted and to validate the prototype drawn.

There were three possible scenarios to be considered:

- S1 - the pocket probe far away from the mobile probe (no interferences).
- S2 - the pocket probe on top of the mobile probe with a cover dividing them.
- S3 - the pocket probe on top of the mobile probe without any cover dividing them.

Figure 5.2 takes into consideration the average signal level, with a 95% confidence interval, signal strength level captured by a Samsung Galaxy S7 in the 3 scenarios previously described, on 4G (LTE), 3G (HSPA) and 2G (EDGE) on the left, middle and bottom, respectively. The obtained results are analyzed below, considering 20 measures with 8 seconds of interval each. Furthermore, the figure 5.2 also presents a detailed table with the results statistical measures (mean, standard deviation and confidence interval).

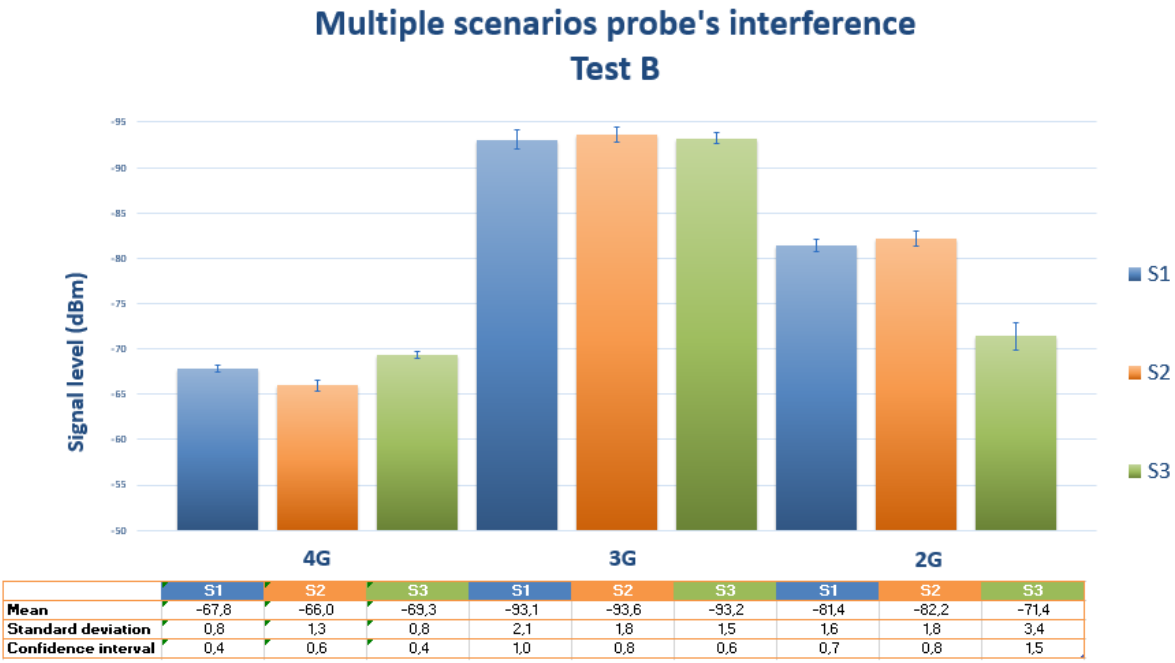


	S1	S2	S3	S1	S2	S3	S1	S2	S3
Mean	-66,9	-70,8	-68,7	-93,2	-92,7	-92,9	-73,6	-73,9	-70,9
Standard deviation	16,9	0,9	0,9	1,1	1,1	2,3	1,9	1,5	1,7
Confidence interval	7,3	0,4	0,4	0,5	0,5	1,1	0,8	0,6	0,7

Figure 5.2 - Test A: Mobile probe's interference on pocket's radio signal strength

By analyzing the obtained results, it is seen that on 4G, the best-case scenario was the one with the pocket probe far away from the mobile (S1), because the closer the signal level measured is to zero, the better it is. Using 3G the best-case scenario was the pocket on top of the mobile probe with a cover dividing them (S2) and lastly, using the 2G technology the best scenario was the scenario where there was not a metal cover dividing both probes (S3). On a deeper analysis and plotting the confidence interval, it is seen that the results between the case scenarios are similar, although the differences of the signal strength values, as absolute.

To try explaining these results was performed a similar test, using the same mobile technologies and deployment scenarios, but at a different time of the day. The results obtained are analyzed below, in figure 5.3.



	S1	S2	S3	S1	S2	S3	S1	S2	S3
Mean	-67,8	-66,0	-69,3	-93,1	-93,6	-93,2	-81,4	-82,2	-71,4
Standard deviation	0,8	1,3	0,8	2,1	1,8	1,5	1,6	1,8	3,4
Confidence interval	0,4	0,6	0,4	1,0	0,8	0,6	0,7	0,8	1,5

Figure 5.3 - Test B: Mobile probe's interference on pocket's radio signal strength

Comparing the overall values, of the signal level retrieved from the network, with the ones obtained in the first test, it is seen that these signal values are worst in all the deployed scenarios. This can be explained, because the time of the day that Test B was performed coincided with the end of the work day, a known busy hour for the mobile network.

Comparing the deployed scenarios side by side, on both tests A and B, it was verified that the best-case scenarios results didn't match at all. Even over, the best-case scenario on Test A becomes the worst one on Test B. The same inconsistency between tests A and B happened in 3G, as well. Only on 2G the best-case scenario was maintained between tests. Therefore, it was concluded that the signal strength level is more affected by external factors like the number of people using the network at a specific moment, the in-building penetration or the multipath environment created due to physical obstructions, than that the position of the pocket probe in relation to the mobile probe [79].

Taking these results in consideration, the prototype scenario for the ArQoS Pocket solution integration, depicted in figure 5.1, was validated for production.

5.2 Application tests

It's very important to test the application overall functionalities and performance, because it affects directly the user solution perception. In this context, the application compatibility in different devices with different Android versions was tested and the application responsiveness was calculated by measuring the times between transitions of the application pages. Lastly, the application battery consumption was also measured.

All these factors have a huge importance in every Android application, because a multiple device compatibility allows more devices to run the app, and therefore, potentially more users using it, as well. Higher times between pages' transition in an app makes the user uncomfortable and waiting, negatively affecting usability. Furthermore, considering the use-case scenario where the solution is used as a stand-alone app, not connected to any probe, and therefore not charging, it is not desirable that the app consumes much of the user's phone battery.

5.2.1 Devices compatibility

In the initial application development, it was used a Xiaomi Redmi 3S. However, in order to test VoLTE and VoWiFi features, the primary target device of our solution became the Samsung Galaxy S7. Both devices specifications can be seen in the Appendix C.

It is important to test the application compatibility with other devices to ensure that users, which use the application as stand-alone, in other others, without the management system interaction, have a good experience, as well.

The application compatibility was tested, by creating Android virtual devices and by executing the app on different devices, with different Android versions, API levels and/or different screen sizes. This allowed detecting possible anomalies with both, application features or the UI.

Our solution targets the Android version 6.0.0 and above, but it was verified that multiple devices with a lower version could still run the app, and perform all its associated features. In the Table 5.1, it is presented detailed information with the devices tested.

Model	Emulated (Yes or No)	Android version	Scren size (inch)	API level	Compatible (Yes or no)
Nexus One	Yes	Android 2.3.3	3.7	10	No
Nexus One	Yes	Android 4.3.0	3.7	18	No
Nexus One	Yes	Android 4.4.2	3.7	19	No
Nexus One	Yes	Android 5.1.0	3.7	22	Yes
Nexus S	Yes	Android 5.1.0	4.0	22	Yes
Galaxy Nexus	Yes	Android 5.1.0	4.65	22	Yes
Nexus 4	Yes	Android 6.0.0	4.7	23	Yes
Nexus 5X	Yes	Android 7.1.0	5.2	25	Yes
Xiaomi Redmi 3S	No	Android 6.0.1	5.1	23	Yes
Samsung Galaxy S7	No	Android 7.0.0	5.0	24	Yes

Table 5.1 – Compatibility test: Which devices run the application

At the beginning, the same device (Nexus One) was maintained, varying only its Android version and API level. This allowed to detecting the minimum Android version able to run the application, which is as shown above, the Android 5.1 (Lollipop).

After, to try induce visual bugs on the application, it was varied the device's model and screen sizes, maintaining both Android version and the API level used. Still, no issues on the UI were found, being the devices able to run the application.

Lastly, it was also tested by emulation the new Android 7.1.0 (Nougat) with the API level 25 in a Nexus 5X, which has the larger screen with 5.2 inches. Even so, no visual problems were detected and all the features in the application that did not needed a SIM card to be tested were, still functional.

Although the code targets the API 23, many devices with lower API levels could still run the application with no issues, which is really satisfying. It was also verified that Android devices with a version below than 5.1, could not run the application due to needed manifest permissions. Furthermore, was not even possible to install the application in devices with the API 14 or below.

5.2.2 Pages load times

The second performed test intended to test the application response times. This is a key test to evaluate the usability perceived by the user of the ArQoS Pocket solution. In order to simulate a more real user interaction with the app, the pages' load times were measured on the first page access and after the first attempt, which had help from the memory cache.

It was found that most of the transitions between distinct pages in the app are smooth, being executed in milliseconds. The operations that require more time are done in the background never locking the UI. Taking this into consideration the pages considered in this test were: the application homepage, the anomalies history and the radiologs history pages, as they are the ones that can take the most time to load, due to the potential heavy content that they could contain to process.

The first page is seen after an initial splash screen. This splash screen vanishes when a callback is sent, saying that the UI is updated and ready to be displayed. Both anomalies and radiologs history pages, can be seen after a tap on the respective icons at the sliding menu, which is depicted in the figure 4.2. The results presented on both tables 5.2 and 5.3 consider the first page, as the "Dashboard" page, since this can be changed on the Settings page. Furthermore, they consider the average time of 20 measures, measured calculating the difference of times recorded in the logfile from the beginning of the transition to the end.

As described in section 4.1, the application used the Android *SharedPreferences,* to internally save the anomalies results and the "radiologs" feature was developed following the same approach. However, it was considered that the correct approach was to save all these results in a database, similarly to what was happening already as with tests results.

In the table 5.2 are detailed the page load times, obtained before the database implementation for anomalies and radiologs history pages with 2 and 180 entries, respectively. On the other hand, in the table 5.3 are detailed the load times for the exact same pages, both with 2 and 180 entries each, but after the database development for these features.

Pages load time (first time)			**Pages load time (with cached)**	
Initial page	Anomalies History	Radiologs History	Anomalies History	Radiologs History
2,6s	2,2s	2,7s	0,8s	1,5s

Table 5.2 - Performance test: Pages load times before the database implementation

Load Page Time (first time)				Load Page Time (with cache)	
Initial page	Entries	Anomalies History	Radiologs History	Anomalies History	Radiologs History
2,6s	**2**	1,5s	1,5s	0,6s	0,7s
-	**180**	1,6s	1,7s	0,7s	0,9s

Table 5.3 - Performance test: Pages load times after the database implementation

We consider that waiting for the callback in the splash screen to show the first page user interface with the correct details to the user is the best option. If the pages have never been opened once, the delay is higher than the after opening with cache, as expected.

The change applied in order to store these objects in a database, instead of the *SharedPreferences,* was really useful, allowing to improve the application performance, by lowering the times previously obtained either with or without cache and with less or more entries. This can be explained because *SharedPreferences* is a key/value store, which makes it difficult on reading large structured data. Instead, *SQLite* is designed to manage and search large amounts of same structured data, querying only sub sets of the data that matches certain criteria.

By analyzing the results, it's also seen that the radiologs page with 180 entries, lowered the load time (~37%) on the first access and (40%) on further queries or history visits. On the other hand, the anomalies history page with 2 entries only had its load time reduced (~32%) on the first access, and (25%) in subsequent accesses. This can be explained because, although *SQLite* is more efficient on handle larger structured data, if the amount of data is not so big, as is the case of the anomalies, the *SharedPreferences* may provide a less complicated processing that could cause less overhead on simple accesses. An example of this can be seen on anomalies history page loading with 2 entries, which have similar load times: 0,8s with the *SharedPreferences* and 0,6s with the *SQLite* database. Even on these conditions, the SQLite database achieved a lower page load time.

The higher load times on the history pages in comparison to the other application pages may also be due to the Google maps call, and the associated processing, for example, the markers place at the correct geographically position. Notwithstanding the times presented, especially after the first consultation are very satisfactory, with the app performing all operations in less than 1 second.

5.2.3 Battery consumption

The last test made measured the app's battery consumption while taking radiologs and, executing tests like HTTP Download or sending SMSs in the background, repeatedly. For this test, the radiologs were enabled on the Settings page, registering the phone, a snapshot of the mobile network, every 30 seconds. Furthermore, it was scheduled the sending of an SMS every minute, during four hours in order to simulate a higher battery consumption use-case.

It was verified that the registration of radiologs stopped after one hour. With further investigation it was found that from Android Marshmallow (version 6) and beyond, a feature called *Doze* was added on Android, aiming on helping that the device's battery last longer.

Doze mode intelligently shuts down processes, saving the device's battery when it is not in use for a certain time. "Google had claimed to achieve two times longer standby time on Nexus 9." [68] with *Doze* on the device. The problem with *Doze* is that there is nothing that the user can do about it, there are no switches or settings to toggle, if the device has an Android 6 or higher version *Doze* is just there, slumbering the device when it's idle.

As said, *Doze* runs almost invisibly in the background. For it to really kick in, the device must be with the screen off and not charging. This was confirmed several times, running tests with recursion, in other words, scheduling the same test repeatedly. Even with the USB cable plugged in, was verified that *Doze* has a higher priority than the Android *AlarmManager*, which is being used to wake the phone at a certain time, in order to run a specific scheduled test. This causes the phone to not wake up and, therefore not execute the specified task at time.

The results, depicted in the figure 5.4, were obtained from the *Battery Historian* [86] tool from Google, which analyzes the battery consumption of a device, using Android "bugreports" files. For four hours, the phone was sending SMSs and taking radiologs.

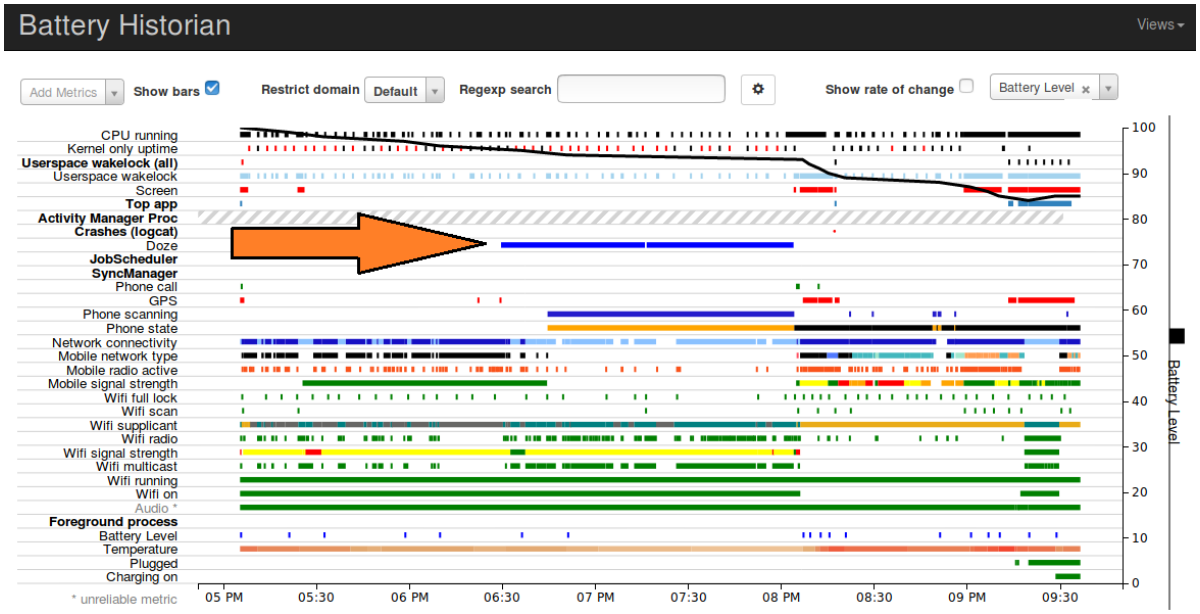


Figure 5.4 - Performance Test: Battery Consuption

The total battery consumption of our solution over during these four hours was 0,21%. This can be explained, because the radiologs registration stopped and several SMSs were not sent due to the *Doze mode,* as depicted in the figure 5.4 with a blue color from the 06:30 p.m. to 08:00 p.m.

When the screen is turned off and the device is not charging, it was verified that, 9 minutes between each alarm, is the minimum time for the *AlarmManager,* to wake up the device, at the exact expected time. As this issue causes a big impact on the test's scheduling of our solution, a common error message was documented for the pocket probes, saying that the task was not executed due to "snooze".

Another issue that was continuously happening, was the fact that when a test with a lower recursion interval was scheduled, the *AlarmManager* woke the phone 2 or 3 seconds later than expected, causing the last test iteration to not run. For example, a test that should send four SMSs sent instead only three. This issue had to do with the same Android restrictions, previously explained. The solution adopted in order to run all iterations on a test with recursion, was not to let schedule tests with a recursion interval lower than 30 seconds.

With this test, it became clear that the battery consumption will not be a problem in our solution due to the Android *Doze* feature, which made us to adapt our solution according to its restrictions. We consider that the changes and conditions applied to answer the issues previously described, made our solution more robust in the sense that now, all the iterations of a test are seen in the test history page either if they were, or not executed for some reason. Either way, the

user always has a feedback message on the respective task details page in order to understand what happened with that certain test iteration.

5.3 Tasks evaluation

The tasks performed by the application are a very important requirement, that needs to be functional and well implemented, independently of the scenario where the application is deployed.

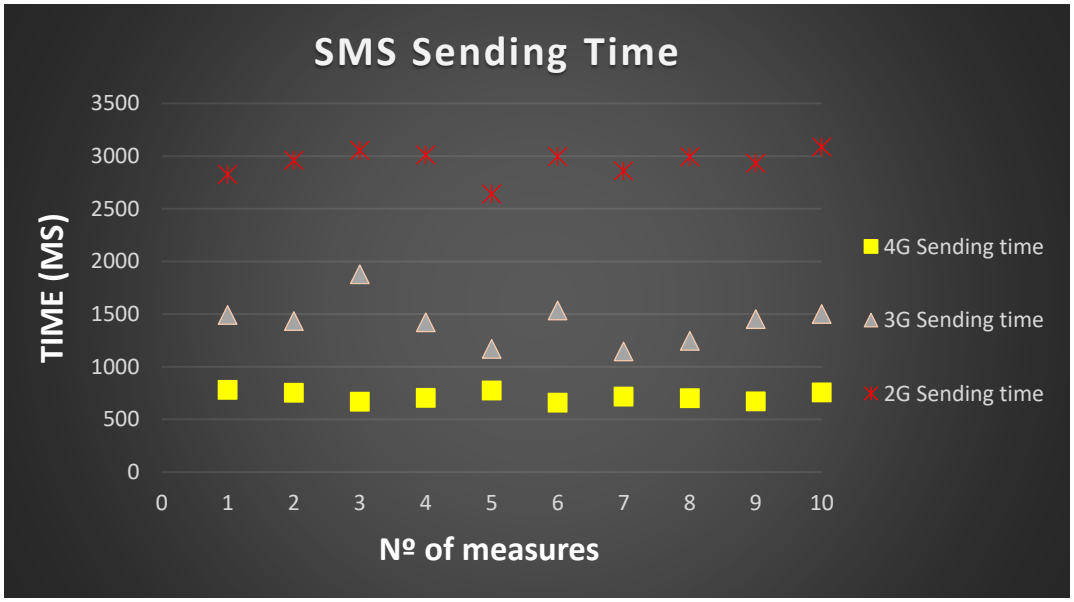
There were already input and output parameters defined for each test identified by the ArQoS NG probes. Is being collected, as many parameters as possible, from the ones already documented from the other ArQoS probes and, since this is a different ArQoS solution, running in an Android terminal, new parameters are being retrieved and documented, as well. The management system needs to evolve, in a close future in order to have support and integrate these new values/parameters from the ArQoS Pocker probes.

The tasks validated in this section are the ones already mentioned, in section 4.2.3. These tasks have their specific metrics, whose examples can be seen in Appendix B. As already mentioned, since this is a solution running on a real Android terminal, not every parameter in the list could be obtained. In the app, these parameters/metrics can be seen at the task's details page, as depicted in figure 4.9.

5.3.1 SMS tests

The SMSs sent through ArQoS Pocket should have similar behaviors and times compared with the ones sent/received by the mobile probes deployed, already on the network. Knowing beforehand these measures, allows a quicker validation of the experiment.

To validate the SMSs latency measured by the new application, the values were compared with the delivery and sending times measured on the ArQoS NG probes. Figure 5.5 depicts the time that an SMS spent until it reaches the SMS Center, as sending time and to the destination number as the delivery time, varying the mobile technology of the devices. The tests were performed in the same operator's network.



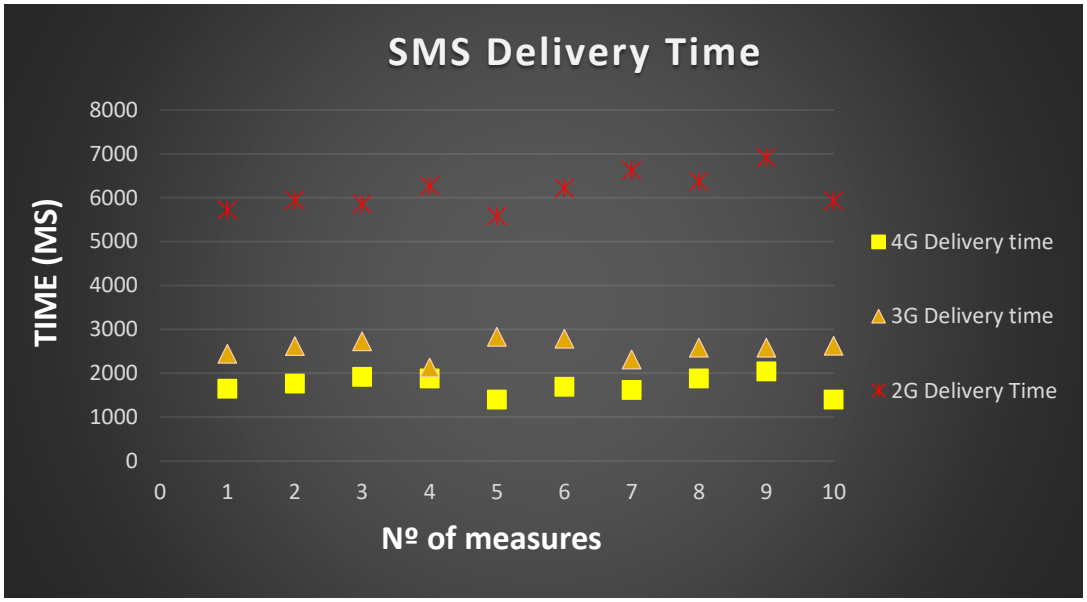


Figure 5.5 - SMS sending and delivery times on multiple technologies

The obtained results, depicted in the figure 5.5, meet the expectations, as when it goes from 4G to 2G, the average times of delivery and sending an SMS, increase. Furthermore, they also match the end-to-end times measured in the other ArQoS probes, which were 2.1s using the 4G LTE technology, 2.3s using 3G and 6,3 seconds on 2G (EDGE).

Table 5.4 details the pocket measured average times, previously depicted in the figure 5.5, as well as, the standard deviation and a 95% confidence interval of the values.

	4G		3G		2G	
	Sending time	Delivery time	Sending time	Delivery time	Sending time	Delivery time
Mean	718,4	1729,2	1427,3	2568,7	2935,8	6146,9
Standard deviation	44,8	216,3	211,9	216,7	131,2	418,6
Confidence interval	27,8	134,0	131,3	134,3	81,3	259,4

Table 5.4 – SMS Test: Statistical measures

When the phone is using the 4G LTE technology, it also maintains an HSPA connection in background for Voice calls and SMS. When a call is made, the device falls back to 3G or 2G and the LTE radio is disabled to save power. However, the obtained results shows that both the sending and delivery times are lower in 4G than 3G. This is explained, because contrary to voice calls, the SMS is sent very quickly in the signaling plane, so the LTE radio stays on. Taking this in consideration, the SMS over 4G has improvements compared to 2G/3G networks, due to the lower latency and more efficient transaction on the SGs interface of the first.

Sending an SMS from a different operator network or between different network technologies increases these times, which meets the expectations, because the path that the SMS must travel in both cases is longer. Other factors that can influence these times are: location issues, network traffic or mobile device issues. If a device as low battery level, that may also impact negatively the message delivery time, as in some devices the CPU (Central Processing Unit) is slowed down in order to the device last longer, which can increase the SMS latency.

5.3.2 Radiologs

To validate this feature, the device was taken on a drive-test through the city, with the application running in the background. Radiologs records were already tested, but to catch events like active roaming, handovers or cell reselections, there was the need to travel with the device. For this test radiologs were taken with a periodicity of 20 seconds.

Figure 5.6 depicts a map view from the “radiologs history page” after the test drive execution, with many radiologs taken periodically. It is possible to see 178 radiologs that were registered during the tests. It is also possible to see, 16 events containing captured at different locations. Unfortunately, it was not possible to take a mobile probe along with the pocket probe to compare both results and events measured, because they were all being used for development.

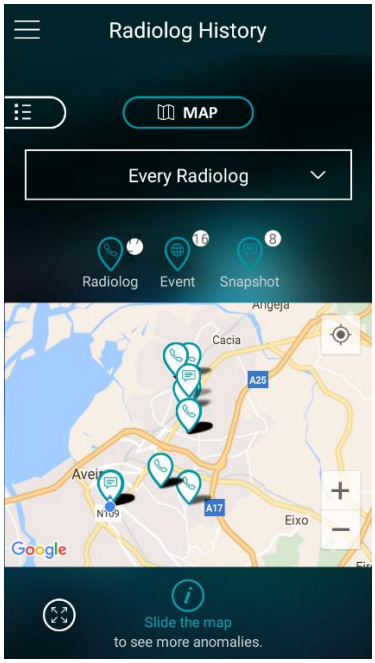


Figure 5.6 - Radiolog map view after drive-test

By opening those events it can be seen that, they are radiologs, taken apart from the periodical ones, capturing specific occurrences on the network, for example, automatic cell reselections done by the device as it moves to a different location. During this test, 14 cell reselections and 2 handovers events were captured. Detailed information about the content of these events can be found in figures 5.7 and 5.8.

Figure 5.7 - Event: Cell reselection

Another test was made in this regard in order to try capture and validate a *PLMN* change. For this, the SIM Card was withdrawn of the device and another SIM from a different operator was placed. The result of this procedure was, as expected, an automatic event triggered with the details presented in figure 5.8. The full radiolog content can be seen in Appendix C, containing radiolog common parameters such as the timestamp, GPS location, IMEI, among others, an object with the information about the mobile network technology, the neighbor's cell information and lastly, the identification of the event captured.

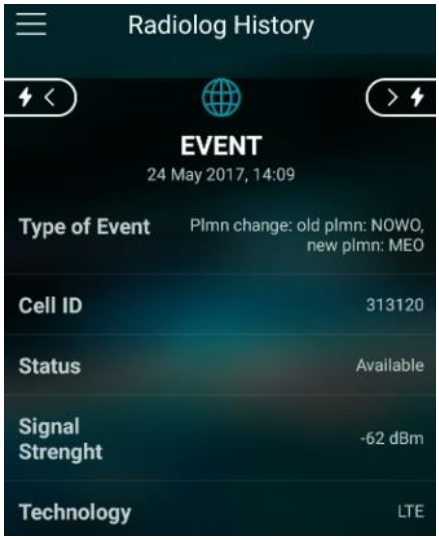


Figure 5.8 - Event: PLMN change

5.3.3 Logfile Analysis

The Android logfile can be useful in order to monitor passively the user interaction, not only with this solution, but for every feature on the phone, making possible to detect the user usual behaviors and preferences, as well as, events triggered by the Android OS.

With the objective of determining what the logfile has to offer, specific tests were made to discover more information from the network than the one already retreived from the internal Android classes. Furthermore, it was analyzed the log informations from OTT apps like Facebook or YouTube can be retrieved, as well.

The first test made started by placing the mobile phone in an anechoic chamber, which does not let the signal pass from the outside and, therefore forces an active call to drop. It was found the

presence of interesting network information such as error causes for dropped calls. When the call drops due to coverage issues a message with a "protocol_error_unspecified" is displayed into the logfile. This information can be parsed in the future and may substitute the QoE oriented anomalies report page in the app.

Mobile network information is extracted from internal Android classes like *CellInfo*, *CellIdentity*, *TelephonyManager*, among others. It was found that the logfile contains calls to the *SignalStrength API*, similarly to what is being done in our solution, containing the signal level in dBm's, LTE Asu level, among other parameters [87].

Time was also invested in observing what more can be inferred from the log about calls, phone integrated apps like dialer and lastly, features like battery or Bluetooth. Periodic appearances from the Android Battery Service are displayed, containing the battery level, status, health, voltage, temperature among others battery associated values.

Interactions with the device like pressing the "Home" button could be seen, as well, searching for the "*performOnHomePressed*" tag. Having this in mind, it was verified if the characters typed on the phone dialer, were being saved in the logfile likewise, which would be a major security problem. Fortunately, and meeting the expectations, only the press and release events could be seen. These actions messages can be seen below, in snippet 5.1.

```
I/InputReader: Touch event's action is 0x1 (deviceType=0) [pCnt=1, s=]
....
I/InputDispatcher: Delivering touch to current input target: action: 0x1
I/InputReader: Touch event's action is 0x0 (deviceType=0) [pCnt=1, s=0.1200 ]
```

Snippet 5.1 - Logfile information: Dialer touchscreen

Wi-Fi associations, disassociations and Bluetooth information were also tested, and lots of information was encountered about the processes that the phone does in the background such as, Bluetooth bond state changes, when the discovery process starts, the connection status, when Bluetooth is turned off, and the internal broadcast states numbers for those transitions. The extracted information can offer an overview of the internal protocol process.

Other great point of focus throughout the project was the study of the audio on the phone calls. Android has three phone states *IDLE, OFFHOOK* and *RINGING.* These states could be seen on the logfile, along with other state updates, notifications, requests and more interesting yet, call ends, drops or hang ups timestamps. In the snippet 5.2, it is shown an example of a call establishment.

```
D/PhoneUtils: setAudioMode()...OFFHOOK
D/CallManager: setAudioMode useInCallMode = false, Baseband = csfb
D/PhoneUtils: setAudioMode() no change: MODE_IN_CALL
D/Ringer: stopRing()...
D/Ringer: - stopRing: null mRingHandler!
W/Vibrator: Vibrator.cancel()
D/CallNotifier: - posting UPDATE_IN_CALL_NOTIFICATION request...
D/PhoneUtils: - hangup(Call): regular hangup()...
D/PhoneUtils: ==> hangup = true
D/PhoneApp: pokeUserActivity()...
D/PhoneUtils: Not supported: BargeIn
D/PhoneUtils: ConnectionHandler: updating mute state for each connection
D/PhoneUtils: setMuteInternal: using setMicrophoneMute(true)...
```

Snippet 5.2 - Logfile information: Connected call with microphone mute

Similarly, it is also possible to check if a call has ended through the "need to play *CALL_ENDED* tone" phrase, as well as, if a call was not answered or if an active call dropped. This information can be useful and substitute partially the anomaly's page report since the information is automatically retrieved by parsing and analyzing the Android logfile. However, this can only be used on rooted phones and many specific manufacturers' device messages may be different from phone to phone.

5.3.4 OTT Apps Analysis

In section 2.6, several solutions were described as supporting OTT applications, like Facebook, Dropbox or Skype. It is not known what that "support" means exactly, but in order to try to obtain information about these Over the Top applications running almost on every device, the Android logfile was analyzed with special focus on finding the maximum possible information about these apps.

The messages presented in this subsection are from a Xiaomi Redmi 3S and may vary from phone to phone. Furthermore, the applications tested were YouTube, Facebook, Facebook messenger and WhatsApp. After opening any Android app, it is possible to see the app's package name appearing in the logfile, for example, *com.google.android.youtube* or *com.facebook.katana.*

The YouTube website has an analytics page, for its users easily check their subscriptions, likes or views. When a user opens a video from the YouTube's app it can be seen, in the logfile, an URL call to a specific API for handling those statistics. It is also possible to capture a lot of information

about the video, audio parameters such as *AudioFocus*, *stateUpdate*, *tansportControlFlags*, among others.

In the case of Facebook, the amount of the information that can be extracted from the logfile is extensive. Despite personal information is not seen, it is possible to see when a user clicks on the Facebook history button to check the chronology, or when the user checks a person profile page. Clicking on the messenger button from Facebook app shows a message with "Start proc" followed by the messenger Android package name, similarly to what happens when other app's are launched.

In the WhatsApp application, it's possible to see when a user opens a conversation with a contact filtering the Android logfile for the phrase "***Displayed*** *com.whatsapp/.Conversation:".* VoIP calls made through these applications have a lot more information, especially when comparing with voice calls made using the operator's network voice service. An example of a VoIP call made through the WhatsApp app and its in-call audio information can be seen in Appendix F.

This test scratched the surface, since many of the messages presented may vary from phone to phone, but there is now a clear idea of the information that can be extracted here. This information may be included in the application, supporting these well-known OTT apps or generating automatically events catching, for example, the percentage of dropped calls on the device.

Chapter 6

Conclusions and Future Work

The present chapter serves as a conclusion of the accomplished work, analyzing all the requisites that were fulfilled and making a retrospective of the same. Some possibilities for future improvement are also addressed, notably by updating the application.

This project aimed at the development of an application for Android smartphones in order to analyze the network performance and retrieve QoE and QoS metrics, complementing what is already done by operator's fixed and mobile probes, scattered across the network.

The Android operating system was chosen in detriment of the others, due to the high percentage of market share that it has at the moment and, equally important, because that market share tends to increase even more in the future.

We are very satisfied with the user interface design and with the app-user interaction, having always been followed the requirements and templates given by the usability team. However, to completely validate all the new features on the UI, it is still necessary to a study or inquire the end users.

Test's results are already being sent to the management system with the attached GPS coordinates, Wi-FI and mobile network information. All the SMSs tests were implemented and support was given for the test scheduling and recursion. Furthermore, following the requirements defined in section 3.1, the app is not allowing two tests that need the same resources, to run simultaneously. It is also not allowed to run an on-demand test, if there is already a scheduled test that would interfere with it.

For future work, was observed the need to review the voice tests implementation (answer, reject and hang up), because they only work, at the moment, on a rooted Samsung Galaxy S7, due to the simulation of input events through the *ADB shell.* Other desirable feature to implement regarding the voice tests are: support for new technologies like VoLTE/VoWiFi, registering, for example, on which network (mobile or Wi-Fi) an SMS or a call is sent/made.

The radiologs/snapshots feature is implemented, functional and automatically retrieving the same network events as the other ArQoS probes. Furthermore, the radiolog's *JSON* content is already structured and ready to be sent.

The connection with the management system is working well, as all the requests and notifications with a higher priority were implemented. The automatically change of APN for the results delivery is also tested and working when the application is running as *system app*. Moreover, due to time constraints, there are still some requests to be implemented, such as reset/reboot the probe, configuration of the radiologs/scanlogs time interval and enable/disable via the management system, among other internal processes like probe-generated service and equipment alarms, that ArQoS NG probes have implemented.

Other future feature to be added in our solution, which would help in the integration with the other ArQoS probes, in a non-user interaction scenario, is boot the device automatically when it starts to charge.

Finally, it is still needed to improve the application with the Android logfile information, which can give useful information about the state of an active call, OTT apps and new parameters to be added in the radiologs.

References

[1] **X. P. Kenechi Okeleke, M. Rogers**, The Mobile Economy 2017. 27 February 2017.

[2] "Improve Network Service Quality", Optimize customer experience and network performance, http://www.alticelabs.com/content/products/BR_ArQoS_ALB_EN.pdf, Accessed at February 2017.

[3] **A. Jayanthiladevi, H. Premlatha, and G. Nawaz**, "Analysis study of Seamless Integration and Intelligent Solution in any situation by the Future Advanced Mobile Universal Systems 4G-(FAMOUS 4G)", IEEE International Conference on Emerging Trends in VLSI, Embedded Systems, Nano Electronics and Telecommun.

[4] **Mora**, "2G, 3G, 4G or the evolution of mobile networks," 2013. [Online]. Available: http://empireone.com.au/2g-3g-4g-mobile-network-evolution/. Accessed at March 2017.

[5] Qualcomm, (2014). "The Evolution of Mobile Technologies: 1G to 2G to 3G to 4G LTE" [Online], Available: https://www.qualcomm.com/documents/evolution-mobile-technologies-1g-2g-3g-4g-lte. Accessed at March 2017.

[6] GSM: Architecture, Tutorial Point, http://www.tutorialspoint.com/gsm/gsm_architecture.htm, Accessed at March 2017.

[7] 3GPP Technical Specification (2013) Numbering, Addressing and Identification, TS 23,003 v11.6.0 Section 19.6, www.3gpp.org, Accessed at March 2017.

[8] **R. L. Aguiar**, Apontamentos da cadeira de Redes Móveis, Universidade de Aveiro, Slides 2016/1017

[9] Internet Resource, NEWCOM Deliverables 23.3: http://www.newcom-project.eu/images/Delivarables/D23.3-Secondreportontoolsandtheirintegrationontheexperimentalsetups.pdf

[10] **R. Singh, R. Chauhan,** "A Review Paper: Voice over Internet Protocol", International Journal of Enhanced Research in Management & Computer Applications, ISSN: 2319-7471 Vol. 3 Issue 1, January-2014.

[11] **G. Fettweis, M. Krondorf, S. Bittner**, GFDM—generalized frequency division multiplexing, in: 69th Vehicular Technology Conference, IEEE, 2009, pp. 1–4.

[12] **M. Mukherjee**, et al. Reduced out-of-band radiation-based filter optimization for UFMC systems in 5G, in: Wireless Communications and Mobile Computing Conference, IWCMC, 2015, pp. 1150–1155.

[13] **N. Van der Neut**, et al. PAPR reduction in FBMC systems using a smart gradient-project active constellation extension method, in: 21st International Conference on Telecommunications, ICT, 2014, pp. 134–139.

[14] **G. Wunder, S.A. Gorgani, S.S. Ahmed**, Waveform optimization using trapezoidal pulses for 5G random access with short message support, in: IEEE 16th International Workshop on Signal Proc.: Advances in Wireless Comm., 2015, pp. 76–80.

[15] Ericsson Mobility Report: On the Pulse of the Networked Society, Ericsson, 2016. Available at: https://www.ericsson.com/assets/local/mobility-report/documents/2016/ericsson-mobility-report-november-2016.pdf

[16] **F. Boccardi, R. W. Heath, A. E. Lozano, T. L. Marzetta, and P. Popovski**. Five disruptive technology directions for 5G. IEEE Communications Magazine, 52(2):74–80, 2014.

[17] **H. Elshaer, F. Boccardi, M. Dohler, and R. Irmer**. Downlink and uplink decoupling: A disruptive architectural design for 5G networks. In IEEE GLOBECOM, pages 1798–1803, 2014.

[18] Wireless World Research Forum, 2011. [Online] Available at: http://www.wwrf.ch/files/wwrf/content/files/ publications/outlook/Outlook7.pdf.

[19] **M. Mustaqim, K. Khan, M. Usman**, ”LTE advanced: Requirements and technical challenges for 4G cellular network”, Journal of Emerging Trends in *Computing* and Information Sciences, vol.3, Issue.5, pp. 665-671, May 2012.

[20] Altice Labs, "Support system operations", http://www.alticelabs.com/pt/sistemas_suporte_operacoes.html#garantia_servico, Accessed at February 2017.

[21] **H. Wang, S. Chen, H. Xu, M. Ai, and Y. Shi**, A Software Defined Decentralized Mobile Network Architecture toward 5G, IEEE Network March/April 2015.

[22] **S. Rasthofer**, "The Android Logging Service – A Dangerous Feature for User Privacy?" May 2013.

[23] **M. Rumney** (Agile Technologies), LTE and the Evolution to 4G Wireless, Design and Measurement Challenges: Wiley Publication, March 2013.

[24] *A One-Time Overview of Global 5G Initiatives as of the First Quarter of 2014*, Jun. 2014.

[25] **A. Gupta, R. Kumar Jha,** A Survey of 5G Network: Architecture and Emerging Technologies, IEEE August 7, 2015.

[26] **C. Welch**, “Before it took over smartphones, Android was originally destined for cameras" http://www.theverge.com/2013/4/16/4230468/android-originally-designed-for-cameras-before-smartphones” p. 1, 2013.

[27] IDC, “Smartphone OS Market Share, 2016 Q3.” [Online]. Available: http://www.idc.com/promo/smartphone-market-share/os. Accessed at May 2017.

[28] **S. Hill**, “Which smartphone OS wins 2016?” 2016. [Online]. Available: https://www.digitaltrends.com/mobile/best-smartphone-os/. Accessed at May 2017.

[29] Android website, "The Android History", https://www.android.com/versions/nougat-7-0/, Accessed at April 2017.

[30] **M. Mercato,** "A doce história do Android", AndroidPIT, 2015. Available at: http://www.androidpit.com.br/historia-do-android.

[31] Platform Versions, https://developer.android.com/about/dashboards/index.html Official Android website, Accessed at April 2017.

[32] **J. Hildenbrand**, "Everything you need to know about rooting your Android", androidcentral, 2016. [Online]. Available: http://www.androidcentral.com/root. Accessed at April 2017.

[33] **W. Gordon**, "Everything You Need to Know About Rooting Your Android Phone," *lifehacker*, 2013. [Online]. Available: http://lifehacker.com/5789397/the-always-up-to-date-guide-to-rooting-any-android-phone. Accessed at April 2017.

[34] **G. Sims**, "What is root," *androidauthority*, 2016. [Online]. Available: http://www.androidauthority.com/what-is-root-680584/. Accessed at April 2017.

[35] **Sergio**, "Explaining the behavior of an Android application: System apps vs Non-System apps," *ricston*, 2013. [Online]. Available: https://www.ricston.com/blog/explaining-behavior-android-application-system-apps-nonsystem-apps/. Accessed at May 2017.

[36] **H. Q. Raja**, "How To Install Any App As A System App On Android," *addictivetips*, 2012. [Online]. Available: http://www.addictivetips.com/mobile/how-to-install-any-app-as-system-app-on-android/. Accessed at May 2017.

[37] **Y. Park**, 5G Vision and Requirements, 5G Forum, Korea, Feb. 2014.

[38] 5GNOW, (5th Generation Non-Orthogonal Waveforms for Asynchronous Signaling) is a European collaborative Research Project Supported by the European Commission Within FP7 ICT Call 8, 2015.

[39] **M. Hatton**, The Global M2M Market in 2013. London, U.K.: Machina Research White Paper, Jan. 2013

[40] **E. Elkin**, "The Secret Value of VoLTE", TMCnet, April 2014.

[41] "4G VoLTE devices list in India – Updated (09-04-2017)," *newsmarkets*, 2017. [Online]. Available: http://newsmarkets.in/4g-volte-devices-in-india-updated/. Accessed at Frebuary 2017.

[42] **M. Wright**, "The Next Generation of Voice Calling Starts Today – Telstra VoLTE", September 2015.

[43] Taqua, "Top 10 Questions about VoWiFi," 2015.

[44] Fierce WirelessTech, "ACG Research: With VoWiFi, it's all about the economics", Oct. 2015.

[45] abc7ny, "Consumer reports: cell phone voice quality put to the test", 2015. [Online]. Available: http://abc7ny.com/technology/consumer-reports-cell-phone-voice-quality-put-to-the-test/788284/. Accessed at May 2017.

[46] **E. Malykhina**, "Why Is Cell Phone Call Quality So Terrible?," 2015. [Online]. Available: https://www.scientificamerican.com/article/why-is-cell-phone-call-quality-so-terrible. Accessed at May 2017.

[47] **M. Gikas**, “3 reasons voice quality on smart phones still sucks,” 2014. [Online]. Available: http://www.consumerreports.org/cro/news/2014/05/3-reasons-voice-quality-on-smart-phones-still-sucks/index.htm. Accessed at May 2017.

[48] **J. Hecht**, “Why Mobile Voice Quality Still Stinks—and How to Fix It,” 2014. [Online]. Available: http://spectrum.ieee.org/telecom/wireless/why-mobile-voice-quality-still-stinksand-how-to-fix-it. Accessed at May 2017.

[49] ITU-T Recommendation P.10/G.100-Amendment 2. Vocabulary for performance and quality of service. Recommendations of the ITU,Telecommunications Sector, 12 2012.

[50] **P. L. Callet, S. Möller, and A. Perkis,** “Qualinet white paper on defi- nitions of quality of experience,” in European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003). Wadern, Germany: Dagstuhl, 2013.

[51] **R. Schatz, T. Hoßfeld, L. Janowski, and S. Egger,** “From packets to people: Quality of experience as a new measurement challenge,” in Data Traffic Monitoring and Analysis, vol. 7754,

[52] **T. Yamazaki, M. Eguchi, T. Miyoshi and K. Yamori**, “Quality of Experience Modeling with Psychological Effect for Interactive Web Services”, 2014 IEEE Network Operations and Management Symposium (NOMS), pp.1-4, April 2014.

[53] **L. Munroe**, “Here’s how Mesosphere built a process for regular, efficient usability tests,” 2016. [Online]. Available: https://mesosphere.com/blog/2016/06/30/design-usability-tests. Accessed at May 2017.

[54] **A. Bhattacharya**, “Android just hit a record 88% market share of all smartphones,” 2016. [Online]. Available: https://qz.com/826672/android-goog-just-hit-a-record-88-market-share-of-all-smartphones/. Accessed at May 2017.

[55] GSMArena, “Xiaomi Redmi 3s.” [Online]. Available: http://www.gsmarena.com/xiaomi_redmi_3s-8150.php. Accessed at May 2017.

[56] GSMArena, “Samsung Galaxy S7.” [Online]. Available: http://www.gsmarena.com/samsung_galaxy_s7-7821.php. Accessed at May 2017.

[57] **H. Schulzrinne**, “A comparison of SIP and H.323 for internet telephony”, Department of computer science, Columbia University New York.

[58] **S. K. Sonkar, R. Singh, R. Chauhan, A. Singh** “A Review Paper: Security on Voice over Internet Protocol from Spoofing attacks”, International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 3, May 2012.

[59] **A. Kund, I. S. Misra, S.K. Sanyal, and S. Bhunia, "**VoIP Performance over Broadband Wireless Networks under Static and Mobile Environments." International Journal of Wireless & Mobile Networks, 2010. 2 (4): p. 82-93.

[60] Android website, "Media Player", https://developer.android.com/guide/topics/media/mediaplayer.html. Accessed at March 2017.

[61] **A. S. Kamel,** "Real time performance evaluation of voice over IP call quality under varying network conditions", International Journal of Applied Science and Technology Vol. 1 No. 6; November 2011.

[62] **M. K. Muller, S. Schwarz and M. Rupp,** "QoS investigation of proportional fair scheduling in LTE networks", IFIP 2013,Valencia, pp.1-4, Nov 2013

[63] **R. Mugisha , N. Ventura,** "Packet scheduling for VOIP over LTE-A", Conference Publications of IEEE AFRICON 2013, Mauritius, Sept. 2013

[64] **P. Callet, S. Möller, A. Perkis,** Qualinet White paper on Definitions of Quality of Experience (QoE), 2012

[65] **A. Chang,** "How HD Voice works to make your calls sound drastically better." [Online]. Available: https://www.wired.com/2013/04/how-hd-voice-works-to-make-your-calls-clearer/, Accessed at April 2017.

[66] **C. Hoffman**, "Is It Illegal To Root Your Android or Jailbreak Your iPhone?," 2014. [Online]. Available: http://www.makeuseof.com/tag/illegal-root-android-jailbreak-iphone/. Accessed at April 2017.

[67] **V. Tikhvinskiy and G. Bochechka**, ''Perspectives and quality of service requirements in 5G networks,'' J. Telecommun. Inf. Technol., no. 1, pp. 23–26, 2015.

[68] **R. Shaikh,** "Test reveals 3x more battery life time with Android Marshmallow", Wccftech. http://wccftech.com/android-marshmallow-battery-tests/. Accessed at March 2017.

[69] Android website, "Supported Media Formats", https://developer.android.com/guide/topics/media/media-formats.html. Accessed at April 2017.

[70] Netmarketshare, "Mobile/Tablet Operating System Market Share", https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1. Accessed at April 2017.

[71] Paessler, PRTG Network Monitor, https://www.br.paessler.com/apps. Accessed at March 2017.

[72] Streambow, Xperience system, http://streambow.com/#xperiencesystem. Accessed at March 2017.

[73] Accuver, XCAL-Ranger, http://accuver.com/?page=products%7Cviewpage%7Cxcal-ranger. Accessed at March 2017.

[74] InfoVista, TEMS Pocket, http://www.tems.com/products-for-radio-and-core-networks/radio-network-engineering/portable-testing-for-wireless-networks#whats-new. Accessed at March 2017.

[75] VIAVI, TrueSite Solution, http://www.viavisolutions.com/sites/default/files/technical-library-items/truesite-solution-truesite-matrix-and-truesite-handheld-provide-scalable-mobile-network-and-service.pdf. Accessed at March 2017.

[76] Focus infocom, XGMA SP, http://focus-infocom.de/measurement-systems/benchmark-systems/xgma-sp. Accessed at March 2017.

[77] Meritech, Sigma-ML, https://www.meritechsolutions.com/Product_SigmaMl.html. Accessed at March 2017.

[78] Rohde-Schwarz, SwissQual QualiPoc Android, https://www.rohde-schwarz.com/us/product/qualipoc_android-productstartpage_63493-55430.html. Accessed at March 2017.

[79] Telco antennas, "Poor Mobile Network Coverage Explained - Weak Signal", https://www.telcoantennas.com.au/site/poor-mobile-network-coverage-explained-weak-signal. Accessed at May 2017.

[80] Open Handset Alliance, [Online]. https://www.openhandsetalliance.com/index.html. Accessed at June 2017.

[81] Google, [Online]. https://www.google.com. Accessed at June 2017.

[82] Bluetooth, Bluetooth Technology Website [Online]. https://www.bluetooth.com/. Accessed at June 2017.

[83] SQLite, SQLite Home Page [Online]. https://www.sqlite.org/. Accessed at June 2017.

[84] PESQ, [Online]. http://www.pesq.org/. Accessed at June 2017.

[85] POLQA, [Online]. http://www.polqa.info/. Accessed at June 2017.

[86] Battery Historian, [Online]. https://github.com/google/battery-historian. Accessed at May 2017.

[87] Android website, "Class Index", https://developer.android.com/reference/classes.html. Accessed at April 2017.

[88] ReactiveX, RxAndroid, [Online] https://github.com/ReactiveX/RxAndroid. Accessed at March 2017.

[89] Requery, [Online] https://github.com/requery/requery. Accessed at March 2017.

APPENDIX A

5G Related Activities

Research Project / Institutions / Research Groups	**Research area**	**HTTP location**
5GNOW (5th Generation Non-Orthogonal Waveforms for asynchronous signaling)	Non-orthogonal waveforms	http://www.5gnow.eu/
5Ci PPP (5G Infrastructure Public Private Partnership)	Next generation of communication networks, ubiquitous super-fast connectivity	http://5g-ppp.cu
COMBO (Convergence of fixed and Mobile Broadband access aggregation networks)	Fixed Mobile Converged (FMC) broadband access/aggregation networks	http://www.ict-combo.eu/
iJOIN (Interworking and JOINT Design of an Open Access and Backhaul Network	RAN-as-a-Service, radio access based upon small cells, and a heterogeneous backhaul	http://www.ict-ijoin.cu/
MAMMOET (MAssive MiMO for Efficient Transmission)	Massive MIMO	http:// www.mammoct-projcct.cu
METIS (Mobile and wireless communications Enablers for Twenty-twenty (2020) Information Society)	Provide a holistic framework 5G system concept	https://www.mctis2020.com'
MCN (MobileCloud Networking)	Mobile Network. Decentralized Computing, Smart Storage	http://www.mobile-cloud-networking.eu/site/'
MOTO (Mobile Opportunistic Traffic Offloading)	Traffic offloading architecture	http://www.ict-ras.eu/index .php/ras-projects./moto
PHYLAWS (PHYsical Layer Wireless Security)	Security approaches for handsets and communications nodes	http://www.phylawvict.org/
TROPIC (Traffic Optimization by the Integration of Information and Control)	Femtocell networking and cloud computing	http://www.ict4ropic.cu/
5GrEEn	Environmentally friendly 5G mobile network	https://www.eiticlabs.cu/news-events/news/article/toward-green-5g-mobile-networks-5green-new- project- launched/#allView
University of Edinburgh	Indoor wireless communications capacity	www.ed.ac.uk
University of Surrey 5G Innovation Centre (5GIC)	Lowering network costs. Anticipating user data needs to pre-allocate resources. Dense small cells. Device-to- device communication. Spectrum sensing (for unlicensed spectrum)	http://www.surrey.ac.uk/5gic/

Table 1 - 5G related activities in Europe [24]

Research Project / Institutions / Research Groups	Research area	HTTP location
Berkeley SWARM Lab	Third layer of information acquisition, synchrony to the Cloud, pervasive wireless networking, novel ultra-low power technologies.	https://swarmlub.eecs.berkeley.edu/letter-executive-director#overlay context=node/5/panel_content
Berkeley Wireless Research Center (BWRC)	Radio Frequency (RF> and Millimeter Wave (mmWave) technology. Advanced Spectrum Utilization. Energy Efficient Systems and ocher Integrated Wireless Systems and Applications	http://bwrc.eecs.berkeley.edu/
Broadband Wireless Access **&** Applications Center (BWAC)	Opportunistic spectrum access and allocation technologies. Millimeter wave wireless. Wireless cyber security, Cognitive sensor networks of heterogeneous devices. Image and video compression technologies. 1C and low-power design for broadband access/applications	https://bwac.arizona.edu/
Center for Wireless Systems and Applications (CWSA) at Purdue University	Devices and materials. Low power electronics. Communications. Networking. Multimedia traffic. Security	http://cwsawcb.ecn.purduc.edu/
ChoiceNet Project	Architectural design for the Internet of the near future	https://code.renci.org/gf/projecl/choicen el/
Clean Slate Project at Stanford University for research on Software-Defined Networking (SDN)	Open Flow. Software Defined Networking, and the Programmable Open Mobile Internet	http://clcanslate.stanfiird.edu/
expressive Internet Architecture (XIA) Project	Internet Architecture	http://www.cs.cmu.cdu/-xia/
Intel Strategic Research Alliance (ISRA)	Enabling new spectrum, improving spectral efficiency and spectral reuse, intelligent use of multiple radio access technologies, and use of context awareness to improve quality of service and wireless device power efficiency.	https://www.intel-university-collaboration.net/focused-research
Joint University of Texas, Austin and Stanford Research on 5G Wireless	New architectures for dense access infrastructure	http://www.ccc.utcxas.edu/news/profs-dc-veciana-shakkottai-and-collaborators-received-nsf-grant-work-5g-wireles$-networks
Mobility First Project	Architecture centered on mobility	http://mobililyfirM.winlab.nilgers.edu/
Named Data Network (NDN) Project	Named Data Networking (NDN) architecture	http://namcd-data.net
NEBULA Project	Cloud computing data centers	http://nebula-fia.org
NSF Communications & Information Foundation (CIF)	secure and reliable communications	http://www.nsf.gov/funding/pgm_summ.jsp?pimsJd=503300&org=CISE
NSF Computer **&** Network Systems (CNS)	enterprise, core, and optical networks; peer-to-peer and application- level networks; wireless, mobile, and cellular networks; networks for physical infrastructures; and sensor networks	http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503307&org=CISE
NSF Extreme Densification of Wireless Networks	Network densification	http://www.nsf.gov/awardsearch/show Award?AWD_H>= 1343383&HistoricalAwarxls=false
NSF Future Internet Architectures (FIA) Program	Named Data Network (NDN). Mobility First. Nebula. Expressive Internet Architecture (XIA). ChoiceNet	http://www.nets-fia.net/
NSF Grant for Evaluation of 60 GHz Band Communications	Millimeter wave picocells	
Polytechnic Institute of New York University (NYU-Poly) Program	Smart and more cost effective wireless infrastructure	http://engi neeri ng. nyu .edu/
Qualcomm Institute	Robust wireless communication, multimedia communication systems, and devices for next-generation communication, wireless health	http://csro.calit2.net/proposals.html
UCSD Center for Wireless Communications	Low-power circuitry, smart antennas, communication theory, communication networks, and multimedia applications	http://cwc.ucsd.edu/rcscarch/focusarcas.php
Wireless OMIT Center	Spectrum and connectivity, mobile applications, security and privacy and low power systems	http://wireless.csail.mit.edu
Wireless Virginia Tech	Cognitive Radio Networks. Digital Signal Processing. Social Networks. Autonomous Sensor. Communication. Antennas. Very Large Scale Integration	http://wirclcss.vi.edu

Table 2 - 5G related activities in America [24]

Research Project / Research Groups	Research area	HTTP location
IMT-2020 PROMOTION GROUP	5G research and development	http://www.imt-2020.cn/en/introduction
MOST (MINISTRY OF SCIENCE & TECHNOLOGY) 5G PROJECT	Radio-access-network (RAN) architecture. Massive MIMO	http://www.mo8t.gov.cn/eng/programmes1/

Table 3 - 5G related activities in Asia [24]

APPENDIX B

ArQoS NG probe's task output parameters examples

Disassociate WiFi				
Parameter	**Description**	**Format Type**	**Param Type**	**Values**
1	Session duration.	Float	Seconds	Time interval
2	The total number of packets received by this interface.	Int	Packets	Packet count
3	The total number of bytes received by this interface.	Int	bytes	Traffic volume
4	The total number of packets transmitted by this interface.	Int	Packets	Packet count
5	The total number of bytes transmitted by this interface.	Int	Bytes	Traffic volume
6	Connection quality.	Int	%	0 to 100
7	Received signal strength indicator.	int	DBm	RSSI
8	Measured noise level.	Int	DBm	Noise level
9	Signal to noise ratio.	Int	dB	SNR

Table 4 - ArQoS NG probe: Disassociate Wi-Fi task output parameters

Portal login				
Parameter	**Description**	**Format Type**	**Param Type**	**Values**
1	Target URL of authentication process redirection	string	URL	URL
2	Authentication page load time	float	seconds	Time interval
3	Total portal authentication time	float	Seconds	Time interval
4	Authentication response code (form-based authentication only)	int	NumerIcal ID	http response code
5	Authentication response human readable message	string	text	text
6	Name of the file with the downloaded web portal content	string	file	Empty – no content saved File name – saved accessed web portal content
7	Name of the file with the authentication responce	string	file	Empty – no content saved File name – saved server response
8	Packet capture file name	string	file	Empty- no packet capture performed File name – packet capture was saved with the indicated name

Table 5 - ArQoS NG probe: Disassociate Wi-Fi task output parameters

Receive SMS message				
Parameter	**Description**	**Format Type**	**Param Type**	**Values**
1	End to end message delivery time.	int	seconds	time interval
2	Received SMS message text.	string	text	text
3	SMSC message timestamp.	string	text	human readable date and time
4	Sender number.	string	MSI SDN	MSI SDN
5	SMSC number.	string	MSI SDN	MSI SDN
6	End to end message delivery time.	int	milliseconds	time interval
7	Time spent waiting for the received message since the beginning of the task.	int	milliseconds	time interval
8	SMS encoding.	int	numerical ID	- default (GSM 7-bit) - 8-bit - USC2 (16-bit)

Table 6 - ArQoS NG probe: Receive SMS task output parameters

Ping				
Parameter	**Description**	**Format Type**	**Param Type**	**Values**
1	Minimum measured ICMP latency.	float	milliseconds	latency
2	Average measured ICMP latency.	float	milliseconds	latency
3	Maximum measured ICMP latency.	float	milliseconds	latency
4	Send ICMP echo request packet count.	int	packets	packet count
5	Send ICMP echo response packet count.	int	packets	packet count
6	ICMP echo packet loss.	int	packets	packet count
7	Packet capture file name.	string	file	empty - no packet capture performed file name - packet capture was saved with the indicated name
8	Destination IP address or name.	string	IP address	host name or address
9	IP address obtained from DNS lookup	string	IP address	server address
10	Operation log filename. File with additional information about the performed operation and any errors that might have occurred.	string	file	empty - no log file saved file name - log file name
11	Source IP address	string	IP address	address
12	Server name resolution time.	int	milliseconds	time interval

Table 7 - ArQoS NG probe: PING task output parameters

APPENDIX C

Radiolog with an associated event


```
{
    "radiolog": [
        {
            "module": 0,
            "iccid": "89351060000611821081",
            "imsi": "268069620647135",
            "timestamp": 1495631356,
            "mac": "861111039801587",
            "gps": "40.62965876,-8.64616829",
            "network": {
                "status": 0,
                "mode": 10,
                "cellid": "313120",
                "plmn": "MEO",
                "roaming": 0,
                "rxlevel": "-62",
                "mcc": 268,
                "mnc": 6,
                "pci": 4,
                "tac": "48040",
                "rsrp": -86,
                "rsrq": -6
            },
            "neighbours": [
                {
                    "pci": 11,
                    "rsrp": -188,
                    "rsrq": -28
                },
                {
                    "pci": 3,
                    "rsrp": -186,
                    "rsrq": -28
                }
            ],
            "event": {
                "type": 3,
                "origin": "old plmn: NOWO, new plmn: MEO"
            }
        }
    ]
}
```


Snippet 1 - Radiolog with an associated event

APPENDIX D

Used devices specifications

Xiaomi Redmi 3S		
NETWORK	Technology	GSM / HSPA / LTE
LAUNCH	Announced	2016, June
	Status	Available. Released 2016, June
BODY	Dimensions	139.3 x 69.6 x 8.5 mm (5.48 x 2.74 x 0.33 in)
	Weight	144 g (5.08 oz)
	SIM	Dual SIM (Micro-SIM/Nano-SIM, dual stand-by)
DISPLAY	Type	IPS LCD capacitive touchscreen, 16M colors
	Size	5.0 inches (~71.1% screen-to-body ratio)
	Resolution	720 x 1280 pixels (~294 ppi pixel density)
	Multitouch	Yes
		- MIUI 8
PLATFORM	OS	Android 6.0.1 (Marshmallow)
	Chipset	Qualcomm MSM8937 Snapdragon 430
	CPU	Octa-core 1.4 GHz Cortex-A53
	GPU	Adreno 505
MEMORY	Card slot	microSD, up to 256 GB (uses SIM 2 slot)
	Internal	16 GB, 2 GB RAM
CAMERA	Primary	13 MP, f/2.0, phase detection autofocus, LED flash, check quality
	Features	Geo-tagging, touch focus, face/smile detection, HDR, panorama
	Video	1080p@30fps, check quality
	Secondary	5 MP, f/2.2, 1080p
SOUND	Alert types	Vibration; MP3, WAV ringtones
	Loudspeaker	Yes
	3.5mm jack	Yes
		- Active noise cancellation with dedicated mic
COMMS	WLAN	Wi-Fi 802.11 b/g/n, Wi-Fi Direct, hotspot
	Bluetooth	4.1, A2DP
	GPS	Yes, with A-GPS, GLONASS, BDS
	Infrared port	Yes
	Radio	FM radio
	USB	microUSB 2.0
FEATURES	Sensors	Accelerometer, gyro, proximity, compass
	Messaging	SMS(threaded view), MMS, Email, Push Mail, IM
	Browser	HTML5
	Java	No
		- Fast battery charging

		- DivX/Xvid/MP4/H.264 player - MP3/WAV/eAAC+/FLAC player - Photo/video editor
BATTERY		Non-removable Li-Ion 4100 mAh battery
MISC	Colors	Gold, Dark Gray, Silver
	SAR EU	0.62 W/kg (head) 0.43 W/kg (body)
	Price	About 120 EUR
	Performance	Basemark OS II: 882 / Basemark OS II 2.0: 882
	Display	Contrast ratio: 1087:1 (nominal), 2.913 (sunlight)
	Camera	Photo / Video
	Loudspeaker	Voice 66dB / Noise 70dB / Ring 70dB
	Audio	Noise -94.3dB / Crosstalk -91.8dB
	Battery life	Endurance rating 104h

Table 8 - Xiomi Redmi 3S specifications [55]

Samsung Galaxy S7		
NETWORK	Technology	GSM / HSPA / LTE
LAUNCH	Announced	2016, February
	Status	Available. Released 2016, March
BODY	Dimensions	142.4 x 69.6 x 7.9 mm (5.61 x 2.74 x 0.31 in)
	Weight	152 g (5.36 oz)
	Build	Corning Gorilla Glass 4 back panel
	SIM	Single SIM (Nano-SIM) - G930F Dual SIM (Nano-SIM, dual stand-by) - G930FD
		- Samsung Pay (Visa, MasterCard certified) - IP68 certified - dust/water proof over 1.5 meteres and 30 minutes
DISPLAY	Type	Super AMOLED capacitive touchscreen, 16M colors
	Size	5.1 inches (~72.1% screen-to-body ratio)
	Resolution	1440 x 2560 pixels (~577 ppi pixel density)
	Multitouch	Yes
	Protection	Corning Gorilla Glass 4
		- Always-on display - TouchWiz UI
PLATFORM	OS	Android 6.0 (Marshmallow), upgradable to 7.0 (Nougat)
	Chipset	Exynos 8890 Octa
	CPU	Octa-core (4x2.3 GHz Mongoose & 4x1.6 GHz Cortex-A53)
	GPU	Mali-T880 MP12
MEMORY	Card slot	microSD, up to 256 GB (dedicated slot) - single-SIM model (G930F, G930W8) microSD, up to 256 GB (uses SIM 2 slot) - dual-SIM model (G930FD)
	Internal	32/64 GB, 4 GB RAM
CAMERA	Primary	12 MP, f/1.7, 26mm, phase detection autofocus, OIS, LED flash

	Features	1/2.5" sensor size, 1.4 µm pixel size, geo-tagging, simultaneous 4K video and 9MP image recording, touch focus, face/smile detection, Auto HDR, panorama
	Video	2160p@30fps, 1080p@60fps, 720p@240fps, HDR, dual-video rec
	Secondary	5 MP, 1/4.1" sensor size, 1.34 µm pixel size, f/1.7, 22mm, dual video call, Auto HDR
SOUND	Alert types	Vibration; MP3, WAV ringtones
	Loudspeaker	Yes
	3.5mm jack	Yes
		- 24-bit/192kHz audio - Active noise cancellation with dedicated mic
COMMS	WLAN	Wi-Fi 802.11 a/b/g/n/ac, dual-band, Wi-Fi Direct, hotspot
	Bluetooth	4.2, A2DP, LE, aptX
	GPS	Yes, with A-GPS, GLONASS, BDS
	NFC	Yes
	Radio	No
	USB	microUSB 2.0, USB Host
FEATURES	Sensors	Fingerprint (front-mounted), accelerometer, gyro, proximity, compass, barometer, heart rate, SpO2
	Messaging	SMS(threaded view), MMS, Email, Push Mail, IM
	Browser	HTML5
	Java	No
		- Fast battery charging (Quick Charge 2.0) - Qi/PMA wireless charging (market dependent) - S-Voice natural language commands and dictation - OneDrive (115 GB cloud storage) - MP4/DivX/XviD/WMV/H.264 player - MP3/WAV/WMA/eAAC+/FLAC player - Photo/video editor
BATTERY		Non-removable Li-Ion 3000 mAh battery
	Talk time	Up to 22 h (3G)
	Music play	Up to 62 h
MISC	Colors	Black, White, Gold, Silver, Pink Gold
	SAR	1.40 W/kg (head) 1.59 W/kg (body)
	SAR EU	0.41 W/kg (head) 0.62 W/kg (body)
	Price	About 500 EUR
TESTS	Performance	Basemark OS II: 2004 / Basemark OS II 2.0: 2128
	Display	Contrast ratio: Infinite (nominal), 4.376 (sunlight)
	Camera	Photo / Video
	Loudspeaker	Voice 69dB / Noise 69dB / Ring 71dB
	Audio	Noise -92.5dB / Crosstalk -92.7dB
	Battery life	Endurance rating 80h

Table 9 - Samsung Galaxy S7 specifications [56]

APPENDIX E

Logfile's VoIP call information

```
I/ActivityManager: Displayed com.whatsapp/.VoipActivity: +254ms

Audio: D/audio_hw_primary: adev_open_input_stream: enter: sample_rate(16000) channel_mask(0x10) devices(0x80000004) stream_handle(0xab5adf68) io_handle(772) source(7)

D/compress_voip: voice_extn_compress_voip_pcm_prop_check: VoIP PCM property is enabled

D/compress_voip: voice_extn_compress_voip_open_input_stream: enter

D/compress_voip: voip_set_mode: enter, format=1

D/compress_voip: voip_set_mode: Derived mode = 12

I/AudioFlinger: AudioFlinger's thread 0xef070008 ready to run

D/audio_hw_primary: in_standby: enter: stream (0xab5adf68) usecase(27: compress-voip-call)

W/AudioFlinger::EffectModule: EffectModule 0xab54ad00 destructor called with unreleased interface

W/AudioFlinger::EffectHandle: disconnect Effect handle 0xab523ae0 disconnected after thread destruction

D/audio_hw_primary: adev_close_input_stream: enter:stream_handle(0xab5adf68)

D/compress_voip: voice_extn_compress_voip_close_input_stream: enter

D/compress_voip: voip_stop_call: enter, out_stream_count=0, in_stream_count=0

E/compress_voip: voip_stop_call: Could not find the usecase (27) in the list

E/audio_hw_primary: adev_close_input_stream: Compress voip input cannot be closed, error:-22

I/WhatsAppJni: EnsureThreadAttached: attached current thread to JVM

D/audio_hw_extn: audio_extn_get_parameters: returns

I/str_params: key: 'audio_mode' value: ''

D/audio_hw_extn: audio_extn_get_parameters: returns

I/str_params: key: 'voip_out_stream_count' value: ''

D/audio_hw_extn: audio_extn_get_parameters: returns

I/str_params: key: 'voip_sample_rate' value: ''

D/AudioPolicyManagerCustom: Set VoIP and Direct output flags for PCM format

I/AudioFlinger: openOutput(), module 1 Device 1, SamplingRate 16000, Format 0x000001, Channels 1, flags 801

D/audio_hw_primary: adev_open_output_stream: enter: sample_rate(16000) channel_mask(0x1) devices(0x1) flags(0x801) stream_handle(0xab4677a0)

D/compress_voip: voice_extn_compress_voip_pcm_prop_check: VoIP PCM property is enabled

D/compress_voip: voice_extn_compress_voip_open_output_stream: enter

D/compress_voip: voip_set_mode: enter, format=1

D/compress_voip: voip_set_mode: Derived mode = 12

D/audio_hw_primary: adev_open_output_stream: Stream (0xab4677a0) picks up usecase (compress-voip-call) ...

I/AudioFlinger: AudioStreamOut::open(), mHalFormatIsLinearPcm = 1

I/AudioFlinger: HAL output buffer size 320 frames, normal sink buffer size 320 frames

I/AudioFlinger: AudioFlinger's thread 0xab539d50 ready to run

V/AudioPolicyManagerCustom: getOutput() returns new direct output 776

V/SRS_Proc: ParamSet string: bluetooth_enabled=0

V/SRS_ProcWS: SRS_Processing - SourceOutAdd - No Available Slot for 0xab539d50

D/audio_hw_primary: out_standby: enter: stream (0xab4677a0) usecase(27: compress-voip-call)

D/audio_hw_primary: out_standby: Ignore Standby in VOIP call

W/AudioRecord: AUDIO_INPUT_FLAG_FAST denied by client; transfer 1, track 16000 Hz, primary 48000 Hz

D/audio_hw_primary: adev_open_input_stream: enter: sample_rate(16000) channel_mask(0x10) devices(0x80000004) stream_handle(0xab55a8b8) io_handle(778) source(7)
```

Snippet 2 - Logfile information: WhatsApp VoIP call information

APPENDIX F

Loaded Test Status Request: Probe response


```
{
    "probenotification": {
        "associatedResponse": {
            "data": [
                {
                    "brokenIterations": 0,
                    "datafim": "20200717133554",
                    "dataini": "20170531141300",
                    "endevent": 0,
                    "executedIterations": 0,
                    "internalState": 0,
                    "iterationMaxDuration": 40,
                    "macrosPerIteration": 1,
                    "modulo": 0,
                    "skippedIterations": 0,
                    "startevent": 7,
                    "state": 1,
                    "tasksPerIteration": 2,
                    "testeid": "T5",
                    "testname": "Send 2 SMS SEM METADATA"
                },
                {
                    "brokenIterations": 0,
                    "datafim": "20200717133554",
                    "dataini": "20170424154100",
                    "endevent": 0,
                    "executedIterations": 0,
                    "internalState": 0,
                    "iterationMaxDuration": 20,
                    "macrosPerIteration": 1,
                    "modulo": 0,
                    "skippedIterations": 0,
                    "startevent": 7,
                    "state": 1,
                    "tasksPerIteration": 1,
                    "testeid": "T7",
                    "testname": "PTEmpresas Browsing"
                },
                {
                    "brokenIterations": 0,
                    "datafim": "20200717133554",
                    "dataini": "20170424154100",
                    "endevent": 0,
                    "executedIterations": 0,
                    "internalState": 0,
                    "iterationMaxDuration": 20,
                    "macrosPerIteration": 1,
                    "modulo": 0,
                    "skippedIterations": 0,
                    "startevent": 7,
                    "state": 1,
                    "tasksPerIteration": 1,
                    "testeid": "T8",
                    "testname": "MEO Music Browsing"
                },
                 …
            ]
        },
        "equipmenttype": "13",
        "ipaddress": "172.21.77.130",
        "macaddress": "357810082121048",
        "msg_type": 254,
        "serialnumber": "357810082121048",
        "timestamp": "20170705154234"
    }
}
```


Snippet 3 - Get probe's loaded tests status request